

Security and Privacy for Edge-Assisted Internet of Things

Lead Guest Editor: Ding Wang

Guest Editors: Qi Jiang, Shi-feng Sun, and Chunhua Su





Security and Privacy for Edge-Assisted Internet of Things

Security and Communication Networks

Security and Privacy for Edge-Assisted Internet of Things

Lead Guest Editor: Ding Wang

Guest Editors: Qi Jiang, Shi-feng Sun, and Chunhua
Su





Copyright © 2023 Hindawi Limited. All rights reserved.

This is a special issue published in "Security and Communication Networks." All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Chief Editor

Roberto Di Pietro, Saudi Arabia

Associate Editors

Jiankun Hu , Australia
Emanuele Maiorana , Italy
David Megias , Spain
Zheng Yan , China

Academic Editors

Saed Saleh Al Rabae , United Arab Emirates
Shadab Alam, Saudi Arabia
Goutham Reddy Alavalapati , USA
Jehad Ali , Republic of Korea
Jehad Ali, Saint Vincent and the Grenadines
Benjamin Aziz , United Kingdom
Taimur Bakhshi , United Kingdom
Spiridon Bakiras , Qatar
Musa Balta, Turkey
Jin Wook Byun , Republic of Korea
Bruno Carpentieri , Italy
Luigi Catuogno , Italy
Ricardo Chaves , Portugal
Chien-Ming Chen , China
Tom Chen , United Kingdom
Stelvio Cimato , Italy
Vincenzo Conti , Italy
Luigi Coppolino , Italy
Salvatore D'Antonio , Italy
Juhriyansyah Dalle, Indonesia
Alfredo De Santis, Italy
Angel M. Del Rey , Spain
Roberto Di Pietro , France
Wenxiu Ding , China
Nicola Dragoni , Denmark
Wei Feng , China
Carmen Fernandez-Gago, Spain
AnMin Fu , China
Clemente Galdi , Italy
Dimitrios Geneiatakis , Italy
Muhammad A. Gondal , Oman
Francesco Gringoli , Italy
Biao Han , China
Jinguang Han , China
Khizar Hayat, Oman
Azeem Irshad, Pakistan

M.A. Jabbar , India
Minho Jo , Republic of Korea
Arijit Karati , Taiwan
ASM Kayes , Australia
Farrukh Aslam Khan , Saudi Arabia
Fazlullah Khan , Pakistan
Kiseon Kim , Republic of Korea
Mehmet Zeki Konyar, Turkey
Sanjeev Kumar, USA
Hyun Kwon, Republic of Korea
Maryline Laurent , France
Jegatha Deborah Lazarus , India
Huaizhi Li , USA
Jiguo Li , China
Xueqin Liang, Finland
Zhe Liu, Canada
Guangchi Liu , USA
Flavio Lombardi , Italy
Yang Lu, China
Vincente Martin, Spain
Weizhi Meng , Denmark
Andrea Michienzi , Italy
Laura Mongioi , Italy
Raul Monroy , Mexico
Naghme Moradpoor , United Kingdom
Leonardo Mostarda , Italy
Mohamed Nassar , Lebanon
Qiang Ni, United Kingdom
Mahmood Niazi , Saudi Arabia
Vincent O. Nyangaresi, Kenya
Lu Ou , China
Hyun-A Park, Republic of Korea
A. Peinado , Spain
Gerardo Pelosi , Italy
Gregorio Martinez Perez , Spain
Pedro Peris-Lopez , Spain
Carla Ràfols, Germany
Francesco Regazzoni, Switzerland
Abdalhossein Rezai , Iran
Helena Rifà-Pous , Spain
Arun Kumar Sangaiah, India
Nadeem Sarwar, Pakistan
Neetesh Saxena, United Kingdom
Savio Sciancalepore , The Netherlands

De Rosal Ignatius Moses Setiadi ,
Indonesia
Wenbo Shi, China
Ghanshyam Singh , South Africa
Vasco Soares, Portugal
Salvatore Sorce , Italy
Abdulhamit Subasi, Saudi Arabia
Zhiyuan Tan , United Kingdom
Keke Tang , China
Je Sen Teh , Australia
Bohui Wang, China
Guojun Wang, China
Jinwei Wang , China
Qichun Wang , China
Hu Xiong , China
Chang Xu , China
Xuehu Yan , China
Anjia Yang , China
Jiachen Yang , China
Yu Yao , China
Yinghui Ye, China
Kuo-Hui Yeh , Taiwan
Yong Yu , China
Xiaohui Yuan , USA
Sherali Zeadally, USA
Leo Y. Zhang, Australia
Tao Zhang, China
Youwen Zhu , China
Zhengyu Zhu , China

Contents


Deep Graph Embedding for IoT Botnet Traffic Detection

Bonan Zhang , Jingjin Li , Lindsay Ward , Ying Zhang , Chao Chen , and Jun Zhang 
Research Article (10 pages), Article ID 9796912, Volume 2023 (2023)

VNGuarder: An Internal Threat Detection Approach for Virtual Network in Cloud Computing Environment

Li Lin , Huanzeng Yang, Jing Zhan , and Xuhui Lv
Research Article (15 pages), Article ID 1242576, Volume 2022 (2022)







Cryptanalysis of an Efficient and Secure Certificateless Aggregate Signature-Based Authentication Scheme for Vehicular Ad Hoc Networks

Xiaodong Yang , Aijia Chen, Zhisong Wang, Xiaoni Du, and Caifen Wang
Research Article (12 pages), Article ID 4472945, Volume 2022 (2022)



A New Malware Detection Method Based on VMCADR in Cloud Environments

Luxin Zheng  and Jian Zhang 
Research Article (13 pages), Article ID 4208066, Volume 2022 (2022)






Cryptocurrency Mining Malware Detection Based on Behavior Pattern and Graph Neural Network

Rui Zheng , Qiuyun Wang , Jia He , Jianming Fu , Guga Suri , and Zhengwei Jiang 
Research Article (8 pages), Article ID 9453797, Volume 2022 (2022)




An Efficient Outlier Detection Approach for Streaming Sensor Data Based on Neighbor Difference and Clustering

Saihua Cai , Jinfu Chen , Baoquan Yin, Ruizhi Sun, Chi Zhang, Haibo Chen, Jingyi Chen, and Min Lin
Research Article (14 pages), Article ID 3062541, Volume 2022 (2022)

Blockchain-Empowered Secure and Privacy-Preserving Health Data Sharing in Edge-Based IoMT

Xueli Nie , Aiqing Zhang , Jindou Chen , Youyang Qu , and Shui Yu 
Research Article (16 pages), Article ID 8293716, Volume 2022 (2022)



A Novel Trusted Software Base for Commercial Android Devices Using Secure TF Card

Yuting Zhou , Bo Zhao , and Yang An 
Research Article (12 pages), Article ID 6731277, Volume 2022 (2022)


A Hybrid Opportunistic IoT Secure Routing Strategy Based on Node Intimacy and Trust Value

Lin Yu , Gang Xu , ZhiFei Wang , Na Zhang , and FengQi Wei 
Research Article (12 pages), Article ID 6343764, Volume 2022 (2022)

Lattice-Based Self-Enhancement Authorized Accessible Privacy Authentication for Cyber-Physical Systems

Jinhui Liu , Yong Yu , Houzhen Wang, and Huanguo Zhang
Research Article (9 pages), Article ID 8995704, Volume 2022 (2022)

Anomaly Detection of System Call Sequence Based on Dynamic Features and Relaxed-SVM

Xiaoyao Liao, Changzhi Wang , and Wen Chen




Research Article (13 pages), Article ID 6401316, Volume 2022 (2022)

Black-Box Adversarial Attacks against Audio Forensics Models

Yi Jiang  and Dengpan Ye 

Research Article (8 pages), Article ID 6410478, Volume 2022 (2022)

Network Embedding-Based Approach for Detecting Collusive Spamming Groups on E-Commerce Platforms

Jinbo Chao , Chunhui Zhao , and Fuzhi Zhang 



Research Article (13 pages), Article ID 4354086, Volume 2022 (2022)

Outsourced Mutual Private Set Intersection Protocol for Edge-Assisted IoT

Jing Zhang , Rongxia Qin , Ruijie Mu , Xiaojun Wang , and Yongli Tang 







Research Article (11 pages), Article ID 3159269, Volume 2021 (2021)

A Scalable Security Protocol for Intravehicular Controller Area Network

Zi-An Zhao , Yu Sun , Dawei Li, Jian Cui, Zhenyu Guan, and Jianwei Liu


Research Article (13 pages), Article ID 2314520, Volume 2021 (2021)

High-Capacity Robust Behavioral Steganography Method Based on Timestamp Modulation across Social Internet of Things

Mingliang Zhang , Xiangyang Luo , Pei Zhang , Hao Li , Yi Zhang , and Lingling Li 





Research Article (16 pages), Article ID 6351144, Volume 2021 (2021)

Database Padding for Dynamic Symmetric Searchable Encryption

Ruizhong Du, Yuqing Zhang , and Mingyue Li





Research Article (12 pages), Article ID 9703969, Volume 2021 (2021)

Gene Sequence Clustering Based on the Profile Hidden Markov Model with Differential Identifiability

Xujie Ren , Tao Shang , Yatong Jiang , and Jianwei Liu 


Research Article (9 pages), Article ID 8296417, Volume 2021 (2021)

Security and Privacy for Edge-Assisted Internet of Things Security Proof for the SKKE Protocol

Xiangyang Wang , Chunxiang Gu , Fushan Wei , and Siqi Lu 



Research Article (14 pages), Article ID 9029664, Volume 2021 (2021)

Achieve Efficient and Privacy-Preserving Compound Substring Query over Cloud

Fan Yin, Rongxing Lu , Yandong Zheng, and Xiaohu Tang

Research Article (13 pages), Article ID 7941233, Volume 2021 (2021)

Publicly Verifiable $M + 1$ -Price Auction Fit for IoT with Minimum Storage

Po-Chu Hsu  and Atsuko Miyaji 

Research Article (10 pages), Article ID 1615117, Volume 2021 (2021)






Contents

A Commitment Scheme with Output Locality-3 Fit for the IoT Device

Hideaki Miyaji , Yuntao Wang , Akinori Kawachi , and Atsuko Miyaji 


Research Article (10 pages), Article ID 2949513, Volume 2021 (2021)

NSGA-II-Based Granularity-Adaptive Control-Flow Attestation

Jing Zhan , Yongzhen Li , Yifan Liu , Hongchao Li , Shuai Zhang , and Li Lin 




Research Article (16 pages), Article ID 2914192, Volume 2021 (2021)

A Blockchain-Based CP-ABE Scheme with Partially Hidden Access Structures

Yang Ba, Xuexian Hu , Yue Chen, Zenghang Hao, Xuewei Li, and Xincheng Yan



Research Article (16 pages), Article ID 4132597, Volume 2021 (2021)

An Improved Coercion-Resistant E-Voting Scheme

Yuanjing Hao , Zhixin Zeng , and Liang Chang 

Research Article (11 pages), Article ID 5448370, Volume 2021 (2021)

Intelligent Intrusion Detection Based on Federated Learning for Edge-Assisted Internet of Things

Dapeng Man , Fanyi Zeng, Wu Yang, Miao Yu , Jiguang Lv, and Yijing Wang

Research Article (11 pages), Article ID 9361348, Volume 2021 (2021)

IoT-Based Autonomous Pay-As-You-Go Payment System with the Contract Wallet

Shinya Haga  and Kazumasa Omote 

Research Article (10 pages), Article ID 8937448, Volume 2021 (2021)

Multi-Authority Criteria-Based Encryption Scheme for IoT

Jianguo Sun , Yang Yang , Zechao Liu , and Yuqing Qiao 

Research Article (15 pages), Article ID 9174630, Volume 2021 (2021)

Research Article

Deep Graph Embedding for IoT Botnet Traffic Detection

Bonan Zhang ¹, Jingjin Li ¹, Lindsay Ward ², Ying Zhang ¹, Chao Chen ²,
and Jun Zhang ¹

¹School of Physics and Electronic Information, Yunnan Normal University, Kunming 650000, China

²College of Science and Engineering, James Cook University, Townsville QLD 4811, Australia

Correspondence should be addressed to Ying Zhang; yingzhang27@ynnu.edu.cn and Jun Zhang; junzhang@ynnu.edu.cn

Received 15 September 2021; Revised 6 March 2022; Accepted 23 May 2022; Published 25 October 2023

Academic Editor: Ding Wang

Copyright © 2023 Bonan Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Botnet attacks have mainly targeted computers in the past, which is a fundamental cybersecurity problem. Due to the booming of Internet of things (IoT) devices, an increasing number of botnet attacks are now targeting IoT devices. Researchers have proposed several mechanisms to avoid botnet attacks, such as identification by communication patterns or network topology and defence by DNS blacklisting. A popular direction for botnet detection currently relies on the specific topological characteristics of botnets and uses machine learning models. However, it relies on network experts' domain knowledge for feature engineering. Recently, neural networks have shown the capability of representation learning. This paper proposes a new approach to extracting graph features via graph neural networks. To capture the particular topology of the botnet, we transform the network traffic into graphs and train a graph neural network to extract features. In our evaluations, we use graph embedding features to train six machine learning models and compare them with the performance of traditional graph features in identifying botnet nodes. The experimental results show that botnet traffic detection is still challenging even with neural networks. We should consider the impact of data, features, and algorithms for an accurate and robust solution.

1. Introduction

Botnets are networks consisting of multiple compromised devices connected to the Internet [1]. These compromised devices can be used to perform malicious activities such as distributed denial-of-service attacks, data theft, and spamming [2, 3]. For botnet, it is not necessary to have a powerful host to accomplish complex tasks. To achieve the attack task, it needs to infect as many devices as possible. So many botnets are now targeting their infections on IoT devices. A number of IoT botnets have now emerged, such as BASHLITE [4], Carna [5], and Mirai [6]. One of them, Mirai, temporarily crippled Krebs with a denial-of-service attack in September 2016 [7]. Moreover, its attack volume exceeded 600 Gbps, one of the largest attacks on record. For IoT devices, their connection to the Internet makes it easier for botnets to spread. Furthermore, many IoT devices use initial passwords, making it easier for botnets to infect these devices. How to detect these botnet-infected IoT devices is the current key research problem.

From the perspective of structures, botnets can be classified as centralized command and control (C&C) structures or decentralised peer-to-peer (P2P) structures [8]. The bot header can direct the bot nodes efficiently for malicious activities through a hierarchical structure in centralized forms. The apparent hierarchical nature of this structure makes it easily detectable. To make botnets harder to detect, attackers are now building botnets using P2P structures. There are no hosts in the P2P structure for assigning tasks. The owner of the botnet can join any part of the botnet. Furthermore, any part of the botnet can control the entire botnet for attacks. This makes the detection of botnets more difficult than in the centralized model [9]. The focus of this paper is on detecting such P2P structured botnets.

In order to distinguish botnet traffic from background traffic, previous works mainly detect botnets from the following three categories: The first is botnet node detection by traffic patterns. This type of approach trains machine learning algorithms to find botnet traffic in the background

traffic by using traffic information, including communication time, port number, packet size, communication protocol, etc. as features. The second approach is to identify bot nodes by using prior knowledge, such as domain names and DNS blacklists. The attacker may evade both methods of detection by modifying the traffic packets. On the other hand, these two detection methods may also cause the leakage of user privacy. The last method is to distinguish botnet traffic from normal traffic by identifying the network topology. In previous studies, researchers have found that botnets often have particular topologies, such as mixing rates [10] and features of connection graph components [11, 12]. Centralized botnets manage individual nodes for attacks through a hierarchical structure. For botnets with a P2P structure, a high rate of mixing can often be found, as they need to pass information quickly to organise their attacks. On the other hand, to protect users from privacy breaches, more and more algorithms are now avoiding this problem by using federated learning. The details of each of these approaches will be discussed in the “Related Work.”

Machine learning has shown its success in various cybersecurity applications, such as malware detection [13, 14], cyber threats/incidents detection [15–18], and software vulnerability detection [19–21]. Researchers have also applied machine learning in the detection of botnet traffic. However, most existing detection models require a lot of detailed traffic information for analysis. These content-based features have many limitations in practical use. The traffic details may be encrypted or may even have been intentionally modified by the attacker. Now some models solve this problem by identifying the traffic topology [10]. However, these models are labour-intensive in defining topological features and perform multiple prefiltering steps. In practice, the models also need to be tuned to data characteristics. This project achieves graph feature extraction by using graph neural network model to identify the network topology. The advantage of using a GNN model is that the relationships between nodes in the network can be effectively identified. Nodes at each layer in the model pass information to their neighbours to exchange information and update their state. As the number of layers in the model increases, the nodes will contain information about their neighbours at more hops, thus enabling the identification of the network topology. In this project, we first trained a GNN (graph neural network) model for generating graph embedding data and then tested the performance of six machine learning models when using this data for detection.

To summarise, the contributions we made in this paper are as follows:

- (i) We provide detailed analysis of the algorithms previously used to detect botnet nodes and their suitable environments. The advantages and disadvantages of the different algorithms are summarised.
- (ii) A GNN model has been tailored to transform traffic data into graph embedding data. The model effectively captures the relationships between nodes and

provides a basis for identifying the topology of the network structure.

- (iii) We test and analyze the performance of six machine learning algorithms for detecting botnet nodes when using graph embedding data.

The remainder of the paper is structured as follows: in Section 2, we review the reason why IoT devices are vulnerable to botnet infection and state-of-the-art techniques that are used for detecting bot nodes. In Section 3, we present in detail how the data was generated and the GNN model we used to extract the graph embedding features. In Section 4, we describe the datasets and model evaluation criteria that were used for evaluating model performance, along with the evaluation results. We discuss the limitations of our work and point out some future directions in Section 5. Section 6 concludes our work.

2. Related Work

In this section we will first describe the new trend that IoT devices are compromised to become botnets. We will then review the three categories of algorithms currently in use to detect botnets. Each of these methods has its own applicable scenarios and drawbacks, and we will elaborate them in this section to provide researchers with a reference for their choice of algorithm. On the other hand in order to be able to detect anomalous nodes, machine learning algorithms will inevitably use the user’s private data for prediction. To address this problem, it has recently been proposed to avoid privacy breaches by using federated learning [22]. We will discuss this point in this section as well.

2.1. Botnet Attack from IoT Devices. DDoS attacks by infecting IoT devices have now become very common [6, 23]. The reasons are twofold. First, most of the IoT devices in widespread use today do not have powerful computing capabilities, so most IoT devices need to be connected to the Internet or other IoT devices to achieve more functionalities [24]. This structure makes IoT devices vulnerable to intrusion. Second, most IoT devices use insecure default passwords or simple passwords. For most consumers, they will probably only use a simple password or even just a default password when they purchase an IoT device [25]. For botnets aiming to infect IoT devices, it is possible to infect a large number of devices by simply brute-force logging in with randomly selected usernames and passwords from a list of preconfigured credentials [6, 26]. In cases where the password is changed to a simple password, it can also be easily guessed. The model designed by Wang et al. fits popular ciphers well and obtains a coefficient of determination greater than 0.97 [27]. In another paper, Wang et al. show that an attacker can achieve a success rate of up to 73% in 100 guesses against a regular user if he has some information about the user of the device being attacked [28]. There is a trend that the widespread use of IoT devices causes an increasing number of DDoS attacks to be launched from infected IoT devices.

2.2. Detection by Traffic Pattern. Identifying botnets by analyzing traffic patterns is an effective method. Communication between botnets is often characterized by command and control channels and often uses specific protocols for communication. Botnet traffic can be effectively identified by analyzing the communication patterns between networks. BotSniffer is a typical approach to identifying bot nodes by analyzing traffic patterns [29]. The model consists of the monitoring engine and the correlation engine. The monitoring engine inspects network traffic and passes suspected traffic records to the relevant engine for analysis. The model identifies botnets in background traffic by assuming that most nodes in the botnet network should react similarly. These botnets should have a similar structure and content of messages or a similar distribution of IP addresses and port ranges for scanning activity. The model clusters traffic messages by analyzing the similarity between messages. Thus, the set of botnet nodes is aggregated into a cluster. In 2016, Bartos et al. proposed a method to detect malicious traffic based on information from network packets as well [30]. This method uses the statistical features computed from network traffic as a training set and identifies malicious behaviour in network traffic by training a classifier. The method has a high precision (reaching 90%) and successfully identifies malware and variant samples not defined in the previous training.

These methods of botnet identification by using detailed information about network traffic packets generally have a high degree of precision and accuracy. However, a 2014 paper by Rndic and Laskov suggests that attackers can evade detection by manipulating information about network traffic packets [31]. In their experiments, the attackers were able to successfully avoid detection through automated inference algorithms and approximation algorithms without any detailed information about the detection system. The accuracy of the classification detector drops dramatically (from almost 100% to 28–33%) for these malicious behaviours that avoid detection by manipulating network traffic packets. Another study has also shown that attackers can also evade traffic monitoring by deliberately manipulating their communication patterns or encrypted channels [29]. And all these methods need to collect the privacy data of nodes into a central server for processing, which may cause the risk of user privacy leakage.

2.3. Detection by DNS Blacklist. Some botnet identification methods need to be supported by prior knowledge. These methods identify botnets by analyzing traffic data access information to define whether this access is associated with a botnet by comparing the domain name that appears against a blacklist of network domains obtained through prior knowledge. These methods avoid malicious behaviour by disabling botnet nodes from communicating with hosts [32]. This type of defence is passive and can effectively prevent botnets from launching malicious attacks. However, the blacklist used by the model is generally public, and there is a possibility that the information is not updated on time.

2.4. Detection by Graph Feature. As graph neural network technology evolves, there are more and more methods to identify botnets by their particular network topology. For botnets with a P2P structure there is generally a high rate of mixing. This is due to the fact that, in order to organise malicious attacks, botnets need to be more efficient than normal networks in terms of disseminating information. In the case of centralized botnets, a hierarchical structure is evident in the topology diagram. Based on this feature, previous research has proposed that P2P botnets can be detected based on their fast mixing rate [10]. Some studies have also shown that botnet detection can also be achieved by analyzing the number and size of connection graph components [11]. However, distinguishing the topology of a botnet from that of a normal network from the vast volume of network traffic is very difficult. Using this detection method first requires manual work to label topological features and narrows the range by performing several prefiltering steps. The method also requires parameters to be adjusted according to the data. Zhou et al. published a method to identify topologies from massive network traffic using GNNs in 2020 [33]. This method achieved good prediction results without prior knowledge and manual processing. However, when the training dataset was small, the model was unable to output good prediction results.

2.5. Federated Learning. Traditional machine learning algorithms require a central server for centralized data collection and processing. Each edge node needs to upload its data to the central server, which poses a risk to protecting the data privacy of edge nodes. On the other hand, due to various privacy laws and regulations, companies do not directly access users' private data. This creates difficulties in how to detect anomalous nodes in the network legally. To address this problem, Konečný et al. proposed a federated learning approach that trains machine learning models without compromising privacy [34]. This method is a distributed machine learning method, where nodes at the edge do not need to upload training data to contribute to the global model training. At the same time, the privacy leakage problem is faced by traditional distributed deep learning algorithms. A collaborative distributed deep learning paradigm was proposed by Liu et al. to address this problem [35]. The method is similar to the federated learning approach proposed by Google, where nodes at the edge also do not need to upload data to a central server to train deep learning models. For federated learning, the technical bottleneck is how to reduce its communication overhead for use in a wide range of devices. Previous algorithms generally reduce the communication overhead of federated learning by designing efficient stochastic gradient descent algorithms [36] and compression models [37].

3. Data Processing and Graph Feature Engineering

3.1. Data Processing. We used the botnet dataset collected by Zhou et al. in 2020 [33] (the dataset can be downloaded at

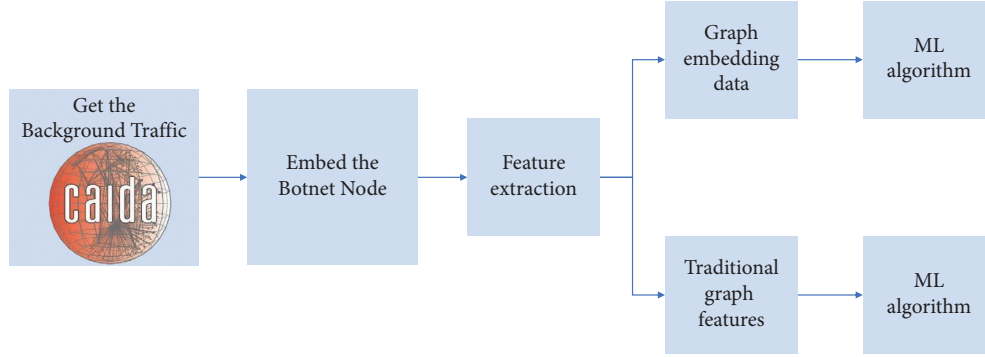


FIGURE 1: Flowchart for deep graph embedding feature botnet node detection experiments.

<https://zenodo.org/record/>). The background traffic for this dataset is all traffic information collected by CAIDA’s monitors in 2018. After aggregating the traffic maps, the subnet traffic was selected for the experiment. The selection of subnet level traffic matches the network traffic topology map pattern available for the enterprise or research organisation. A subset of nodes from the subnet traffic is then randomly selected as botnet nodes. Botnet generation is achieved by embedding the botnet with the P2P structure captured by García et al. into these nodes [38]. The fact that decentralised botnets are not as well defined as centralized botnets with a hierarchical structure makes them more challenging to identify. Therefore, this project focuses on detecting decentralised P2P botnets. We use anomalous traffic detection as a binary classification problem to classify the nodes in a graph. The traffic graph uses IPs as nodes and traffic between IPs as edges. Figure 1 illustrates how we perform the data processing and feature extraction process.

The dataset we used in this experiment has 100 graphs, each containing an average of 140 000 nodes and 700 000 edges. Each of these graphs contains 3000 botnet nodes. The 100 graphs used in this experiment were randomly divided into a training set and a test set in a ratio of 9 : 1. In order to protect user privacy, specific IP information is hidden in this project to achieve data masking.

In this experiment, we process the dataset in two ways. One is to generate graph embedding data by the GNN algorithm introduced in the previous section. The other is to select traditional graph features for training. In the dataset using graph features, each node contains the following features: its own degree, the maximum, minimum, and average values of the degrees of its neighbours.

3.2. Graph Feature Engineering. In order to compare our proposed method of extracting graph features with state-of-the-arts, we have designed two sets of experiments in the evaluation. The first set is our proposed method. The method converts raw network traffic data into graph embedding data by using a trained GNN model. The second set uses traditional graph features as training data. Figure 2 shows the flowchart of the extracted graph feature project.

When training the GNN model, the last layer of the model is the linear layer, which outputs the prediction results for each node. When taking the graph embedding data, we remove the last layer and take the output of its previous layer as the graph embedding features for each node.

In this work, in order to automatically identify the topological features of a botnet under massive background Internet communication graphs, a neural graph networks (GNN) model was designed to accomplish this task [33, 39]. GNN is a neural network model that uses multiple convolutional layers to represent data using vectors. Capturing nodes in this way will contain important information about the node. GNN is ideally suited for identifying the network topology. In each layer of the GNN model, each node updates its state and passes information to its neighbouring nodes. Thus, after multiple layers of messaging, the model will automatically identify the topological relationships of the nodes in the graph.

The GNN model, which we used in this experiment, contains 12 layers. As the number of model layers increases, the nodes will contain more information about neighbouring nodes within hops, leading to better topology identification. Between layers, Zhou et al. used ReLu as the nonlinear activation function [33]. After each layer, we added the bias vector. The model uses the Adam optimiser [40], which uses cross-entropy to calculate the loss, with the learning rate set to 0.005 and the weight decay of the optimiser set to $5e - 4$. The model implementations are based on Pytorch [41] and Pytorch Geometric [42]. The model structure is shown in Figure 3.

For input data, we used $G = \{V, A\}$ to define each graph. We define the set of nodes $\{v_1, \dots, v_n\}$ consisting of n unique nodes as V . And A is an adjacency matrix to show whether there exists a connection between v_i and v_j . When there is a connection between nodes v_i and v_j , we set $a_{ij} = 1$. The diagonal node degrees $\text{Diag}(d_1, \dots, d_n)$ is denoted by D , and the degrees are calculated by the formula $d_i = \sum_{j=1}^n a_{ij}$.

The node feature matrix of the data after layer l is $X^{(l)} \in R^{n \times h}$. The vector $x_i^{(l)}$ in each row is the feature vector of node v_i . h is the size of the node feature vector.

We use the learnable matrix $W^{(l)}$ to update the node vector of each layer:

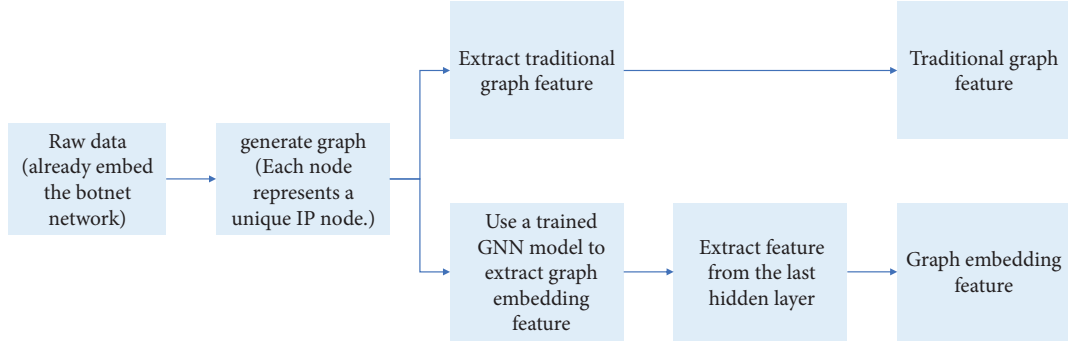


FIGURE 2: Flowchart for extracting graph features.

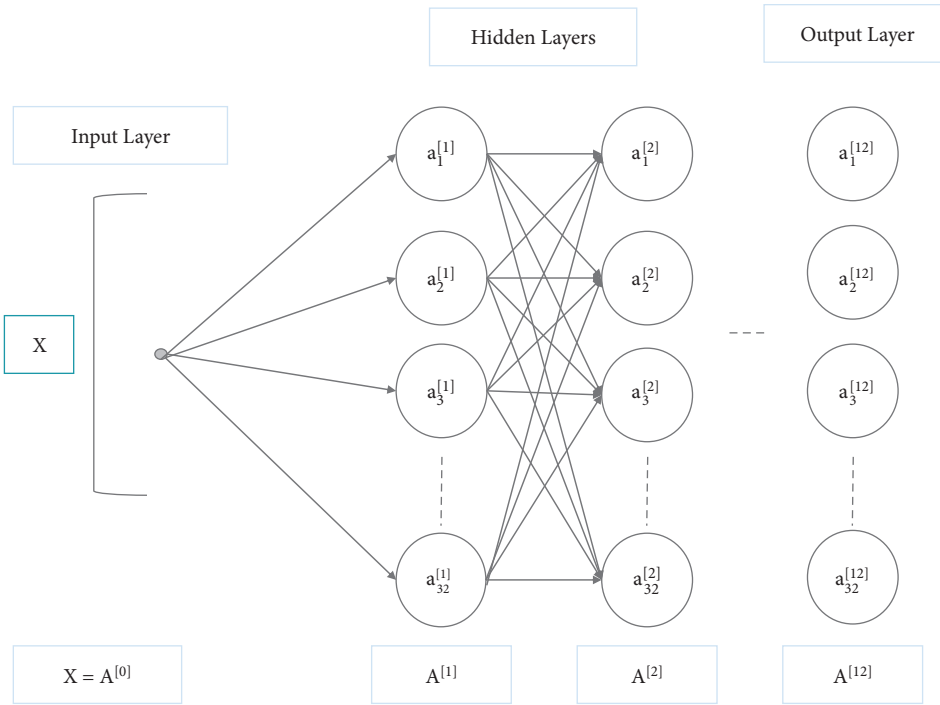


FIGURE 3: A 12-layer GNN model with all nodes of each graph as input layers. After 12 layers of GNN training, each node will be represented by 32 vectors.

$$x_i^{(l)} = x_i^{(l-1)} W^{(l)}. \quad (1)$$

In order to obtain information about neighbouring nodes, the representation of each node is the average of its direct neighbours.

$$x_i^{(l)} = \sum_{j=1}^n \frac{a_{ij}}{\sqrt{d_i d_j}} x_j^{(l)}. \quad (2)$$

In this experiment, we used a GNN model with many layers and normalisation at each layer to avoid numerical instability. It has been proved that GNN's model performance can be significantly improved by using normalisation [39]. The data processing for each layer can be expressed with the following expressions:

$$X^{(l)} = \sigma(\bar{A} X^{(l-1)} W^{(l)}), \quad (3)$$

$$\bar{A} = D^{-1/2} A D^{-1/2}.$$

σ represents the nonlinear activation function ReLU. In our previous description, each node's data is updated according to the data of its neighbouring nodes. So after the L-level transformation, each node's data contains information about its adjacent nodes within its L-hop. This helps us to identify its topology. This is why the multilayer GNN model was chosen for the data transformation. When training the GNN model, we added a linear layer to output the predictions for each node.

$$X^{(l)} = (X^{(l-1)} U^{(l)} \sigma(\bar{A} X^{(l)} W^{(l)})). \quad (4)$$

The learnable transformation matrix at layer l is denoted by $U^{(l)}$. The node data after the L layer will be input into the linear layer, and the softmax algorithm is used for the final classification. By adjusting \bar{A} , it is possible to control how neighbouring node features are normalised prior to aggregation, resulting in different variants of the GNN model. In $\bar{A} = D^{-1/2}AD^{-1/2}$, \bar{A} is calculated from the degree of the source node and the degree of the target node. The graph attention network is also a GNN model variant that calculates \bar{A} using a learnable nonlinear function based on node features, with each edge being independently normalised.

In this project, we modified the calculation of \bar{A} according to the method proposed by Zhou et al. [33] to identify the topology more efficiently. Botnets with a P2P structure have a high rate of rapid mixing. In order to better identify the fast mixing rate [10] of the botnet, a random walk was used for normalisation. In our model, \bar{A} is calculated as $\bar{A} = D^{-1}$. Unlike the previous equation, the calculation of \bar{A} is only related to the degree of the source node and does not take into account the degree of the target node, thus achieving a random walk.

This experiment evaluates the botnet detection performance on six machine learning algorithms: logic regression, random forest, decision tree, Naive Bayes, k-nearest neighbour, and random forest. These six machine learning algorithms are chosen because they converge faster during training and are more suitable for anomaly detection for a large number of nodes in order to compare the difference between our graph embedding based features and the traditional graph features. Six machine learning algorithms with the same parameters are used in the experiments to training on each of the two datasets.

4. Experimental Evaluation

4.1. Evaluation Metrics. The dataset we used in this project was very unbalanced, with only 2% of the nodes being botnet nodes. It is not reasonable to evaluate model performance by accuracy in this case. This project focuses on whether the detection model can effectively find botnet nodes. In this experiment, we import some information retrieval metrics to evaluate the performance. 1) Positive and negative: Suppose there is a node N . The classifier's output is whether N belongs to a botnet or not. Researchers commonly use the true positives (TP), false positives (FP), and false negatives (FN) to evaluate the classifier. These metrics are defined below:

- (a) True positive: The botnet node is correctly classified as botnet class.
- (b) False positive: The normal node is incorrectly classified as botnet class.
- (c) True negative: The normal node is correctly classified as normal class.
- (d) False negative: the botnet node is incorrectly classified as normal class.

The relationship between these criteria and the classification results is shown in Table 1.

TABLE 1: Evaluation metrics.

	Predicted negative	Predicted positive
True negative	TN	FP
True positive	FN	TP

In order to test the recognition performance, we introduced true positive rate (TPR) and false positive rate (FPR) as evaluation metrics in this experiment.

TRP represents the ratio of the number of nodes correctly classified as botnet nodes to the number of all botnet nodes and is calculated as follows:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (5)$$

FPR is defined as the ratio of the number of nodes incorrectly classified as botnet nodes to the number of all normal nodes.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \quad (6)$$

In order to provide a more comprehensive analysis of the performance of each algorithm, we have introduced the metrics, precision, recall, and F -measure.

- (a) Precision represents the ratio of the number of nodes correctly classified as botnet nodes to the number of all nodes predicted to be botnet nodes.

$$\text{Precision} = \frac{\text{TP}}{\text{FP} + \text{TP}}. \quad (7)$$

- (b) Recall is also known as detection ratio. It is obtained by calculating the ratio of the total number of nodes correctly classified as botnet nodes to the total number of botnet nodes.

$$\text{Recall} = \frac{\text{TP}}{\text{FN} + \text{TP}}. \quad (8)$$

- (c) The F -measure is a comprehensive evaluation of the algorithm's precision and recall criteria. The values of both detection rate and precision affect the calculation of the F -measure, making it easy for us to make comparisons. It is calculated by the following formula:

$$F - \text{measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (9)$$

4.2. Comparisons among Different ML Algorithms. In this section, we evaluate the performance of each algorithm when using graph embedding data for training. In this set of experiments, each algorithm uses 90 graphs from the dataset for training and 10 graphs for validation of the detection. The dataset used in these experiments was extremely unbalanced. In order to obtain more reasonable detection results, we adjust the weights of the bot nodes and normal nodes by setting the parameter "class_weight" to be balanced, and we sampled the botnet node data.

Table 2 shows the results when the graph embedding data is used for training. From Table 2, we can see that the three algorithms logical regression, Naive Bayes, and KNN are not very effective in detecting botnets. The botnet detection rate of most algorithms is less than 50%. Tree-based algorithms such as XGBoost and decision tree are susceptible to bot nodes and achieve a bot node detection rate of over 98%. In terms of comprehensive performance evaluation, random forest has a relatively low false positive rate, a high precision rate, and the second highest F -measure value while retaining a high bot node detection rate. This suggests that tree-structure-based machine learning algorithms are superior to other algorithms in identifying topologies. On the other hand, tree-structure-based algorithms can find the vast majority of bot nodes, but their precision and F -measure values are generally low.

We use the graph embedding data to train the algorithm in this part of the experiment. We used 32 features to represent each node. However, these features are not independent of each other. Thirty-two features work together to generate a vector of nodes. For logistic regression, support vector machines, and naive Bayesian algorithms, these algorithms do not capture the relationship between the features. This makes these algorithms perform poorly when trained with graph-embedded data. For algorithms based on tree structures such as XGBoost, the prediction process requires multiple features to work together to make a prediction. This makes tree-structure-based machine learning algorithms more advantageous when using graph-embedded data for prediction. Also, we can find that the KNN algorithm has better performance in botnet node detection. This is since the KNN algorithm also works with multiple features to predict the results.

4.3. Comparisons with Traditional Graph Feature Set. This section uses the same dataset for training, but the extracted features are traditional graph features. The features include the degree of the node and the maximum, minimum, and mean of its neighbourhood degrees. We have selected the same raw data for training as in the previous section to facilitate our comparison of the performance changes of the algorithm. Table 3 shows the performance of the different algorithms when using traditional features.

From Table 3, we can see that all six machine learning algorithms show a sharp drop in performance when trained with traditional features compared to using graph-embedded features. Both the random forest algorithm and the naive Bayesian algorithm have a detection rate of less than 5%. The XGBoost algorithm, which has a high botnet node detection rate, decision tree, and logistic regression algorithms all have a false positive rate of over 90%. Such performance is unacceptable for botnet node detection. Figure 4 shows a comparison of the results of the different algorithms after training with different features.

These results show that it is not enough to use only two hops of the neighbour node for identifying botnet nodes. An algorithm using only this information cannot identify the topological features of the network. To recognise topological

TABLE 2: Evaluation result.

Algorithm	Metric			
	Recall	FPR	Precision	F -measure
XGBoost	98.5	37.6	5.4	10.2
Decision tree	99.1	38.5	5.3	10.1
Random forest	73.8	0.6	72.0	72.9
Logistic regression	44.5	27.3	3.4	6.3
KNN	64.0	0.1	96.1	76.8
Naive Bayes	0.6	0.7	2.2	0.9

TABLE 3: Tradition feature result.

Algorithm	Metric			
	Recall	FPR	Precision	F -measure
XGBoost	99.1	98.9	2.2	4.3
Decision tree	94.6	94.7	2.2	4.3
Random forest	1.9	1.9	2.1	1.9
Logical regression	91.4	91.3	2.2	4.3
KNN	0	0	4.5	0
Naive Bayes	0	0	0	0

features, we need to obtain the hidden representations of the nodes for more sophisticated learning. The advantage of using GNNs is that the features of each node are generated by the joint influence of the features of its surrounding nodes. And with the multilayer GNN model, the features of each node will be related to the features of the nodes within its multihop. The GNN can effectively help us obtain the topology's implicit features, which allows training with graph-embedded features to outperform the performance by using only traditional features.

4.4. Impact of Imbalanced Data. This section evaluates the impact of the bot to nonbot node ratio on the machine learning algorithms described above. In previous experiments, the datasets we used for training and testing were extremely unbalanced. In this section, the training dataset retains all the botnet nodes and the same number of normal nodes when training the classifier. The training set contains 270 000 botnet nodes and 270 000 normal nodes. The same test dataset used in Section 4.2 was used when testing the classifier's performance.

From Table 4, we can find that, except for the logical regression algorithm, all the algorithms achieved a recall rate of over 90%. Compared with Table 2, the detection rates of random forest, KNN, and naive Bayesian algorithms are considerably higher. In particular, the naive Bayesian algorithm was enhanced from the original 0.6% to 99%. However, the high detection rate of the naive Bayesian algorithm comes at the cost of the false positive rate. It has a false positive rate of 99%. At the same time, we can observe that the random forest and KNN algorithms, which have a significant increase in detection rate, have a greater reduction in precision and an increased false positive rate. The false positive rate for both has increased from less than 1% originally to around 12%. The precision of the random forest

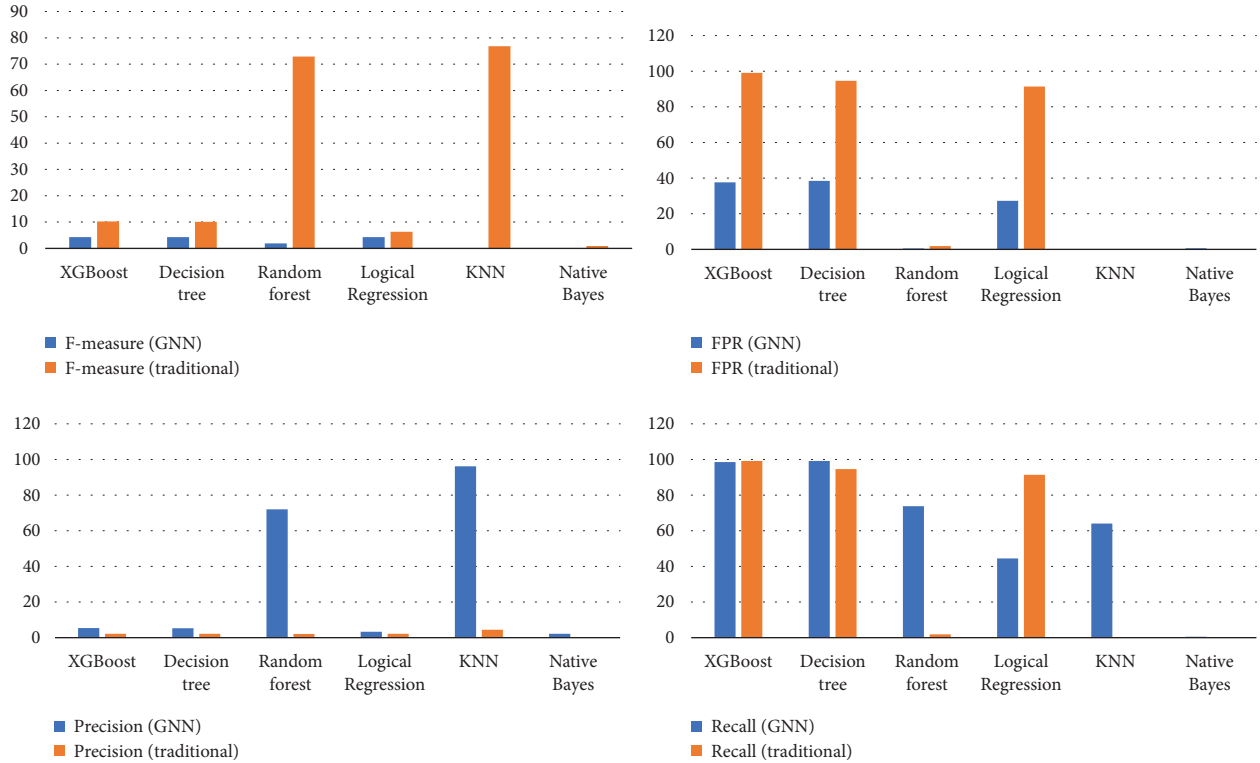


FIGURE 4: Comparison of the results of training with different features.

TABLE 4: Traditional features (balanced dataset) results.

Algorithm	Metric			
	Recall	FPR	Precision	F -measure
XGBoost	98.3	35.0	5.8	11.0
Decision tree	99.3	38.9	5.3	10.1
Random forest	92.5	11.5	15.1	26.0
Logical regression	25.9	13.8	4	6.9
KNN	93.2	12.2	14.4	25.0
Naive Bayes	99.0	99.0	2.2	4.2

algorithm was reduced from 72% to 15.1%. Furthermore, the precision of the KNN algorithm was reduced from 96.1% to 14.4%. There was no significant change in performance for both the XGBoost algorithm and the decision tree algorithm, with the four criteria performing in line with Table 2. The combined performance of the six machine learning algorithms does not improve significantly when trained with a balanced dataset.

5. Discussion

In this work, we focus on how to detect malicious behaviour, but both prevention and detection are equally crucial for stopping malicious behaviour. Much work is currently being done to enable nodes to perform specified actions to prove to other nodes that they are legitimate. For the current growing number of IoT devices, the limited resources on them make them vulnerable to some specific cyberattacks. Authentication and Key Agreement (AKA) needs to be established to

protect communication between IoT devices and remote servers. The main issue being researched is how to balance security and usability in designing AKA protocols. The AKA protocol designed by Qiu et al. is based on chaotic map and uses the “Fuzzy-Versifiers” and “Honey” techniques [43]. The approach proposes a secure three-factor AKA protocol for lightweight mobile devices based on an extended chaotic graph. However, whether the approach can be deployed on IoT devices with more resources than mobile devices has not been validated. On the other hand, Wang et al. argue that cloud centres are necessary under the current conditions of limited node computing resources [44]. With the cloud centre, it can help make intelligent decisions and ease the pressure on nodes for computing and storage. This approach effectively reduces the computational pressure on the sensor nodes, making their computational cost only equivalent to that of the symmetric encryption algorithm. For our future research, we also plan to combine both detection and prevention of malicious behaviour in order to prevent it more efficiently.

6. Conclusion

In this paper, we propose a way for extracting botnet graph embedding features and comprehensively analyze the performance of different machine learning algorithms. To perform this evaluation, we embedded network traffic generated by 300,000 bot nodes captured from the actual network into background traffic containing 14,000,000 nodes to create a network graph. In this paper, the GNN model is applied to capture the topology of a botnet with a

P2P structure. In order to investigate the botnet detection capabilities of different classifiers, experiments were conducted using different sampling methods and feature extraction methods. A few insights were generated from our evaluations. First, the tree-structure-based machine learning algorithms outperform other algorithms. Second, machine learning algorithms cannot correctly distinguish a botnet from a normal network if the node data contains fewer hops of information about its neighbours. Third, the detection rate of botnet nodes can be improved by adjusting the data balance, but it will increase the false positive rate of the algorithm. We hope these insights can help researchers or cybersecurity professionals better detect botnet traffic.

Data Availability

The data supporting this paper were taken from previously reported studies and datasets, which have been cited. The processed data are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no potential conflicts of interest.

Acknowledgments

The research and publication of this article were funded by the National Natural Science Foundation of China (Grant no. 62062069).

References

- [1] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies*, pp. 268–273, Athens, Greece, June 2009.
- [2] N. Ianelli and A. Hackworth, "Botnets as a vehicle for online crime," *CERT Coordination Center*, vol. 1, no. 1, 2005.
- [3] B. Paul, T. Holz, M. Kotter, and G. Wicherski, "Know your enemy: tracking botnets," *The HoneyNet Project & Research Alliance*, vol. 2005, 2005.
- [4] C. Cimpanu, "There's a 120,000-Strong IoT DDoS botnet lurking around," *Softpedia*, 2016, <https://news.softpedia.com/news/there-s-a-120-000-strong-iot-ddos-botnet-lurking-around-507773.shtml>.
- [5] S. Christian and H. Judith, "Mapping the Internet: a hacker's secret Internet Census," *Spiegel*, 2013, <https://www.spiegel.de/international/world/hacker-measures-the-internet-illegally-wit-h-carna-botnet-a-890413.html>.
- [6] M. Antonakakis, T. April, M. Bailey et al., "Understanding the mirai botnet," in *Proceedings of the 26th {USENIX} security symposium*, pp. 1093–1110, Canada, August 2017.
- [7] B. Krebs, "Krebssecurity Hit with Record Ddos", Website, 2016, <https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/>.
- [8] S. Heron, "Botnet command and control techniques," *Network Security*, vol. 2007, no. 4, pp. 13–16, 2007, [Online]. Available:..
- [9] B. Julian, V. Sharma, C. Nunnery, K. Brent ByungHoon, and D. Dagon, "Peer-to-peer botnets: overview and case study," in *Proceedings of the First Workshop on Hot Topics in Understanding Botnets, HotBots'07*, N. Provos, Ed., Cambridge, MA, USA, April 2007.
- [10] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "Botgrep: finding p2p bots with structured graph analysis," in *Proceedings of the 19th USENIX Conference on Security, USENIX Security*, Washington, DC, February 2021.
- [11] M. P. Collins and M. K. Reiter, "Hit-list worm detection and bot identification in large networks using protocol graphs," in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, pp. 276–295, Springer, Gold Coast, Australia, September 2007.
- [12] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, G. Varghese, and H. C. Kim, *Graption: Automated Detection of P2p Applications Using Traffic Dispersion Graphs (Tdgs)*, University of California, Riverside Report, UCR-CS-2008096080, California, USA, 2008.
- [13] J. Qiu, J. Zhang, W. Luo, and Y. PanNepalXiang, "A survey of android malware detection with deep neural models," *ACM Computing Surveys*, vol. 53, no. 6, pp. 1–36, 2020.
- [14] X. Chen, C. Li, D. Wen, Y. ZhangNepalXiangRen, and K. Ren, "Android hiv: a study of repackaging malware for evading machine-learning detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 987–1001, 2020.
- [15] L. Liu, O. De Vel, Q.-L. Han, J. Xiang, and Y. Xiang, "Detecting and preventing cyber insider threats: a survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1397–1417, 2018.
- [16] Y. Miao, C. Chen, L. Han, J. ZhangXiang, and Y. Xiang, "Machine learning-based cyber attacks targeting on controlled information," *ACM Computing Surveys*, vol. 54, no. 7, pp. 1–36, 2022.
- [17] J. Zhang, L. Pan, Q.-L. Han, C. Chen, S. Wen, and Y. Xiang, "Deep learning based attack detection for cps security: a survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, 2021.
- [18] N. Sun, J. Zhang, P. Rimba Rimba, and Y. GaoZhangXiang, "Data-driven cybersecurity incident prediction: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1744–1772, 2019.
- [19] G. Lin, S. Wen, Q.-L. Han, J. Xiang, and Y. Xiang, "Software vulnerability detection using deep neural networks: a survey," *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1825–1848, 2020.
- [20] S. Liu, M. Dibaei, Y. Chen, J. ZhangXiang, and Y. Xiang, "Cyber vulnerability intelligence for internet of things binary," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2154–2163, 2020.
- [21] M. Wang, T. Zhu, T. Zhang, S. YuZhou, and W. Zhou, "Security and privacy in 6g networks: new areas and new challenges," *Digital Communications and Networks*, vol. 6, no. 3, pp. 281–291, 2020.
- [22] E. Jonsson and E. Jonsson, "Anomaly-based intrusion detection: privacy concerns and other problems," *Computer Networks*, vol. 34, no. 4, pp. 623–640, 2000.
- [23] An Wang, W. Chang, S. Mohaisen, and A. Mohaisen, "Delving into internet ddos attacks by botnets: characterization and analysis," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2843–2855, 2018.
- [24] B. L. Risteska Stojkoska, "A review of internet of things for smart home: challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

- [25] C. Macdonald, *Passwords Used in the Biggest Ever Cyberattack Revealed - and '12345' and 'password' Were Top*, Dailymail, 2016, <https://www.dailymail.co.uk/sciencetech/article-3825740/Passwords-used-biggest-DDoS-attack-revealed-12345-password-top.html>.
- [26] T. Easton, “Chalubo Botnet Wants to Ddos from Your Server or Iot Device”, Sophos, 2018, <https://news.sophos.com/en-us/2018/10/22/chalubo-botnet-wants-to-ddos-from-your-server-or-iot-device/>.
- [27] D. Wang, H. Cheng, P. Wang, X. Jian, and G. Jian, “Zipf’s law in passwords,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.
- [28] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, “Targeted online password guessing: an underestimated threat,” in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 1242–1254, Vienna, Austria, July 2016.
- [29] G. Gu, J. Zhang, and W. Lee, *Botsniffer: Detecting botnet command and control channels in network traffic*, in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, California, USA, January 2008.
- [30] K. Bartos, M. Sofka, and V. Franc, “Optimized invariant representation of network traffic for detecting unseen malware variants,” in *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, Austin, TX, August 2016, <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/bartos>.
- [31] N. Rndic and P. Laskov, “Practical evasion of a learning-based classifier: a case study,” in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, Berkeley, CA, USA, May 2014.
- [32] R. Perdisci and W. Lee, “Method and system for detecting malicious and/or botnet-related domain names,” *US Patent*, vol. 10, p. 688, 2018.
- [33] J. Zhou, Z. Xu, A. M. Rush, and M. Yu, “Automating botnet detection with graph neural networks,” *AutoML for Networking and Systems Workshop of MLSys 2020 Conference*, vol. 1, 2020.
- [34] J. Konečný, H. B. McMahan, and X. Felix, P. Richtarik, A. T. Suresh, and D. Bacon, “Federated learning: strategies for improving communication efficiency,” 2016, <https://arxiv.org/abs/1610.05492>.
- [35] Y. Liu, J. J. Q. Yu, J. Kang, and S. Zhang, “Privacy-preserving traffic flow prediction: a federated learning approach,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 2020.
- [36] N. Agarwal, A. T. Suresh, F. Yu, S. Kumar, and H. Brendan McMahan, “cpsgd: communication-efficient and differentially-private distributed sgd,” 2018, <https://arxiv.org/abs/1805.10559>.
- [37] Yi Liu, S. Garg, J. Zhang, J. XiongKangHossain, and M. S. Hossain, “Deep anomaly detection for time-series data in industrial iot: a communication-efficient on-device federated learning approach,” *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2021.
- [38] S. García, M. Grill, J. Zunino, and A. Zunino, “An empirical comparison of botnet detection methods,” *Computers & Security*, vol. 45, pp. 100–123, 2014, [Online]. Available: .
- [39] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, <https://arxiv.org/abs/1609.02907>.
- [40] D. P. Kingma and Ba Jimmy, “Adam: a method for stochastic optimization,” 2014, <https://arxiv.org/abs/1412.6980>.
- [41] P. Adam, S. Gross, F. Massa, and A. Lerer, J. Bradbury, G. Chanan, T. Killeen et al., Pytorch: an imperative style, high-performance deep learning library,” *Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.
- [42] M. Fey and J. Eric Lenssen, “Fast graph representation learning with pytorch geometric,” 2019, <https://arxiv.org/abs/1903.02428>.
- [43] S. Qiu, D. Wang, G. Kumari, and S. Kumari, “Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, 2020.
- [44] C. Wang, D. Wang, and D. He, “Efficient privacy-preserving user authentication scheme with forward secrecy for industry 4.0,” *Science China Information Sciences*, vol. 65, no. 1, Article ID 112301, 2022.

Research Article

VNGuarder: An Internal Threat Detection Approach for Virtual Network in Cloud Computing Environment

Li Lin ^{1,2}, Huanzeng Yang,¹ Jing Zhan ^{1,2} and Xuhui Lv¹

¹College of Computer Science, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China

²Beijing Key Laboratory of Trusted Computing, Beijing 100124, China

Correspondence should be addressed to Li Lin; linli_2009@bjut.edu.cn and Jing Zhan; zhanjing@bjut.edu.cn

Received 25 October 2021; Revised 20 December 2021; Accepted 15 March 2022; Published 16 April 2022

Academic Editor: Chunhua Su

Copyright © 2022 Li Lin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Edge-assisted Internet of things applications often need to use cloud virtual network services to transmit data. However, the internal threats such as illegal management and configuration to cloud platform intentionally or unintentionally will lead to virtual network security problems such as malicious changes of user network and hijacked data flow. It will eventually affect edge-assisted Internet of things applications. We propose a virtual network internal threat detection method called VNGuarder in a cloud computing environment, which can effectively monitor whether the virtual network configuration of legitimate users under the IaaS cloud platform has been maliciously changed or destroyed by insiders. First, based on the life cycle of cloud virtual network services, we summarized two types of internal attacks involving illegal use of virtualization management tools and illegal invocation of virtual network-related processes. Second, based on normal behavior of tenants, a hierarchical trusted call correlation scheme is proposed to provide a basis for discovering that insiders illegally call virtualized management tools and virtual network-related processes on the controller node of the cloud platform or the network node and compute node. Third, a traceable mechanism combining real-time monitoring and log analysis is introduced. By collecting and recording the complete call process of virtual network management and configuration in the cloud platform, and comparing it with the result of the hierarchical trusted call correlation, abnormal operations can be reported to the tenants in time. Comprehensive simulation experiments on the Openstack platform show that VNGuarder can effectively detect illegal management and configuration of virtual networks by insiders without significantly affecting the creation time of tenant networks and the utilization of CPU and memory.

1. Introduction

In the era of Internet of Things (IoT), more and more devices (e.g., sensors, terminals, household appliances, thermostats, televisions, automobiles, and production machinery) are connected to the Internet, and various edge-assisted IoT applications have emerged in industrial manufacture [1,2], medical healthcare [3,4], smart electric power grid [5], and privacy-aware federated learning system [6]. Due to the limited computing and storage capabilities of edge IoT devices, these applications often require virtual network services of cloud platforms to transmit data to be analysed or stored [7]. Network virtualization technology of cloud computing can achieve efficient reuse of network resources and centralized distribution of network traffic. In a cloud system, there are usually three

kinds of networks [8] as follows. *External network* is used to realize the function of tenants' virtual machines accessing the Internet. *Management network* is used to provide internal communication between various components of cloud platform. *Data network* is a virtual network formed by data communication between virtual machines in cloud services, where *management network* is responsible for configuring and managing nodes in virtual network. Since management network and data network are completely controlled by cloud service providers, there are virtual network security problems such as malicious internal cloud managers using their own authority to directly change tenants' network configuration [9], intercept data transmitted in a virtual network [10], illegally read or write virtual network devices [9]. These internal threats will eventually affect the edge-assisted IoT applications.

Cloud security has always been a research hotspot in the field of network security [11]. The existing virtual network security protection researches mainly focus on the prevention of external threats of virtual network [12–14] and the solution of internal threats such as tampering of virtual machine image and storage faced by cloud compute nodes [15–17]. For the prevention of external threats to virtual network, the current work mainly focuses on data transmission security protection in virtual network. For example, Baik et al. [18] proposed a traffic monitoring method based on SDN architecture to find the illegal forwarding of tenant traffic and prevent the malicious tampering, reading and writing to virtual devices by attackers. Zhang et al. [19] proposed to use VPN technology to ensure data transmission security in virtual network and prevent malicious interception by external attackers. However, these researches ignore the virtual network internal threats caused by cloud platform managers. For the internal threats of cloud compute nodes, the current work mainly focuses on security protection such as virtual machine image and storage. For example, Miao and others [20] proposed to see all client-related operations made by KVM by adding a secure nested virtualization layer. In our early work [21], we proposed a virtualization protection framework supporting tracing to protect the important data of IaaS users stored on compute nodes from illegally accessing or maliciously damaging by malicious insiders. However, there have been different functions between network nodes and compute nodes in a cloud platform. Network node is responsible for controlling the virtual network of communication and virtual equipment configuration. The administrators and operators of cloud virtual network services with high authority can directly change a tenant’s virtual network configuration, such as the tenant’s three layer forwarding address, which may make the tenant’s data forwarded to one attacker but the tenant do not know it.

There are three problems that should be solved in order to find the internal threats within virtual network in the cloud computing environment. Firstly, there are many virtual network devices such as *taps* and *tuns* in the cloud platform. Malicious insiders may copy user data by overwriting the information of these virtual devices. Therefore, it is an important prerequisite for judging whether there is a threat by process monitoring of virtual network devices. Secondly, under the cloud virtual network service, tenants can configure and manage virtual networks using virtualization management tools. Because malicious insiders with higher privileges can call these virtualization management processes directly and then change the tenant’s network via the hypervisor, these processes should be monitored. Thirdly, tenants often manage and configure their virtual networks by using visual API calls from cloud services. In this case, malicious insiders can also manage the network by calling these APIs to tamper with or impersonate tenants, so the API interfaces need to be monitored. In particular, it is not easy to distinguish whether the management of a virtual network by an insider comes from a legitimate tenant’s request or from insiders’ own malicious behavior.

To address the above issues, this paper proposes a virtual network internal threat detection approach in cloud computing environment called VNGuarder. In VNGuarder, based on in-depth analysis of network module source codes of cloud service, the API or function call relationships at different levels of virtual network service process of different nodes in cloud platform can be discovered. Exploiting finite state machine algorithm, a hierarchical trusted invocation association scheme based on tenants’ normal behavior is proposed to provide a basis for discovering illegal invocation of insiders to virtualization management tools and virtual network-related processes on controller node or network/compute nodes of cloud platform. Meanwhile, a trace-enable mechanism to find the actual invocation behavior of virtual network from tenants, where some monitoring points are deployed in relevant processes and services of important nodes such as controller node, network node and compute node, and all invocation information collected by monitoring points in real-time are recorded in logs. Finally, a behavior matching algorithm is introduced to find malicious internal threat by comparing the actual trace with the results of hierarchical trusted invocation association.

Compared with the existing work, the main contributions of this paper are as follows.

- (1) Compared with traditional data security transmission methods such as SDN-based traffic monitoring and VPN technology, this paper aims to solve virtual network security problems caused by malicious privileged insiders and has found two kinds of malicious management behaviors: illegal use of virtualization management tools and illegal use of virtual networks.
- (2) Different from the existing single node monitoring schemes, this paper introduces a trace-enable mechanism that integrates real-time monitoring and log analysis, and comparing it with the hierarchical trusted invocation in tenants’ normal behavior, which can improve the detection effect of insider threats.
- (3) The authors have successfully implemented VNGuarder in Openstack platform and conducted several comprehensive experiments to evaluate the effectiveness and performance of VNGuarder. Experimental results show that VNGuarder can detect not only internal threats such as traditional single-process illegal calls, but also important internal threats such as modification of virtual network device configuration through virtualization tools, with only a small performance degradation.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 outlines problem statement, including threat model considered in this paper. Section 4 introduces VNGuarder in detail. Section 5 presents our experimental evaluation results. In Section 6 we provide a discussion on the comparison VNGuarder with our earlier work and other virtual network detection methods. In

Section 7, we conclude the paper and present some possible future work.

2. Related Work

Virtual network security under cloud computing environment has always been a research hotspot in the field of cloud security. Researchers have proposed some detection and protection schemes, including traffic monitoring based on SDN, monitoring at kernel. For example, Hussein et al. [22] proposed to use SDN technology to automate the collection of network traffic, network behavior, security events, and other information, so as to find out the traffic hijacking behavior of malicious insiders. Shao et al. [23] proposed a security model based on SDN technology. By using the SDN control layer, the virtual network and virtual network equipment were divided into independent security management domains, and corresponding security event information was collected and monitored. SDN is a kind of network virtualization technology. The separation of forwarding and control layer is helpful to find threats such as malicious change of virtual equipment and traffic theft under the virtual network, but the controller itself will still face internal threats of illegal management intentionally or unintentionally within the control scope. Carvalho et al. [24] proposed to use Nagios cloud computing platform monitoring module and complete monitoring of monitoring points in the form of plug-ins. Based on monitor of kernel, although it can effectively judge whether the related process modules are abnormal, real-time monitoring will consume a large amount of CPU resources, and such a single abnormal behavior is more likely to cause misjudgment and affect the experience of cloud services.

Insider threats are one of the most serious security challenges facing cloud computing [25]. Internal threats mainly come from cloud service providers, including the intentional or unintentional management and configuration of tenant data and tenant network by cloud service insiders. In terms of internal threat detection and prevention of tenant data, Johannes and others. [26] found that malicious cloud service insiders could steal user data by reading memory and file system information. Some researchers have come up with solutions to these security threats, for example, based on trusted computing, Yu et al. [27] combined trusted computing and cloud security by establishing TPM (short for Trusted Platform Module). Implementing a complete set of trusted systems for detecting and verifying VM identity information, Hussein et al. [28] proposed a framework to review and monitor the hard disk, CPU, and user data in the virtual machine image and protect the security of the virtual machine image in the cloud environment through an expert review method. Kansal et al. [29] proposed an early detection and isolation method called EDIP to mitigate insider attack behaviors. EDIP detects all legitimate clients in the system at the agent level and isolates innocent clients by migrating them to the attack agent. Singh et al. [30] proposed a general behavior-based insider threat detection method, where the behavior is characterized by user activity (such as logon-logoff, device connect-disconnect, file-access, http-url-

requests, e-mail activity). But the above research focuses on solving the internal threats to data storage in the cloud environment, ignoring that malicious insiders can intercept tenant data by monitoring and modifying the tenant's virtual network.

To sum up, the existing virtual network security protection in cloud computing environment mainly focuses on external attackers and ignores insider threats. Therefore, this paper proposes an insider threat detection approach to deal with the security risk in management and configuration of virtual network rather than data transmission.

3. Problem Description

This section introduces application scenarios and threat model considered in this paper.

3.1. Application Scenarios. Figure 1 depicts a common scenario for IaaS cloud service. In IaaS deployment, a cloud user, which has been authenticated and authorized by CSPs, can manage or configure his virtual network by logging in his management interface on cloud platform (such as Openstack). When a user or compute service components' (such as nova) request is sent to virtual network control components (such as neutron-server) in controller node, the virtual network control component will invoke virtualization service process (e.g., agent process) by calling virtualization management tools and the corresponding remote procedure call API. Then a virtual network can be built up by all kinds of virtual network devices (e.g., Bridges, Virtual Switches) and hypervisor, where tenant can manage and configure the virtual network.

The normal startup process of cloud virtual network service by users is shown in Figure 2. After obtaining authorization, a user can manage his private network through cloud platform management interface, such as to create, delete, change, or configure the network services (e.g Firewall, Load Balancing, NAT) or network's core resources (such as Network, Subnet and Port). The user service interface will be invoked when a user manages his virtual network in the console. When the controller node receives the user requests, network service interface such as the Core of neutron-server API or the Extension API in controller node will determine whether to need to invoke a remote call interface in line with the request type. If necessary, the request is forwarded to the compute node of the cloud service or the management implementation interface of the network node through the remote call interface. According to different request services, the management implementation interface invokes the corresponding virtualization process (such as agent process) and finally completes the configuration of virtual network through the virtualization process.

3.2. Attack Model. Since users' private network is exposed to cloud service nodes in current cloud environment, insiders can not only pose security threats to virtual network through direct access but also threaten users' network security by directly calling cloud service-related API. On the

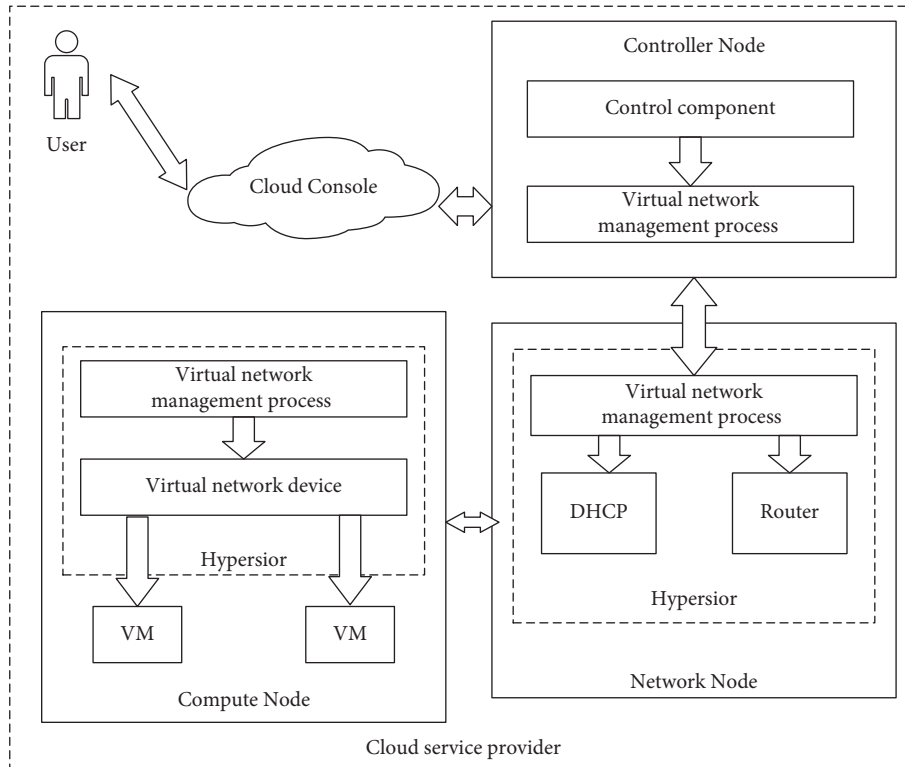


FIGURE 1: Network service in IaaS deployment.

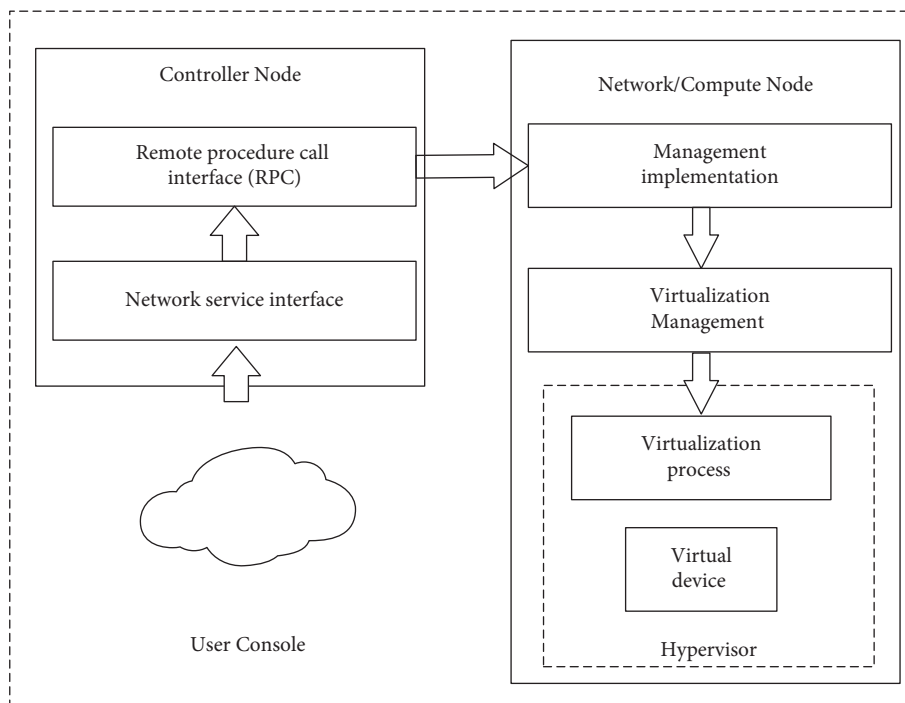


FIGURE 2: Normal startup process in IaaS network service.

mainstream platform, such as Openstack, a malicious insider can invoke different APIs at different nodes. For instance, a malicious insider can invoke the virtual network service management interface named neutral-server at controller node to manipulate the user’s private network, or call libvirt

virtualization management tool with virsh module to change the current user’s virtual network, which cause legitimate users to not be able to access virtual network cloud service. For another example, a malicious insider can call the virtualization process qemu-kvm (short for quick emulator and

kernel-based virtual machine) in a compute node or a network node to enter the virtual device, illegally rewrite the virtual device, and forward the copy traffic to the designated virtual host in the virtual network, causing user data security issues. If you only monitor a single API interface or process level, there is no way to tell whether these actions are normal user behavior or are illegal malicious changes from malicious insiders.

There are two main concerns from the perspective of cloud tenants. One is whether his private network will be infiltrated and destroyed by malicious insiders. The other is whether the communication in the cloud network is safe. Since a users' management and configuration to his virtual network is directly exposed to cloud service provider, there must be security threats from insiders or untrusted cloud service providers. As shown in Figure 3 there are two kinds of insider threats in virtual network.

3.2.1. Insider Manages and Configures a User's Network by Making Unauthorized Calls to Virtualization Management Tools or Processes Related to the Virtual Network. Cloud platform administrators can use their administrator privileges to achieve the purpose of changing user network configuration and state by making unauthorized calls to virtualization management tools and related virtual network processes. For example, for qemu virtualization process, the command "brctl add br br0, brctl add if br0 eth0" can be used to create a bridge and add eth0 (represents the first Ethernet card) to the bridge. For virtualization management tool libvirt, Internal operators can use commands "virsh net-undefine VBR", "virsh net-destroy VBR", "virsh net-edit VBR" to delete, stop, and modify the specified virtual network. Apparently, unauthorized calls by these virtualization management tools or virtualization processes associated with the virtual network will pose a security threat to the virtual network.

3.2.2. Insider Attack a Users' Virtual Network by Changing Its Virtual Devices Configuration. The malicious cloud manager can directly configure the user's virtual devices (such as OVS) inside the cloud platform, including changes to the OVS port and network. When a user (e.g. edge devices in IoT) normally uses the cloud platform to store or transfer data, a malicious insider indirectly changes the storage address of the user's data or directly obtains the data by changing the configuration of the user's virtual device.

To detect the above two threats, a method called VNGuarder is proposed in this paper.

4. Design of VNGuarder

In this section, we will give the design of the proposed VNGuarder method, including its working principles and detailed functions.

4.1. Working Principles. For the above application scenario, the basic idea of VNGuarder is to build a hierarchical trusted invocation association by analysing the network module source

code of cloud service. Based on the hierarchical trusted invocation association analysis, various user's trusted behaviors are constructed. Conduct and deploy information acquisition and monitoring modules at various node levels such as network service interface, remote call interface, management implementation interface, virtualization management interface, and virtualization process shown in Figure 2. Meanwhile, the actual management implementation process of cloud virtual network service is monitored in real time. All the calls of API and process of nodes at different levels are recorded. By matching the actual cloud service behavior acquired from acquisition and monitoring modules with trusted behavior of normal user, the illegal management and configuration of users' virtual networks by malicious insiders can be found. The overall workflow is shown in Figure 4.

4.2. Construction of Trusted Behavior

4.2.1. Description of Tenant Trusted Behavior Based on Finite State Machine. In VNGuarder, the user's interaction with his virtual network cloud service is analysed based on all kinds of interfaces, such as virtual network service interface, remote call interface, management implement interface, virtualization management interface, and so on. Mining multilayer APIs association in source codes, the trusted invocation behavior of each legitimate user using the virtual network service under cloud platform is obtained. The keywords of each node are created by the proposed hierarchical trusted invocation association method.

Different types of logical nodes are established according to the invocation relationship between different nodes. From the user entry node to the last implementation node, the behavior of each layer corresponds to the hierarchical trusted invocation association. For example, the management call process of OpenStack Neutron service to virtual network is shown in Figure 5. A user or compute node forwards a request information to controller node of the subscriber (such as Neutron-server) by calling RESTful API or Nova-API, where whether to call core API or extended API depends entirely on the type of virtual network service. And then the request information is forwarded to the plugin, where whether to make RPC (short for Remote Procedure Call) or not also depends on the specific needs of the user. If any RPC is called, the request is forwarded to one compute node or the management implementation interface of network node. The management implementation interface calls the corresponding virtualization process (such as Agent) according to the user's management request.

There is an initial state in a normal hierarchical trusted invocation association process. When an input or jump condition is detected, it will jump to the next state. Therefore, the hierarchical trusted invocation association process of a legitimate user's can be described by a finite state machine. The FSM is described as a five-tuple as follows:

$$M = (Q, \Sigma, \delta, q_0, F), \quad (1)$$

where $Q = \{q_0, q_1, \dots, q_n\}$ is a set of finite states. At any given moment, the network is in a certain state q_i ; $\Sigma =$

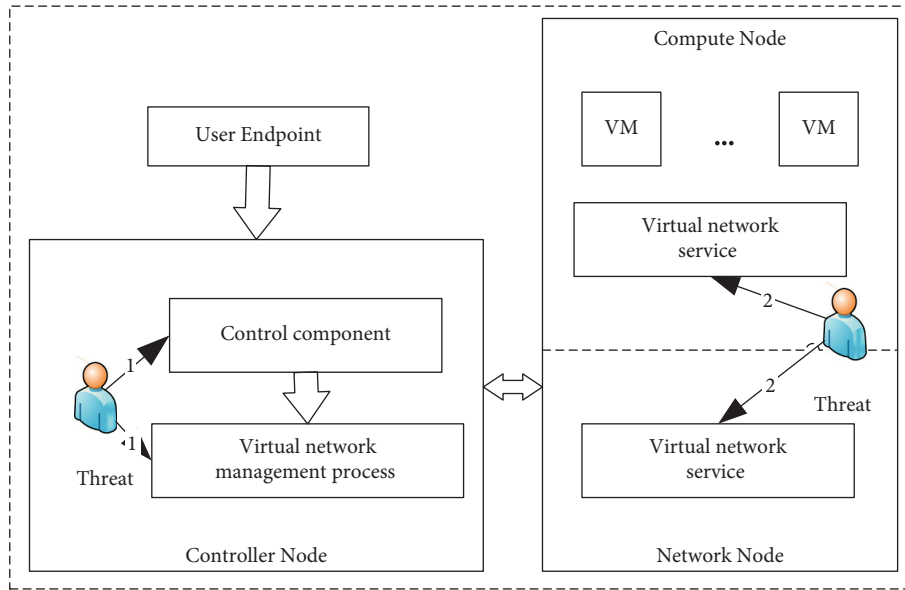


FIGURE 3: Potential insider threats in IaaS service deployment.

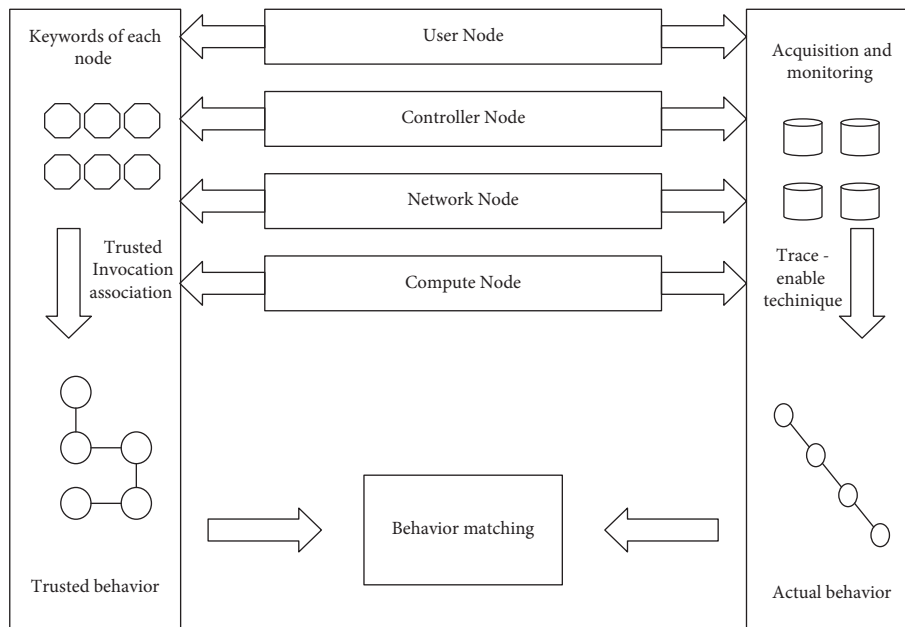


FIGURE 4: Workflow of VNGuarder.

$\{\sigma_1, \sigma_2, \dots, \sigma_n\}$ is a set of finite input σ_j . At any given moment, a given input j is received when the network behavior state changes. $\sigma: Q \times \Sigma \rightarrow Q$ is state transfer function. That says, in a certain state, state after a given input into a new state decided by the state transition function. $q_0 \in Q$ is the initial state of virtual network management behavior, from which the finite state machine begins to receive input. $F \in Q$ is the final state set of virtual network management behavior. A finite state machine does not receive new input after it has reached the final state.

The above FSM is specifically used to describe the trusted behavior of a normal user in virtual network cloud service. Taking the tenant's operation *useraction* on the virtual

network as an example, the tenant's trusted behavior description based on finite state machine is shown in Figure 6.

4.2.2. Tenant Trusted Behavior Generation Based on Hierarchical Invocation Association Analysis. As stated above, a trusted behavior of a normal user in virtual network service is described as an FSM, where each state represents the users' legitimate invocation operations at each level in cloud service. To generate this FSM, the following steps need to be completed. First, for each level of cloud service, the relevant source code-based keywords must be extracted as a keyword database. Second, when a user's trusted invocation operation

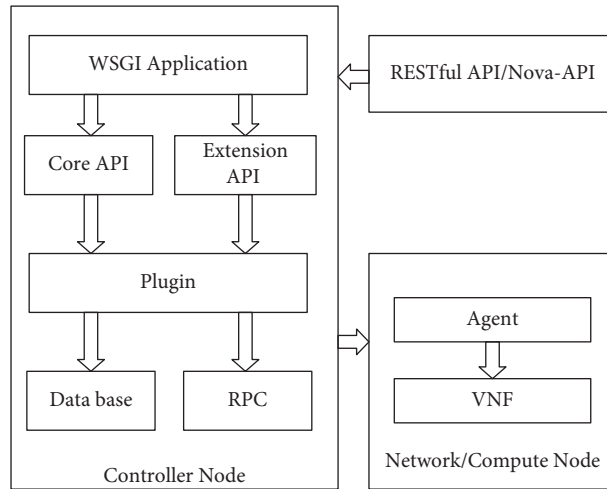


FIGURE 5: Hierarchical trusted invocation association.

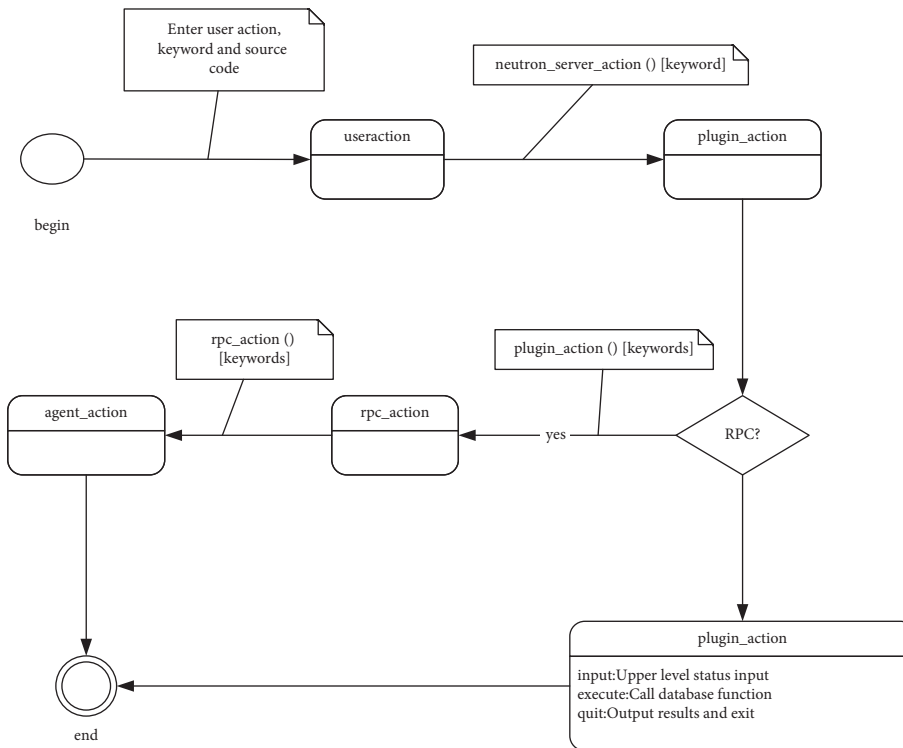


FIGURE 6: The trusted behavior description based on finite state machine for tenant's virtual network action operation.

is obtained, it will be traversed and matched with related to operation function keywords in the keyword databases from various levels of cloud service. And then the corresponding behavior state can be created through hierarchical trusted invocation association. When traversed to the bottom level, the end state constitutes the trusted behavior of a legitimate user. Finally, the generated tenant normal behavior model is stored in a trusted third-party module to prevent internal malicious actors from destroying it. The specific algorithm is shown in Table 1.

Let's take tenant's create network operation *create_network* as an example to illustrate the process of building trust behavior. As shown in Figure 7, the specific steps are as follows.

- (i) Create the root node in the user layer and enter the resource file to get the action *create_network*.
- (ii) Enter the base file in the virtual network control layer, find the corresponding interface function according to the obtained passed parameter *create_network*, and pass the obtained interface function to the lower layer as the parameter.
- (iii) The management process enters the management process document at the virtual network layer, finds the corresponding interface function according to the top of the incoming *create_network ()* function, and judges whether there

TABLE 1: Trusted behavior generation algorithm.

```

INPUT : User action, Keywords, Network service component source code
OUTPUT : Normal trusted behavior
Func createNormalBehavior(action string, keywords []string, source code *file){
    begin_state:= action
    normal_node:= begin_state
    for action!= nil{
        if next_state= next(begin_state) { //read next level
            normal_node = append(node, next_state)
        }
    }
    return normal_node
}

```

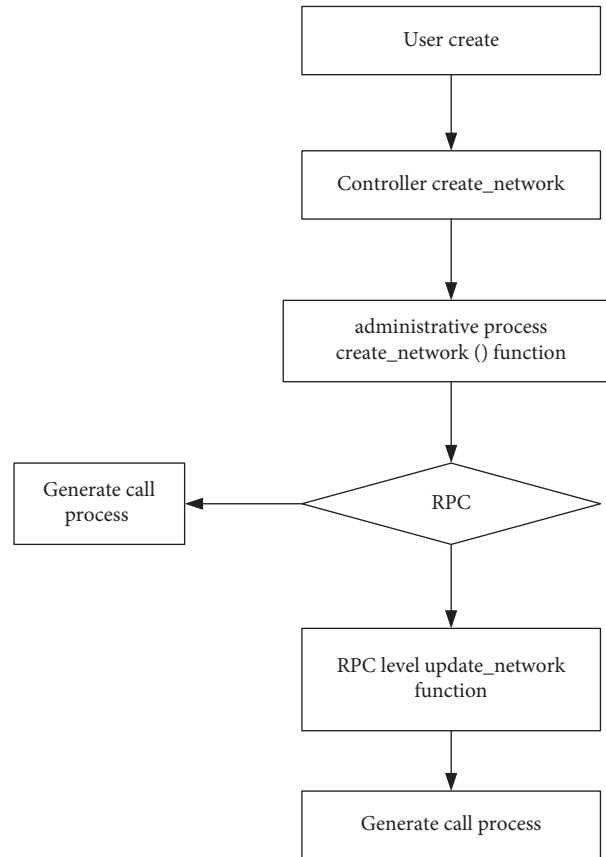


FIGURE 7: A trusted behavior generation instance on create network.

is any RPC behavior, therefore, generates choice node in this layer. If there is no RPC behavior, a complete calling process is generated; otherwise, the interface function of this layer is passed down as a parameter to the next layer.

- (iv) Enter the RPC file in the RPC layer, find the corresponding interface function *update_network ()*, and pass the obtained interface function as a parameter to the lower layer.
- (v) Enter the service process file in the service process layer, find the corresponding interface function through the parameters passed in the upper layer, and the agent creates the calling process and returns for the last layer of behavior.

With the development of cloud virtual network services, users' management requirements for virtual network may be change. When a user has a new operation to his virtual network, the user's trusted behavior can be updated based on the previous source code analysis and log information based on actual operations.

4.3. Actual Behavior Tracing. In VNGuarder, collection points are set up at all levels of virtual network service. For example, the remote call interface is added with the recorded time and the relevant interface information of the output operation. When the corresponding invocation operation occurs in each collection point, the relevant interface or function information and the calling time are printed into a log, that is, *log. (time: API/func)*. Normally, the log

information will be output to network service log of controller node, and management implementation log of network node and compute node in cloud platform. At the same time, the change of virtual network can be found in time through the deployment of real-time monitoring module.

$$\text{Actions} = \left\{ \frac{\{\text{time1: apil}\}}{\text{func1}} \right\}, \left\{ \frac{\{\text{time2: api2}\}}{\text{func2}} \right\}, \left\{ \frac{\{\text{time13: api3}\}}{\text{func3}} \right\}, \dots \dots \left\{ \frac{\{\text{timen : apin}\}}{\text{funcn}} \right\}. \quad (2)$$

The trace-enable algorithm is shown in Table 2.

4.4. Behavior Matching. Comparing the above traced-enable actual behavior with the results of hierarchical trusted invocation association, the current request can be considered as a normal request initiated by a legitimate user if a complete match can be made. If there is not an exact match, then it should be an unauthorized request, which may be malicious attacks from insiders. In addition, although malicious behavior can be triggered from any node in cloud platform, it will eventually be executed at the bottom. Therefore, in order to optimize the matching efficiency, behavior matching should start from the bottom. The insider threat detection algorithm based on behavior matching is shown in Table 3.

5. Evaluation

5.1. Experimental Environment. VNGuarder has been deployed in Openstack cloud platform and the specific implementation architecture is shown in Figure 8. In the experiment, three physical machines with 15 7400 CPU, 8G memory, and Ubuntu operating system were used. One of them acts as a controller node, while the other two act as network and compute nodes, respectively. The above-mentioned hierarchical trusted invocation association scheme and the proposed FSM algorithm are partially deployed to the controller node to generate the complete trusted behavior of a legitimate tenant as a matching object. At the same time, the trace-enable mechanism is deployed to the controller node, network node, and compute node to realize real-time monitoring to neutron agents, where collection points are set on the interfaces of neutron-server, neutron-plugin, RPC remote procedure call, neutron-agent management implementation interface, and virtualization management interface, etc. When real-time monitoring module detects related operations, each collection point will output the corresponding action and its time information. Finally, according to the information collected at each collection point, the behavior matching scheme is adopted to identify the behavior of malicious insiders by comparing the traced behavior with the previous normal behavior of the legitimate tenant. The interaction diagram is shown in Figure 9.

5.2. Validity Check. In this section, three groups of experiments are conducted to verify the validity of VNGuarder, when a malicious cloud insider is engaged in illegal call from different nodes and perform different malicious operations.

Once the virtual network configuration changes, the behavior information of each interface invoked is logged and then the actual behavior can be traced in order of time. That means

First, an experiment was performed to verify that malicious changes made by the malicious manager to the user's virtual machine port resource, such as changing the IP address of virtual interface, can be detected. In the experiment, we implement malicious operations by directly changing virtual devices in compute nodes. Under the OpenStack platform, the carrier body of port is realized by *ovs*. The *ovs agent* will receive the message from *neutron-server* and then configure the port, when the port changes. We configure the *ovs agent* such as adding ports to the original network by using the command "*ovs-vsctl add-port br0 eth*". As shown in Figure 10, after VNGuarder has been deployed, the actual invocation behavior is built by trace-enable mechanism, if the monitoring module dynamically monitors the port invocation occurred at some point in real time. When the behavior matching module completes the comparison between the actual invocation behavior and the hierarchy trusted invocation, it feedbacks the final detection result.

Second, an experiment was performed to verify that malicious configuration by the cloud operator to the user's subnet resources, such as changing the IP scope of the user's network. In the experiment, in order to achieve a malicious operation different from the first experiment, the neutron-plugin process of controller node is directly called by passing parameters for the plugin function, and then the corresponding driver and RPC process are called to realize the malicious configuration of subnet. As shown in Figure 10, the monitoring module of VNGuarder reenters real-time monitoring mode after having completed the first monitoring. It can detect and reflect the subnet's invocation behavior.

Finally, an experiment was performed to verify that the destruction of router resource by the cloud operator, such as adding ports to the router. Compared with the previous experiment, the neutron-server process is invoked in this experiment. A malicious insider can call the function in the controller class in the neutron-server, pass parameters by using the function, and then continue to call the plugin process, RPC process, and agent process to achieve the purpose of changing the router. As shown in Figure 10, the monitoring module in VNGuarder has been working. Whenever the router invocation behavior occurs, the VNGuarder approach can detect and report back.

5.3. Detection Rate. Repeated experiments were carried out on the three typical attack behaviors mentioned above and normal operation behaviors of users. Statistical analysis was performed on whether the malicious change operations of

TABLE 2: Trace-enable algorithm.

Input: log file, real-time monitoring information output: Current operation correlation information

```

func trace_back(log * file, time * file){
  for log! = nil{
    first_node,err: = traverse(log_i,time)
    if err! = nil{
      Break
    }else{
      Virtual_node:= append(node, first_node)
      next_node:= traverse(log_i,time)
    }
  } return virtual_node
}

```

TABLE 3: Behavior matching algorithm.

INPUT:virtual_node, normal_node output: Boolean value

```

Func match(virtual_node, normal_node){
  flag:= 0
  If virtual_node[:-1]! = nil{
  If virtual_node[:-1] == normal_node[:-1]{
  For virtual_node.next! = normal_node.next{
  flag = 1
  Break
  } if flag == 1{
  Return false
  }Else{ return true
  }
  }
}

```

port resource, subnet resource, router resource, and the normal operation of users could be correctly monitored after the deployment of VNGuarder. As shown in Figure 11, the attacks caused by the above three typical operations can be correctly monitored more than 90%, and there are fewer misjudgments for normal user’s operations with VNGuarder.

5.4. Impact on Virtual Network Creation Time. In VNGuarder, the trace-enable mechanism extracts process information and generates a log file, which provides some performance for virtual machine runs and virtual network creation. In the experiment, we have counted network creation time with and without VNGuarder deployment. We created a virtual network for a tenant in the cloud environment, and recorded the time from the start of creating a virtual machine to the completion of the virtual network as the virtual network creation time, and conducted the experiment 10 times. As shown in Figure 12, where the horizontal axis represents the number of experimental rounds, and the vertical axis represents the time to set up a virtual network. The results show that VNGuarder deployment has little effect on the virtual network creation time.

5.5. CPU and Memory Performance Loss. VNGuarder is deployed on different cloud hosts and only responds when the virtual network is configured. Therefore, the cloud host

performance, such as CPU usage and memory usage, is affected only when the virtual network configuration is performed. Otherwise, the cloud host performance is not significantly affected. At the same time, VNGuarder only monitors the working nodes in real-time which can reduce unnecessary CPU and memory usage. In this experiment, some performance detection tools are used to conduct statistical analysis of CPU and memory consumption and compare the performance changes before and after deployment of VNGuarder. The benchmark is calculated as follows.

As shown in Figure 13 and Figure 14, the horizontal axis represents the experiment times, the vertical axis represents the average decline rate of CPU or memory which reflects the performance changes without network operation and with the network configuration after VNGuarder has been deployed. The results show that the virtual machine is operated normally and the CPU or memory performance is basically unaffected when the network configuration operation is not involved. When the network configuration operation exists, the performance overhead will change significantly, but the range of change is still within the acceptable range.

5.6. Functional Comparison with the Existing Work. In this subsection, we compare VNGuarder with the existing related work [27–30], including the internal threat points of cloud system that different solutions focus on, whether they can monitor single node or multiple nodes of cloud platform, and whether they support behavior association tracking. The Functional Comparison shown in Table 4.

6. Discussion

At present, machine learning methodologies are introduced for mitigating insider threats [31], where multiple data sources are used to find associations. For example, machine learning can analyse whether all intrarelatational behavior is malicious based on the behavior of insiders who have already been flagged as at risk. However, this approach can produce false positives because internal staff are often assigned new permissions when they are assigned new business. Therefore, this kind of methods needs to increase data continuously to improve the accuracy of the training model algorithm. In contrast to the uncertainties associated with machine learning methods, VNGuarder can detect internal threat without changing the hierarchy trusted behavior model based on individual permissions.

In our early work [21], we focus on the protection of data in cloud compute nodes, through the use of behavior trace and access control means to protect the data in a computing node, although this method can effectively avoid the internal malicious personnel directly to the protection of data storage computing node but ignore the internal management of malicious or operations staff through the virtual network malicious way destroy the tenant management network, and then get the user data.

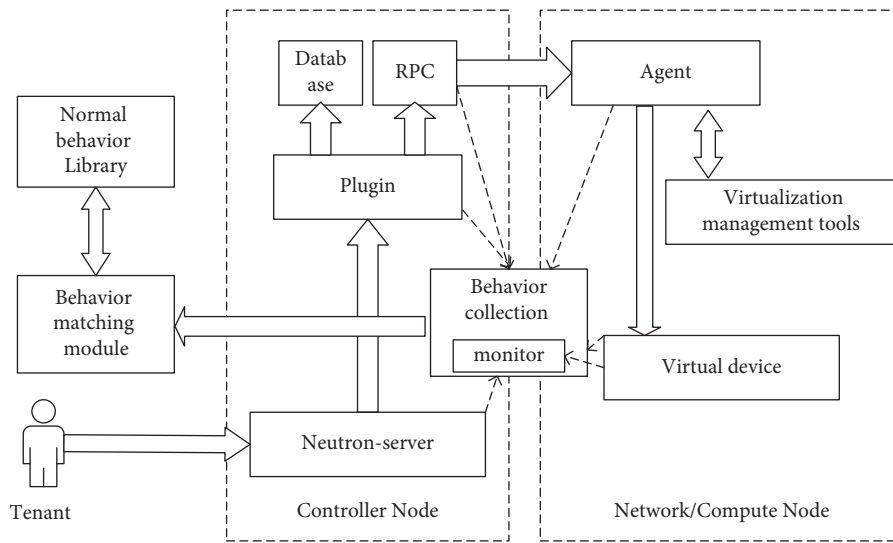


FIGURE 8: VNGuarder deployment in Openstack cloud platform.

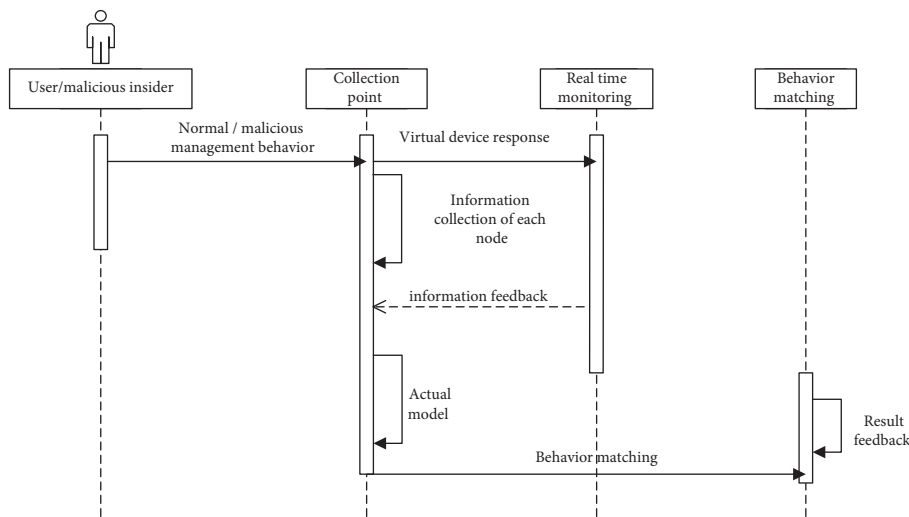


FIGURE 9: The interaction diagram in VNGuarder.

```

openstack@openstack:~$ ./VNGuarder_monitor -d 2021-05-20 -t 10:06:25 -v port
the time:2021-05-20 10:06:25,normal operation on port
openstack@openstack:~$ ./VNGuarder_monitor -d 2021-05-20 -t 11:32:30 -v port
the time:2021-05-20 11:32:30,abnormal and illegal operation on port
openstack@openstack:~$ ./VNGuarder_monitor -d 2021-05-20 -t 11:32:30 -v port
the time:2021-05-20 11:32:30,abnormal and illegal operation on port
    
```

FIGURE 10: The effectiveness verification of VNGuarder in monitoring resource changes such as port, subnet, and router.

Therefore, finding and blocking virtual network malicious management and configuration has become a key to improve the overall prevention of internal malicious threats in cloud computing environment.

At the same time, VNGuarder can effectively reduce the number of threats judged by the system because it compares

and determines whether there is a threat in the trusted hierarchical model only after the collection point has collected the corresponding information. In the traditional way, after the data is obtained, the threat judgment is completed through continuous screening. Therefore, VNGuarder can effectively reduce the complexity in practical application.

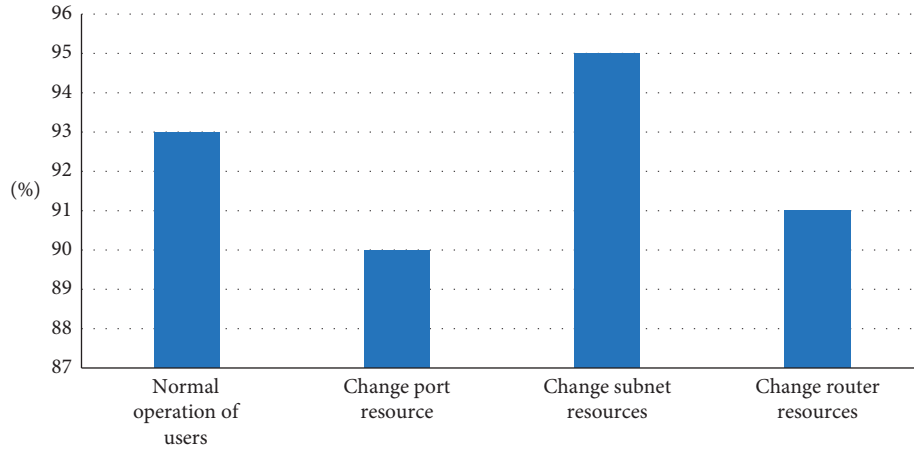


FIGURE 11: Detection rate under different Network management operations with VNGuarder.

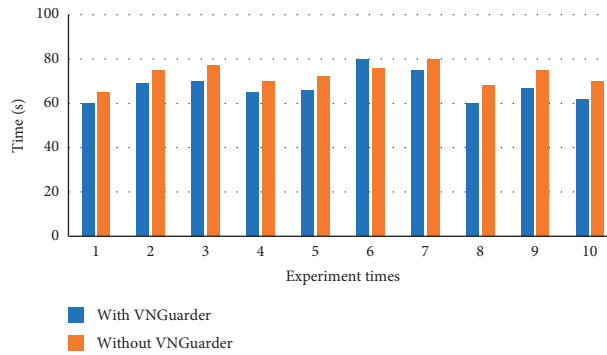


FIGURE 12: Virtual network creation time in 10 rounds under OpenStack with VNGuarder and without VNGuarder.

$$CPU = LCPU - PreCPU$$

$$Memory = LMemory - PreMemory$$

* CPU= Change in utilization rate of CPU

* LCPU= After VNGuarder of CPU

* PreCPU= Per VNGuarder of CPU

* Memory= Change in utilization rate of Memery

* LMemory= After VNGuarder of Memery

* PreMemory= Per VNGuarder of Memery

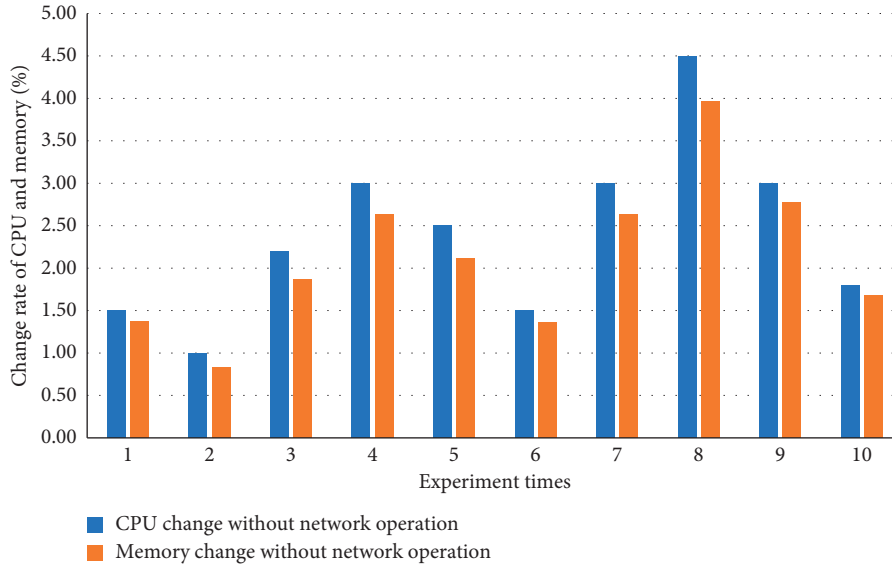


FIGURE 13: Performance change of CPU and Memory in 10 rounds under Openstack without network operation.

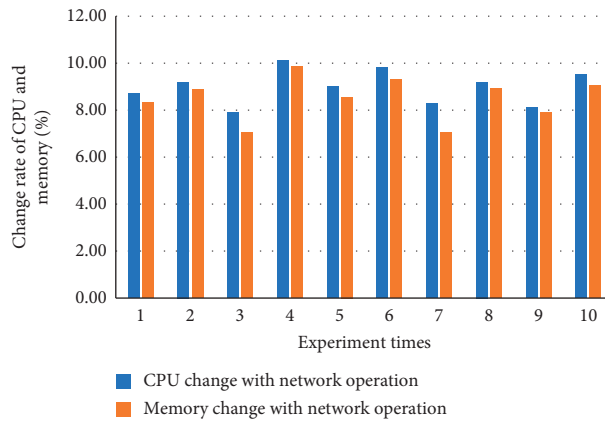


FIGURE 14: Performance change of CPU and Memory in 10 rounds under Openstack with network operation.

TABLE 4: Functional comparison.

Method	Internal threat point	Monitor single node	Monitor multinode	Behavior association tracking
[27]	VM service	Yes	No	No
[28]	VM image	Yes	Yes	No
[29]	Tenants and managers	Yes	No	No
[30]	Tenants and managers	Yes	No	No
VNGuarder	Virtual network service	Yes	Yes	Yes

7. Conclusions and Future Work

Virtual network security in the cloud environment will eventually affect on edge-assisted Internet of things applications. There is little research on insider threat attack to virtual network management. This paper proposes an approach named VNGuarder to protect IaaS users’ private networks from being maliciously changed or destroyed by insider. In VNGuarder, a trace-enable mechanism is introduced to integrate real-time monitoring and log analysis. A hierarchical trusted invocation association method based on tenants’ normal behavior is proposed to provide a basis

for discovering illegal invocation of insiders to virtualization management tools and virtual network-related processes. VNGuarder has been implemented on OpenStack platform, and the effectiveness and performance evaluation have been verified by experiments. The results show that VNGuarder can identify several important insider threats like illegal management and configuration modification to virtual network device through virtualization tools with a small performance overhead.

The ongoing work is twofold. On the one hand, the normal hierarchical trusted invocation behavior is constructed based on the source analysis of OpenStack Neutron

service. However, with the continuous iteration of Neutron service, the source code logic structure may change. We plan to design a trusted invocation behavior modelling method based on machine learning algorithm, in order to improve the efficiency and adaptability of trusted behavior construction. On the other hand, VNGuarder can find a malicious insiders' behavior only when the insider threat is triggered. Hence, a tenant's private network may be exposed under malicious threats for a period of time before he receives attack detection feedback. In this case, we need to further fine-grain and classify the management operations of virtual network, including determining which management interface cloud managers can invoke and which cannot. For the latter, an effective access control mechanism should be designed to control access to reduce the risk of random management and misoperation by malicious insiders.

Data Availability

All data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Science Foundation of China under Grant 61502017 in part by the Natural Science Foundation of Beijing Municipality under Grant M21039

References

- [1] C. Wang, D. Wang, G. Xu, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy for industry 4.0," *Science China Information Sciences*, vol. 65, no. 1, Article ID 112301, 2022.
- [2] W. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, and C. Su, "Blockchain-based reliable and efficient certificateless signature for IIoT devices," *IEEE Transactions on Industrial Informatics*, pp. 1551–3203, 2021.
- [3] W. Li, W. Meng, C. Su, and L. F. Kwok, "Towards false alarm reduction using fuzzy if-then rules for medical cyber physical systems," *IEEE Access*, vol. 6, pp. 6530–6539, 2018.
- [4] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor Authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, 2020.
- [5] W. Wang, H. Huang, L. Zhang, and C. Su, "Secure and efficient mutual authentication protocol for smart grid under blockchain," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2681–2693, 2021.
- [6] Z. Lian, W. Wang, and C. Su, "COFEL: communication-efficient and optimized federated learning with local differential privacy," in *Proceedings of the ICC 2021-IEEE International Conference on Communications*, pp. 1–6, IEEE, Montreal, Canada, June 2021.
- [7] T. Wang, Y. Quan, X. S. Shen, T. R. Gadekallu, W. Wang, and K. Dev, "A privacy-enhanced retrieval technology for the cloud-assisted Internet of things," *IEEE Transactions on Industrial Informatics*, page, 2021.
- [8] "Open Stack Networking architecture-Security Guide documentation (openstack.org)," 2021, <https://docs.openstack.org/security-guide/networking/architecture.html>.
- [9] L. Rui, Y. Jiawei, H. Dongjie, and C. Hua, "On virtual network access based on open VSwitch," *Computer Applications and Software*, vol. 31, no. 5, pp. 308–311, 2014.
- [10] P. Goyal and A. Goyal, "Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark," in *Proceedings of the 2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 77–81, Girne, Northern Cyprus, September 2011.
- [11] M. R. Sutradhar, N. Sultana, H. Dey, and H. Arif, "A new version of kerberos authentication protocol using ECC and threshold cryptography for cloud security," in *Proceedings of the 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pp. 239–244, Kitakyushu, Japan, June 2018.
- [12] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and Software-Defined Networking," *Computer Networks*, vol. 81, pp. 308–319, 2015.
- [13] F. Sabahi, "Cloud computing security threats and responses," in *Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks*, pp. 245–249, Xi'an, China, May 2011.
- [14] N. Agrawal and S. Tapaswi, "Low rate cloud DDoS attack defense method based on power spectral density analysis," *Information Processing Letters*, vol. 138, pp. 44–50, 2018.
- [15] M. O. Alassafi, R. AlGhamdi, A. Alshdadi, A. Al Abdulwahid, and S. T. Bakhsh, "Determining factors pertaining to cloud security adoption framework in government organizations: an exploratory study," *IEEE Access*, vol. 7, pp. 136822–136835, 2019.
- [16] J. Tang, Y. Cui, Q. Li, K. Ren, J. Liu, and R. Buyya, "Ensuring security and privacy preservation for cloud data services," *ACM Computing Surveys*, vol. 49, no. 1, pp. 1–39, 2017.
- [17] A. Pandey and S. Srivastava, "An approach for virtual machine image security," in *Proceedings of the 2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014)*, pp. 616–623, Ajmer, India, July 2014.
- [18] S. Baik, Y. Lim, J. Kim, and Y. Lee, "Adaptive flow monitoring in SDN architecture," in *Proceedings of the 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 468–470, Busan, Korea (South), August 2015.
- [19] K. Zhang, H. F. Wang, Y. C. Ma, and Z. Li, "The status and development trends of virtual private network technology," *Key Engineering Materials*, vol. 474-476, pp. 79–82, 2011.
- [20] T. Haibo Chen and H. Chen, "FlexCore: dynamic virtual machine scheduling using VCPU ballooning," *Tsinghua Science and Technology*, vol. 20, no. 1, pp. 7–16, 2015.
- [21] L. Lin, S. Li, B. Li, J. Zhan, and Y. Zhao, "TVGuarder: a trace-able virtualization protection framework against insider threats for IaaS environments," *International Journal of Grid and High-Performance Computing (IJGHPC)*, vol. 8, no. 4, pp. 1–20, 2016.
- [22] A. Hussein, I. H. Elhadj, A. Chehab, and A. Kayssi, "SDN security plane: an architecture for resilient security services," in *Proceedings of the 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, pp. 54–59, Berlin, Germany, April 2016.

- [23] Y. F. Shao and Z. Jia, "Research on software defined network security technology," *Radio Engineering*, vol. 46, no. 4, pp. 13–17, 2016.
- [24] M. B. de Carvalho, R. P. Esteves, G. da Cunha Rodrigues, L. Z. Granville, and L. M. R. Tarouco, "A cloud monitoring framework for self-configured monitoring slices based on multiple tools," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, pp. 180–184, Zurich, Switzerland, October 2013.
- [25] Csa Top Threats Working Group, "The treacherous twelve cloud computing top threats in 2016," 2016, <https://cloudsecurityalliance.org/artifacts/the-treacherous-twelve-cloud-computing-top-threats-in-2016/>.
- [26] J. Bouche and M. Kappes, "Attacking the cloud from an insider perspective," in *Proceedings of the 2015 Internet Technologies and Applications (ITA)*, pp. 175–180, Wrexham, UK, September 2015.
- [27] Z. Yu, W. Zhang, and H. Dai, "A trusted architecture for virtual machines on cloud servers with trusted platform module and certificate authority," *Journal of Signal Processing Systems*, vol. 86, no. 2-3, pp. 327–336, 2017.
- [28] R. K. Hussein, A. Alenezi, H. F. Atlam, M. Q. Mohammed, R. J. Walters, and G. B. Wills, "Toward confirming a framework for securing the virtual machine image in cloud computing," *Advances in Science, Technology and Engineering Systems Journal*, vol. 2, no. 4, pp. 44–50, 2017.
- [29] V. Kansal and M. Dave, "Proactive DDoS attack detection and isolation," in *Proceedings of the 2017 International Conference on Computer, Communications and Electronics (Comptelix)*, pp. 334–338, Jaipur, India, July 2017.
- [30] M. Singh, B. M. Mehtre, and S. Sangeetha, "Insider threat detection based on user behavior analysis," in *Machine Learning, Image Processing, Network Security and Data Sciences. MIND 2020. Communications in Computer and Information Science*, A. Bhattacharjee, S. Borgohain, B. Soni, G. Verma, and XZ. Gao, Eds., vol. 1241pp. 559–574, 2020.
- [31] Y. Xin, L. Kong, Z. Liu et al., "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.

Research Article

Cryptanalysis of an Efficient and Secure Certificateless Aggregate Signature-Based Authentication Scheme for Vehicular Ad Hoc Networks

Xiaodong Yang ¹, Aijia Chen,¹ Zhisong Wang,¹ Xiaoni Du,¹ and Caifen Wang^{1,2}

¹College of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China

²College of Big Data and Internet, Shenzhen Technology University, Shenzhen 518118, China

Correspondence should be addressed to Xiaodong Yang; y200888@163.com

Received 27 August 2021; Revised 9 February 2022; Accepted 23 March 2022; Published 14 April 2022

Academic Editor: Ding Wang

Copyright © 2022 Xiaodong Yang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Vehicular ad hoc networks (VANETs) apply the Internet of Things technology to provide secure communication among the vehicles, thereby improving the safe driving of vehicles, the utilization of traffic resources, and the efficiency of information services. Due to the openness and vulnerability of wireless networks in VANETs, messages in sharing and transmission are easily attacked and destroyed. Therefore, a large number of schemes for VANETs were proposed to protect the privacy of vehicles and ensure the authentication, integrity, and nonrepudiation of messages. However, most of these schemes have serious security flaws or poor performance issues. Recently, Thumbur et al. presented an efficient certificateless aggregate signature-based authentication scheme for VANETs and then gave the detailed security proof (IEEE Internet of Things Journal, vol. 8, no. 3, pp. 1908–1920, 2021). In this article, we analyze the security of Thumbur et al.'s scheme and demonstrate that it has significant security problems. Specifically, their scheme cannot resist public key replacement attacks from external adversaries, and it is not secure against coalition attacks from malicious vehicles. Then, we improve Thumbur et al.'s scheme to address the security weaknesses. According to the security and performance analysis, the improved scheme enhances the security while maintaining the performance of the original scheme.

1. Introduction

As a new type of mobile network, vehicular ad hoc networks (VANETs) utilize the Internet of Things technology and intelligent transportation mechanisms to achieve secure communication among the vehicles. As a result, it can efficiently improve the safety of vehicle driving, minimize road traffic congestion, avoid traffic accidents, reduce the work intensity of drivers, and provide secure, economic, comfortable, and fast transportation services [1]. VANETs generally involve four types of entities: vehicles with on-board units (OBUs), roadside units (RSUs) installed along the road, a trusted authority (TA) for system initialization, and application servers (ASs) for data analysis [2]. In VANETs, a vehicle employs an OBU to periodically

broadcast its vehicle-related information (such as speed, direction, mileage, fuel consumption, location, and brake pad pressure) and real-time road condition information to nearby vehicles and RSUs every 100–300 milliseconds.

However, VANETs face numerous security challenges and privacy threats while improving traffic management and road safety [3–5]. Because wireless networks in VANETs are open, the information shared among the vehicles is easy to be attacked and destroyed, such as forgery, tampering, injection, and replay attack. If false traffic information is transmitted by an attacker, it will cause a potential traffic accident or even endanger the safety of drivers and passengers [6]. Hence, it is critical to ensure the integrity and authenticity of traffic data in VANETs. Furthermore, the vehicle's private information, such as its real identity, should

be protected to prevent it from being misused, leaked, or accessed by unauthorized entities. At the same time, TA is capable of tracking down the real identity of the vehicle that issued a controversial message [7, 8]. Therefore, conditional privacy must be provided by VANETs.

Digital signature is a cryptographic technology that provides data integrity, anonymity, nonrepudiation, identity authentication, and other security services. Various signature technologies, such as PKC-based and identity-based cryptographic frameworks, are used to construct authentication schemes for VANETs to ensure identity authentication, message integrity, privacy, and traceability. When these schemes are deployed in real-world circumstances, there are some problems, such as complex certificate management or inherent key escrow [9]. Certificateless aggregation signature (CLAS) can not only solve these problems in PKC-based and identity-based settings but also aggregate different signatures of multiple messages into a short signature [10]. Hence, CLAS greatly reduces the computational and communication overhead of signature verification and improves storage efficiency. Due to the advantages of CLAS in terms of security and performance, CLAS is considered to be an efficient and viable solution to some security problems in VANETs.

In recent years, researchers have proposed some CLAS-based authentication schemes for VANETs. Unfortunately, most of these schemes [4, 11–14] are not suitable for VANETs due to time-consuming bilinear operations. Based on the additive elliptic curve group, Cui et al. [15] proposed an authentication scheme for VANETs without pairings. However, Kamil and Ogundoyin [16] found that Cui et al.'s scheme [15] is unable to withstand attacks from a malicious key generation centre (KGC) and presented an improved scheme. Zhao et al. [17] showed that the improved authentication scheme of Kamil and Ogundoyin [16] is still insecure against forgery attacks and then constructed an improved scheme to remedy these security vulnerabilities. In 2021, the flaws in the design of Zhao et al.'s scheme [17] were discovered by Thumbur et al. [18]. Furthermore, Thumbur et al. [18] presented a novel pairing-free CLAS-based authentication scheme for VANETs. Unfortunately, we prove that their scheme is not secure by giving concrete attacks; hence, it fails to achieve the expected security goals.

In this article, we analyze the security of Thumbur et al.'s scheme [18] and demonstrate that it is not secure against public key replacement attacks of external attackers and coalition attacks of malicious vehicles. To overcome these two kinds of forgery attacks, we also give an improved scheme. The main contributions of our work are as follows:

- (1) We show that the CLAS-based authentication scheme for VANETs of Thumbur et al. [18] cannot resist public key replacement attacks
- (2) We provide an attack method to demonstrate that Thumbur et al.'s scheme [18] is not secure against coalition attacks launched by malicious vehicles.
- (3) We improve Thumbur et al.'s scheme [18] to address the security issues
- (4) Our enhanced scheme meets various security requirements in VANETs

TABLE 1: Summary of notations.

Symbols	Descriptions
P	A prime number
F_p	A finite field of order p
G	A cyclic elliptic curve group
P	A generator of G
$H_i (i = 0, \dots, 4)$	Hash functions
(s, b)	Two master secret keys
$(P_{\text{pub}}, T_{\text{pub}})$	Two master public keys
RID_i	The real identity of a vehicle V_i
PID_i	The pseudoidentity of V_i
vsk_i	The partial private key of V_i
x_i	The secret value of V_i
vpk_i	The public key of V_i
m_i	A message
σ_i	A single signature on m_i
σ_{agg}	An aggregate signature
ΔT_i	The validity period of PID_i
$\mathcal{A}_1, \mathcal{A}_2$	Type I and type II adversaries

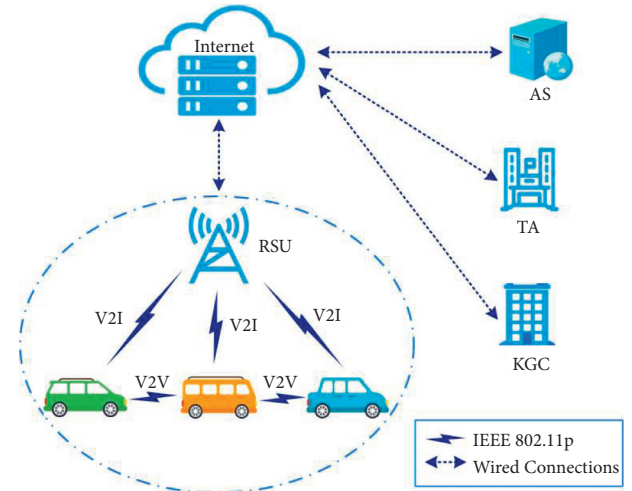


FIGURE 1: The system model of the CLAS-based authentication scheme for VANETs.

The rest of this article is organized as follows. Section 2 describes some preliminaries. Section 3 introduces Thumbur et al.'s scheme [18], and its cryptanalysis is given in Section 4. Section 5 presents our improved scheme and discusses its security and efficiency. Finally, Section 6 concludes this article.

2. Preliminaries

In this section, we briefly introduce the elliptic curve group, the system model, and the formal definition of the CLAS-based scheme for VANETs. Some key symbols are listed in Table 1.

2.1. Elliptic Curve Group and Computational Assumption. Assume that the order of a finite field F_p is a prime $p > 3$. All solutions of the equation $y^2 = (x^3 + ax + b) \bmod p$ form an elliptic curve $E(F_p)$, where $a, b \in F_p$ and $4a^3 + 27b^2 \neq 0$. The

cyclic elliptic curve group
 $G = \{(x, y) \in E(F_p): x, y \in F_p\} \cup \{O\}$ is composed of all points on $E(F_p)$ and an infinite point O . According to the chord-and-tangent rule [19], the addition operation in G is defined as $R = P + Q$, where $P, Q \in G$. The operation $kP = P + \dots + P$ (k times) is called a scalar multiplication in G . For convenience, let P be a generator of G . The computational assumption in G is described as follows.

Elliptic Curve Discrete Logarithm Problem (ECDLP). Given two elements P and $Q = xP$ in G , it is intractable to calculate $x \in Z_p^*$ in polynomial time.

2.2. System Model. As described in Figure 1, the system model of a CLAS-based authentication scheme for VANETs contains five participants: TA, KGC, AS, RSU, and the vehicle equipped with OBU. Communication modes can be divided into two types: upper-level and lower-level. The communication between TA, KGC, AS, and RSU mainly adopts the upper-level mode, which is carried out over a secure wired communication network. The lower-level communication uses the dedicated short-range communication (DSRC) protocol (such as IEEE 802.11P), which mainly occurs in the communication between the vehicle and another, as well as between the vehicle and the RSU. It is worth noting that TA, KGC, and AS are three entities with sufficient computing power, communication bandwidth, and storage capacity. OBU is a resource-constrained device, but RSU outperforms OBU in terms of battery, computation, communication, and storage. The five participants are described in detail below.

- (1) *TA*. It initializes the system, registers vehicles and RSUs, issues pseudonym identities for vehicles, and traces the real identities of malicious vehicles. Because TA is usually the vehicle management department of the local government, it is regarded as a trustworthy authority.
- (2) *KGC*. It is an entity independent of TA, which is primarily responsible for producing each vehicle's partial private key.
- (3) *AS*. It is an application server that collects and analyzes traffic-related messages sent by RSUs.
- (4) *RSU*. This kind of wireless communication equipment is typically situated on the side of a road or at a crossroads and is mainly responsible for managing the communication of all vehicles within its communication range.
- (5) *Vehicle*. It uses the DSRC protocol to broadcast traffic-related messages frequently, such as road conditions, vehicle speed, and location.

2.3. Formal Definition of the CLAS-Based Authentication Scheme for VANETs. As defined in [16, 18], a CLAS-based authentication scheme for VANETs is made up of the nine following algorithms:

- (1) *System Setup*. Given a security parameter λ , TA and KGC execute this algorithm to produce system parameters $params$, two system master keys (s, b) , and two system master public keys (P_{pub}, T_{pub}) .
- (2) *Pseudonym Identity Generation*. On receiving the real identity RID_i of a vehicle V_i , TA executes this algorithm to create the pseudoidentity PID_i for V_i .
- (3) *Partial Private Key Generation*. Upon receiving the pseudoidentity PID_i of a vehicle V_i , this algorithm is executed by KGC to create a partial private key vsK_i for V_i .
- (4) *Set Secret Value*. This algorithm is run by a vehicle V_i to create its secret value x_i .
- (5) *Vehicle Key Generation*. A vehicle V_i runs this algorithm to produce its private key vsK_i and corresponding public key vpK_i .
- (6) *Signature Generation*. This algorithm is executed by a vehicle V_i . Given a message m_i , V_i uses its private key vsK_i to create a single signature σ_i on m_i .
- (7) *Signature Verification*. Taking a message m_i , a single signature σ_i , and the pseudonym identity and public key pairs (PID_i, vpK_i) as inputs, the RSU or other vehicles execute this algorithm to output *True* if σ_i is a valid single signature or *False* otherwise.
- (8) *Aggregate Signature Generation*. Taking n single signatures $\{\sigma_i\}_{i=1}^n$ on n messages $\{m_i\}_{i=1}^n$ as inputs, RSU executes this algorithm to generate an aggregate signature σ_{agg} .
- (9) *Aggregate Signature Verification*. Upon receiving an aggregate signature σ_{agg} , n messages $\{m_i\}_{i=1}^n$, and the pseudoidentity and public key pairs $\{(PID_i, vpK_i)\}_{i=1}^n$ of n vehicles, the AS runs this algorithm to output *True* if σ_{agg} is a valid aggregate signature or *False* otherwise.

As described in [16–18], an authentication scheme for VANETs achieves a wide range of security requirements, such as identity anonymity, traceability, message integrity, and authenticity. Based on these security requirements, a secure CLAS-based authentication scheme for VANETs can withstand the following attackers:

- (1) *Type I Adversary*. This is a type of external attacker that can carry out public key replacement attacks. The attacker has the ability to get the vehicle's secret value or replace its public key. However, the attacker is not allowed to know the KGC's master secret key and the vehicle's partial private key.
- (2) *Type II Adversary*. The attacker is usually a malicious KGC and thus possesses KGC's master key. However, the attacker is unable to compromise the vehicle's secret value and replace the vehicle's public key.
- (3) *Inside Adversary*. This type of attacker is two or more malicious vehicles that launch the coalition attack. The attackers can create a legitimate aggregate signature by providing a few invalid single signatures.

3. Review of Thumbur et al.'s Scheme

In this section, we briefly describe the CLAS-based authentication scheme for VANETs proposed by Thumbur et al. [18].

- (1) *System Setup.* The following operations are carried out in collaboration between TA and KGC:
 - (a) TA and KGC negotiate to choose an elliptic curve group G with prime order p and select a generator P of G and four hash functions $H_0, H_1, H_2, H_3: \{0, 1\}^* \rightarrow Z_p^*$.
 - (b) KGC selects a random value $s \in Z_p^*$ as its master secret key and calculates the corresponding master public key $P_{\text{pub}} = sP$.
 - (c) TA randomly picks $b \in Z_p^*$ as its master secret key and then sets $T_{\text{pub}} = bP$ as its master public key.
 - (d) Output system parameters $\text{params} = \{p, G, P, T_{\text{pub}}, P_{\text{pub}}, H_0, H_1, H_2, H_3\}$.
- (2) *Pseudonym Identity Generation.* To protect the privacy of the identity, the vehicle uses the pseudonym identity issued by TA to hide the real identity for anonymous message communication.
 - (a) A vehicle V_i selects a random value t_i from Z_p^* and calculates $\text{PID}_{i,1} = t_iP$; then it hides its unique real identity RID_i by calculating $K_i = t_iT_{\text{pub}} \oplus \text{RID}_i$ and finally transmits $\{K_i, \text{PID}_{i,1}\}$ to TA.
 - (b) TA uses its master secret key b to obtain V_i 's real identity by calculating $\text{RID}_i = b\text{PID}_{i,1} \oplus K_i$. If RID_i is legal, TA calculates $\text{PID}_{i,2} = H_0(b\text{PID}_{i,1}, \Delta T_i) \oplus \text{RID}_i$, sets V_i 's pseudonym identity $\text{PID}_i = \{\text{PID}_{i,1}, \text{PID}_{i,2}, \Delta T_i\}$, and transmits it to KGC, where ΔT_i is the validity period of PID_i .
- (3) *Partial Private Key Generation.* Upon receipt of the pseudonym identity PID_i for a vehicle V_i , KGC executes the following steps:
 - (a) Pick $r_i \in Z_p^*$ at random and calculate $R_i = r_iP$.
 - (b) Calculate $h_{1i} = H_1(\text{PID}_i, R_i, P_{\text{pub}})$.
 - (c) Calculate the partial private key of V_i as $\text{psk}_i = (r_i + sh_{1i}) \bmod p$.
 - (d) Send $\{\text{PID}_i, R_i, \text{psk}_i\}$ to V_i secretly.

Note that V_i checks the validity of psk_i from KGC by verifying whether $\text{psk}_iP = R_i + h_{1i}P_{\text{pub}}$ holds.
- (4) *Set Secret Value.* A vehicle V_i randomly picks $x_i \in Z_p^*$ as its secret value and calculates $X_i = x_iP$.
- (5) *Vehicle Key Generation.* A vehicle V_i performs the following to produce its private key and public key:
 - (a) Calculate $h_{2i} = H_2(\text{PID}_i, X_i)$.
 - (b) Calculate $Q_i = R_i + h_{2i}X_i$.
 - (c) Set its public key $\text{vpk}_i = (Q_i, R_i)$.
 - (d) Set its private key $\text{vsk}_i = (\text{psk}_i, x_i)$.

- (6) *Signature Generation.* To achieve secure communication, vehicle V_i with PID_i performs the following signature operations on each traffic-related message m_i sent by itself:
 - (a) Pick $u_i \in Z_p^*$ at random and calculate $U_i = u_iP$.
 - (b) Calculate $h_{2i} = H_2(\text{PID}_i, X_i)$.
 - (c) Calculate $h_{3i} = H_3(\text{PID}_i, m_i, \text{vpk}_i, U_i, T_i)$, where T_i is the current timestamp.
 - (d) Calculate $s_i = (u_i + h_{3i}(\text{psk}_i + h_{2i}x_i)) \bmod p$.
 - (e) Output a single signature $\sigma_i = (U_i, s_i)$ on m_i .
 - (f) Send $\{m_i, \sigma_i, T_i, \text{vpk}_i, \text{PID}_i\}$ to the nearby RSU.
- (7) *Signature Verification.* Given a single signature $\sigma_i = (U_i, s_i)$ on a message m_i at timestamp T_i under a pseudonym PID_i and a public key $\text{vpk}_i = (Q_i, R_i)$, the verifier performs the following verification operations if ΔT_i of PID_i and T_i are both valid:
 - (a) Calculate $h_{1i} = H_1(\text{PID}_i, R_i, P_{\text{pub}})$.
 - (b) Calculate $h_{3i} = H_3(\text{PID}_i, m_i, \text{vpk}_i, U_i, T_i)$.
 - (c) Check the single signature verification equation below:

$$s_iP = U_i + h_{3i}(Q_i + h_{1i}P_{\text{pub}}). \quad (1)$$

The verifier accepts σ_i if it holds; otherwise, σ_i is considered invalid.

- (8) *Aggregate Signature Generation.* After receiving n single signatures $\{\sigma_i = (U_i, s_i)\}_{i=1}^n$ on messages $\{m_i\}_{i=1}^n$ from n vehicles, the RSU performs the following aggregation signature operations:
 - (a) Calculate $s = \sum_{i=1}^n s_i$.
 - (b) Output the aggregate signature on $\{m_i\}_{i=1}^n$ as $\sigma_{\text{agg}} = (U_1, \dots, U_n, s)$.
- (9) *Aggregate Signature Verification.* Given an aggregate signature $\sigma_{\text{agg}} = (U_1, \dots, U_n, s)$ on messages $\{m_i\}_{i=1}^n$ under n tuples $\{(\text{vpk}_i = (Q_i, R_i), \text{PID}_i)\}_{i=1}^n$, the AS performs the following aggregate signature verification operations if ΔT_i and T_i are in the period of validity:
 - (a) Calculate $h_{1i} = H_1(\text{PID}_i, R_i, P_{\text{pub}})$ for all $i \in \{1, \dots, n\}$.
 - (b) Calculate $h_{3i} = H_3(\text{PID}_i, m_i, \text{vpk}_i, U_i, T_i)$ for all $i \in \{1, \dots, n\}$.
 - (c) Check the aggregation signature verification equation below:

$$sP = \sum_{i=1}^n U_i + \sum_{i=1}^n h_{3i}(Q_i + h_{1i}P_{\text{pub}}). \quad (2)$$

If it holds, the AS accepts and stores messages $\{m_i\}_{i=1}^n$ since σ_{agg} is valid.

4. Cryptanalysis of Thumbur et al.'s Scheme

Thumbur et al. [18] claimed that their CLAS-based authentication scheme for VANETs is capable of withstanding

forgery attacks and achieves various security requirements. Nevertheless, in this section, we provide two attack methods to demonstrate that their scheme is vulnerable to public key replacement attacks and coalition attacks.

4.1. Public Key Replacement Attack. Assume that V_i is the target vehicle selected by a Type I adversary \mathcal{A}_1 . \mathcal{A}_1 obtains V_i 's pseudoidentity PID_i and public key $vpk_i = (Q_i, R_i)$ and then uses the following attack steps to forge a valid single signature of a message m_i^* :

- (1) Calculate $h_{1i}^* = H_1(PID_i, R_i, P_{pub})$.
- (2) Pick $z_i \in Z_p^*$ randomly and calculate $Q_i^* = z_i P - h_{1i}^* P_{pub}$.
- (3) Replace the new public key of the target vehicle as $vpk_i^* = (Q_i^*, R_i)$.
- (4) Pick $u_i^* \in Z_p^*$ randomly and calculate $U_i^* = u_i^* P$.
- (5) Select a desired message m_i^* and calculate $h_{3i}^* = H_3(PID_i, m_i^*, vpk_i^*, U_i^*, T_i^*)$, where T_i^* is the current timestamp.
- (6) Calculate $s_i^* = (u_i^* + h_{3i}^* z_i) \bmod p$.
- (7) Set $\sigma_i^* = (U_i^*, s_i^*)$ as a forged single signature on m_i^* .

Obviously, σ_i^* is a valid single signature on m_i^* at T_i^* under (PID_i, vpk_i^*) , since

$$\begin{aligned} s_i^* P &= (u_i^* + h_{3i}^* z_i) P \\ &= u_i^* P + h_{3i}^* (z_i P) \\ &= U_i^* + h_{3i}^* (Q_i^* + h_{1i}^* P_{pub}). \end{aligned} \quad (3)$$

In the above attack, \mathcal{A}_1 only replaces the public key of the target vehicle, but the partial private key of the target vehicle is unknown to \mathcal{A}_1 . Therefore, the CLAS-based authentication scheme for VANETs of Thumbur et al. [18] cannot resist public key replacement attacks. The reason why \mathcal{A}_1 can attack successfully is that P_{pub} is offset by Q_i in $s_i P = U_i + h_{3i} (Q_i + h_{1i} P_{pub})$.

4.2. Coalition Attack. Two or more malicious vehicles can produce a valid aggregate signature by providing some false single signatures. Without loss of generality, suppose that V_1 and V_2 are two malicious vehicles. The coalition attack launched by V_1 and V_2 is as follows:

- (1) V_1 arbitrarily selects a message m_1^* and creates a single signature $\sigma_1 = (U_1, s_1)$ on m_1^* by invoking the *Signature Generation* algorithm in Thumbur et al.'s scheme [18].
- (2) V_2 arbitrarily selects a message m_2^* and creates a single signature $\sigma_2 = (U_2, s_2)$ on m_2^* by invoking the *Signature Generation* algorithm in Thumbur et al.'s scheme [18].
- (3) V_1 and V_2 exchange s_1 and s_2 in $\sigma_1 = (U_1, s_1)$ and $\sigma_2 = (U_2, s_2)$ to obtain two invalid single signatures $\sigma_1^* = (U_1, s_2)$ and $\sigma_2^* = (U_2, s_1)$.

- (4) After receiving $\{m_1^*, \sigma_1^* = (U_1, s_2), T_1^*, vpk_1, PID_1\}$ and $\{m_2^*, \sigma_2^* = (U_2, s_1), T_2^*, vpk_2, PID_2\}$ sent by V_1 and V_2 , respectively, the RSU invokes the *Aggregate Signature Generation* algorithm in Thumbur et al.'s scheme [18] to output an aggregate signature $\sigma_{agg}^* = (U_1, U_2, s^*)$ on $\{m_1^*, m_2^*\}$, where $s^* = s_2 + s_1$.

Clearly, σ_i^* is an invalid single signature on m_i^* at timestamp T_i^* under the pseudonym PID_i and the public key $vpk_i = (Q_i, R_i)$, where $i = 1, 2$. However, σ_{agg}^* is a valid aggregate signature for $\{m_1^*, m_2^*\}$ under $\{(vpk_i = (Q_i, R_i), PID_i)\}_{i=1}^2$, since

$$\begin{aligned} s^* P &= (s_2 + s_1) P \\ &= (s_1 + s_2) P \\ &= U_1 + h_{31} (Q_1 + h_{11} P_{pub}) + U_2 + h_{32} (Q_2 + h_{12} P_{pub}) \\ &= \sum_{i=1}^2 U_i + \sum_{i=1}^2 h_{3i} (Q_i + h_{1i} P_{pub}). \end{aligned} \quad (4)$$

Therefore, two incorrect single signatures $\{\sigma_1^*, \sigma_2^*\}$ jointly constructed by V_1 and V_2 can be used to produce a valid aggregate signature σ_{agg}^* . It is concluded that Thumbur et al.'s scheme [18] cannot resist coalition attacks from malicious vehicles. Malicious vehicles can successfully launch coalition attacks because there is no restriction on the order exchange of s_i in $s = \sum_{i=1}^n s_i$. If a CLAS-based authentication scheme for VANETs is vulnerable to coalition attacks, the AS is unable to determine whether a few incorrect single signatures produce a valid aggregate signature, and TA cannot track down the vehicle that sent a contentious message.

5. Improved Authentication Scheme for VANETs

To enhance the security of the CLAS-based authentication scheme for VANETs proposed by Thumbur et al. [18], we present an improved scheme on the basis of the original scheme and analyze its security and performance.

5.1. Our Improved Scheme. We mainly revise the *Signature Generation* method and the definition of hash function H_2 in Thumbur et al.'s scheme [18] to resist the forgery attacks described in Section 4. Our improved scheme is described as follows:

- (1) The algorithms *System Setup*, *Pseudonym Identity Generation*, *Partial Private Key Generation*, and *Set Secret Value* are the same as those in the scheme in [18]. Note that a hash function $H_4: \{0, 1\}^* \rightarrow Z_p^*$ is added to the *System Setup* algorithm in the revised scheme.
- (2) *Vehicle Key Generation.* A vehicle V_i sets its private key $vsk_i = (psk_i, x_i)$ and the corresponding public key $vpk_i = (X_i, R_i) = (x_i P, r_i P)$. Similarly, the AS sets its private key $vsk_{AS} = (psk_{AS}, x_{AS})$ and the corresponding public key $vpk_{AS} = (X_{AS}, R_{AS}) = (x_{AS} P, r_{AS} P)$.

(3) *Signature Generation.* Vehicle V_i with PID_i utilizes its private key $vsk_i = (psk_i, x_i)$ to sign a traffic-related message m_i as follows:

- (a) Pick $u_i \in Z_p^*$ at random and calculate $U_i = u_i P$.
- (b) Calculate $h_{2i} = H_2(PID_i, m_i, P_{pub}, U_i, T_i)$ and $h_{3i} = H_3(PID_i, m_i, vpk_i, U_i, T_i)$, where T_i is the current timestamp.
- (c) Calculate $s_i = (u_i + h_{2i}psk_i + h_{3i}x_i) \bmod p$.
- (d) Output a single signature $\sigma_i = (U_i, s_i)$ on m_i .
- (e) Send $\{m_i, \sigma_i, T_i, vpk_i, PID_i\}$ to the nearby RSU.

(4) *Signature Verification.* Given a single signature $\sigma_i = (U_i, s_i)$ on a message m_i at timestamp T_i under a pseudonym PID_i and a public key $vpk_i = (X_i, R_i)$, the verifier performs the following verification operations if ΔT_i and T_i are both valid:

- (a) Calculate $h_{1i} = H_1(PID_i, R_i, P_{pub})$.
- (b) Calculate $h_{2i} = H_2(PID_i, m_i, P_{pub}, U_i, T_i)$ and $h_{3i} = H_3(PID_i, m_i, vpk_i, U_i, T_i)$.
- (c) Check the single signature verification equation below:

$$s_i P = U_i + h_{2i}(R_i + h_{1i}P_{pub}) + h_{3i}X_i. \quad (5)$$

The verifier accepts σ_i if it holds; otherwise, σ_i is considered invalid.

correctness,

$$\begin{aligned} s_i P &= (u_i + h_{2i}psk_i + h_{3i}x_i)P \\ &= u_i P + h_{2i}(psk_i P) + h_{3i}(x_i P) \\ &= U_i + h_{2i}(R_i + h_{1i}P_{pub}) + h_{3i}X_i. \end{aligned} \quad (6)$$

(5) *Aggregate Signature Generation.* After receiving n single signatures $\{\sigma_i = (U_i, s_i)\}_{i=1}^n$ on messages $\{m_i\}_{i=1}^n$ from n vehicles, the RSU utilizes AS's public key $vpk_{AS} = (X_{AS}, R_{AS})$ to aggregate single signatures.

- (a) Calculate $s = H_4(s_1 X_{AS}, \dots, s_n X_{AS})$.
- (b) Set $\sigma_{agg} = (U_1, \dots, U_n, s)$ as the aggregate signature on $\{m_i\}_{i=1}^n$.

(6) *Aggregate Signature Verification.* Given an aggregate signature $\sigma_{agg} = (U_1, \dots, U_n, s)$ on messages $\{m_i\}_{i=1}^n$ under n tuples $\{(vpk_i = (X_i, R_i), PID_i)\}_{i=1}^n$, the AS utilizes its secret value x_{AS} to perform the following aggregate signature verification operations if ΔT_i and T_i in each message m_i are in the period of validity:

- (a) Calculate $h_{1i} = H_1(PID_i, R_i, P_{pub})$ for all $i \in \{1, \dots, n\}$.
- (b) Calculate $h_{2i} = H_2(PID_i, m_i, P_{pub}, U_i, T_i)$ and $h_{3i} = H_3(PID_i, m_i, vpk_i, U_i, T_i)$ for all $i \in \{1, \dots, n\}$.
- (c) Check the aggregation signature verification equation below:

$$\begin{aligned} s &= H_4(x_{AS}(U_1 + h_{21}(R_1 + h_{11}P_{pub}) + h_{31}X_1), \dots, \\ & \quad x_{AS}(U_n + h_{2n}(R_n + h_{1n}P_{pub}) + h_{3n}X_n)). \end{aligned} \quad (7)$$

If it holds, AS accepts and stores these messages $\{m_i\}_{i=1}^n$ since σ_{agg} is valid.

5.2. Security Analysis. Similar to Theorems 1 and 2 in [18], we use the following Theorems 1 and 2 to show that our enhanced scheme is secure for Type I and II attackers in the random oracle model. In addition, Theorem 3 proves that the aggregate signature is secure in the enhanced scheme as long as the individual signatures participating in the aggregation are secure.

Theorem 1. *In the random oracle model, our improved scheme is existentially unforgeable against Type I attackers under the intractability of the ECDLP.*

Proof. Assuming that a Type I attacker \mathcal{A}_1 can forge a legitimate signature with probability ϵ_1 , there exists an algorithm \mathcal{C}_1 that successfully solves the ECDLP using the forged signature of \mathcal{A}_1 . Given a random ECDLP instance $(P, Q = yP)$, where $y \in Z_p^*$ is unknown to \mathcal{C}_1 , in order to calculate y , \mathcal{C}_1 plays the following interactive game with \mathcal{A}_1 :

- (1) *Initialization Phase.* \mathcal{C}_1 chooses PID^* as the challenged pseudonym identity of the target user. Then, \mathcal{C}_1 picks $s \in Z_p^*$ at random as the master secret key of KGC and calculates $P_{pub} = sP$. After that, \mathcal{C}_1 runs the *System Setup* algorithm to produce parameters $\text{params} = \{p, G, P, T_{pub}, P_{pub}, H_0, H_1, H_2, H_3, H_4\}$ and sends params to \mathcal{A}_1 .
- (2) *Queries Phase.* \mathcal{C}_1 creates six initially empty lists L_0, L_1, L_2, L_3, L_4 , and L_{CU} . \mathcal{A}_1 adaptively initiates a series of queries, and \mathcal{C}_1 responds to them in the following ways:

- (1) *H_0 -Queries.* When \mathcal{A}_1 initiates an inquiry about $H_0(bPID_{i,1}, \Delta T_i)$, \mathcal{C}_1 submits h_{0i} to \mathcal{A}_1 if there exists the tuple $(bPID_{i,1}, \Delta T_i, h_{0i})$ in list L_0 . Otherwise, \mathcal{C}_1 picks $h_{0i} \in Z_p^*$ at random, assigns $h_{0i} = H_0(bPID_{i,1}, \Delta T_i)$, transmits h_{0i} to \mathcal{A}_1 , and stores $(bPID_{i,1}, \Delta T_i, h_{0i})$ in L_0 .
- (2) *H_1 -Queries.* When \mathcal{A}_1 initiates an inquiry about $H_1(PID_i, R_i, P_{pub})$, \mathcal{C}_1 submits h_{1i} to \mathcal{A}_1 if there exists the tuple $(PID_i, R_i, P_{pub}, h_{1i})$ in list L_1 . Otherwise, \mathcal{C}_1 picks $h_{1i} \in Z_p^*$ at random, assigns $h_{1i} = H_1(PID_i, R_i, P_{pub})$, transmits h_{1i} to \mathcal{A}_1 , and stores $(PID_i, R_i, P_{pub}, h_{1i})$ in L_1 .
- (3) *H_2 -Queries.* When \mathcal{A}_1 initiates an inquiry about $H_2(PID_i, m_i, P_{pub}, U_i, T_i)$, \mathcal{C}_1 submits h_{2i} to \mathcal{A}_1 if there exists the tuple $(PID_i, m_i, P_{pub}, U_i, T_i, h_{2i})$ in list L_2 . Otherwise, \mathcal{C}_1 picks $h_{2i} \in Z_p^*$ at random, assigns $h_{2i} = H_2(PID_i, m_i, P_{pub}, U_i, T_i)$, transmits h_{2i} to

- \mathcal{A}_1 , and stores the tuple $(PID_i, m_i, P_{pub}, U_i, T_i, h_{2i})$ in L_2 .
- (4) *H₃-Queries*. When \mathcal{A}_1 initiates an inquiry about $H_3(PID_i, m_i, vpk_i, U_i, T_i)$, \mathcal{E}_1 submits h_{3i} to \mathcal{A}_1 if there exists the tuple $(PID_i, m_i, vpk_i, U_i, T_i, h_{3i})$ in list L_3 . Otherwise, \mathcal{E}_1 picks $h_{3i} \in Z_p^*$ at random, assigns $h_{3i} = H_3(PID_i, m_i, vpk_i, U_i, T_i)$, transmits h_{3i} to \mathcal{A}_1 , and stores the tuple $(PID_i, m_i, vpk_i, U_i, T_i, h_{3i})$ in L_3 .
- (5) *H₄-Queries*. When \mathcal{A}_1 initiates an inquiry about $H_4(s_{1i}X_{AS}, \dots, s_{ni}X_{AS})$, \mathcal{E}_1 submits h_{4i} to \mathcal{A}_1 if there exists the tuple $(s_{1i}X_{AS}, \dots, s_{ni}X_{AS}, h_{4i})$ in list L_4 . Otherwise, \mathcal{E}_1 picks $h_{4i} \in Z_p^*$ at random, assigns $h_{4i} = H_4(s_{1i}X_{AS}, \dots, s_{ni}X_{AS})$, transmits h_{4i} to \mathcal{A}_1 , and stores $(s_{1i}X_{AS}, \dots, s_{ni}X_{AS}, h_{4i})$ in L_4 .
- (6) *Create-User-Queries*. When \mathcal{A}_1 initiates an inquiry about PID_i , \mathcal{E}_1 submits $vpk_i = (Q_i, R_i)$ to \mathcal{A}_1 if there exists the tuple $(PID_i, r_i, R_i, psk_i, x_i, X_i)$ in list L_{CU} . Otherwise, \mathcal{E}_1 executes as follows:
- (a) If $PID_i = PID^*$, \mathcal{E}_1 randomly picks $x^*, h_1^* \in Z_p^*$ and sets $R^* = Q$, $X^* = x^*P$, $r^* = \perp$, $psk^* = \perp$, and $h_1^* = H_1(PID^*, R^*, P_{pub})$. Then, \mathcal{E}_1 transmits $vpk^* = (X^*, R^*)$ to \mathcal{A}_1 and stores $(PID^*, R^*, P_{pub}, h_1^*)$ in L_1 and $(PID^*, \perp, R^*, \perp, x^*, X^*)$ in L_{CU} .
- (b) If $PID_i \neq PID^*$, \mathcal{E}_1 randomly picks $x_i, r_i, h_{1i} \in Z_p^*$, calculates $R_i = r_iP$, $X_i = x_iP$, and $psk_i = (r_i + sh_{1i}) \bmod p$, and sets $h_{1i} = H_1(PID_i, R_i, P_{pub})$. Then, \mathcal{E}_1 transmits $vpk_i = (X_i, R_i)$ to \mathcal{A}_1 and stores $(PID_i, R_i, P_{pub}, h_{1i})$ in L_1 and $(PID_i, r_i, R_i, psk_i, x_i, X_i)$ in L_{CU} .
- (7) *Partial-Private-Key-Queries*. When \mathcal{A}_1 submits an inquiry about PID_i , \mathcal{E}_1 aborts if $PID_i \neq PID^*$. Otherwise, \mathcal{E}_1 searches the tuple $(PID_i, r_i, R_i, psk_i, x_i, X_i)$ in L_{CU} and transmits psk_i to \mathcal{A}_1 .
- (8) *Set-Secret-Value-Queries*. When \mathcal{A}_1 submits an inquiry about PID_i , \mathcal{E}_1 searches the tuple $(PID_i, r_i, R_i, psk_i, x_i, X_i)$ in L_{CU} and transmits x_i to \mathcal{A}_1 .
- (9) *Replace-Public-Key-Queries*. When \mathcal{A}_1 submits an inquiry about (PID_i, R'_i, X'_i) , \mathcal{E}_1 finds the tuple $(PID_i, r_i, R_i, psk_i, x_i, X_i)$ in L_{CU} and replaces (R_i, X_i) with (R'_i, X'_i) .
- (10) *Sign-Queries*. When \mathcal{A}_1 submits an inquiry about (PID_i, m_i) , \mathcal{E}_1 does the following:
- (a) If $PID_i \neq PID^*$, \mathcal{E}_1 runs the *Signature Generation* algorithm to output a single signature $\sigma_i = (U_i, s_i)$ on m_i and transmits σ_i to \mathcal{A}_1 .
- (b) If $PID_i = PID^*$, \mathcal{E}_1 picks $s_i, h_{2i}, h_{3i} \in Z_p^*$ at random and the current timestamp T_i . Then, \mathcal{E}_1 recovers $(PID^*, R^*, P_{pub}, h_1^*)$ and $(PID^*, \perp, R^*, \perp, x^*, X^*)$ from L_1 and L_{CU} , respectively. After that, \mathcal{E}_1 sets $U_i = s_iP - h_{2i}(R^* + h_1^*P_{pub}) - h_{3i}X^*$, transmits

$\sigma_i = (U_i, s_i)$ to \mathcal{A}_1 , and stores $(PID^*, m_i, P_{pub}, U_i, T_i, h_{2i})$ in L_2 and $(PID^*, m_i, vpk^*, U_i, T_i, h_{3i})$ in L_3 .

- (3) *Forgery Phase*. After initiating a limited number of the above-mentioned inquiries, \mathcal{A}_1 creates a valid single signature $\sigma^* = (U^*, s^*)$ on a message m^* under PID_i and $vpk_i = (X_i, R_i)$. If $PID_i \neq PID^*$, \mathcal{E}_1 aborts. Otherwise, according to the Forking lemma [20], \mathcal{E}_1 can obtain another valid signature $\sigma' = (U^*, s')$ with the same random tape and a different value h'_{2i} assigned by H_2 . Since $\sigma^* = (U^*, s^*)$ and $\sigma' = (U^*, s')$ are valid, the following equations hold:

$$\begin{aligned} s^*P &= U^* + h_{2i}^*(R^* + h_1^*P_{pub}) + h_{3i}^*X^*, \\ s'P &= U^* + h'_{2i}(R^* + h_1^*P_{pub}) + h_{3i}^*X^*. \end{aligned} \quad (8)$$

Since $R^* = Q = yP$, $P_{pub} = sP$, and $X^* = x^*P$, the following equations can be easily derived from the above two equations:

$$\begin{aligned} s^*P &= U^* + h_{2i}^*(yP + h_1^*sP) + h_{3i}^*X^*, \\ s'P &= U^* + h'_{2i}(yP + h_1^*sP) + h_{3i}^*X^*. \end{aligned} \quad (9)$$

Then, we have $(s^* - s')P = (h_{2i}^* - h'_{2i})yP + (h_{2i}^* - h'_{2i})h_1^*sP$. Hence, \mathcal{E}_1 calculates the solution of the given ECDLP instance as $y = (h_{2i}^* - h'_{2i})^{-1}[(s^* - s') - (h_{2i}^* - h'_{2i})h_1^*s] \bmod p$.

Let q_{ppk} denote the total number of partial private key queries initiated by \mathcal{A}_1 in the whole game. In *Partial-Private-Key-Queries*, the probability of \mathcal{E}_1 completing the game is at least $(1 - 1/q_{ppk} + 1)^{q_{ppk}}$. In *Forgery Phase*, the probability that \mathcal{E}_1 does not terminate the game is at least $1/q_{ppk} + 1$. If \mathcal{A}_1 can forge a legitimate signature with probability ϵ_1 , then \mathcal{E}_1 successfully solves the ECDLP with probability $(1 - 1/q_{ppk} + 1)^{q_{ppk}}(1/q_{ppk} + 1)\epsilon_1$. However, the ECDLP is intractable in polynomial time; hence, our improved scheme is secure against Type I attacks. \blacksquare

Theorem 2. *In the random oracle model, our enhanced scheme is existentially unforgeable against Type II attackers under the intractability of the ECDLP.*

Proof. Assuming that a Type II attacker \mathcal{A}_2 can forge a legitimate signature with probability ϵ_2 , there exists an algorithm \mathcal{C}_2 that successfully solves the ECDLP by using the forged signature of \mathcal{A}_2 . Given a random ECDLP instance $(P, Q = yP)$, where $y \in Z_p^*$ is unknown to \mathcal{C}_2 , in order to calculate y , \mathcal{C}_2 plays the following interactive game with \mathcal{A}_2 .

- (1) *Initialization Phase*. \mathcal{A}_2 chooses $s \in Z_p^*$ at random as KGC's master secret key and calculates $P_{pub} = sP$. Then, \mathcal{A}_2 runs the *System Setup* algorithm to output parameters $\text{params} = \{p, G, P, T_{pub}, P_{pub}, H_0, H_1, H_2, H_3, H_4\}$. Following that, \mathcal{A}_2 returns params and s to \mathcal{C}_2 .
- (2) *Queries Phase*. \mathcal{C}_2 chooses PID^* as the challenged pseudonym identity of the target user. Similar to

Theorem 1, \mathcal{E}_2 creates six initially empty lists L_0, L_1, L_2, L_3, L_4 , and L_{CU} . \mathcal{A}_2 adaptively initiates a series of queries, and \mathcal{E}_2 responds to them in the following ways:

- (1) H_0 -Queries, H_1 -Queries, H_2 -Queries, H_3 -Queries, and H_4 -Queries are the same as Theorem 1.
- (2) *Create-User-Queries*. When \mathcal{A}_2 initiates an inquiry about PID_i , \mathcal{E}_2 submits $vpk_i = (Q_i, R_i)$ to \mathcal{A}_2 if there exists the tuple $(PID_i, r_i, R_i, psk_i, x_i, X_i)$ in list L_{CU} . Otherwise, \mathcal{E}_2 executes as follows:
 - (a) If $PID_i = PID^*$, \mathcal{E}_2 picks $r^*, h_1^* \in Z_p^*$ at random and sets $X^* = Q$, $R^* = r^*P$, $psk^* = (r^* + sh_1^*) \bmod p$, $x^* = \perp$, and $h_1^* = H_1(PID^*, R^*, P_{pub})$. Then, \mathcal{E}_2 transmits $vpk^* = (X^*, R^*)$ to \mathcal{A}_2 and stores $(PID^*, R^*, P_{pub}, h_1^*)$ in L_1 and $(PID^*, r^*, R^*, psk^*, \perp, X^*)$ in L_{CU} .
 - (b) If $PID_i \neq PID^*$, \mathcal{E}_2 randomly picks $x_i, r_i, h_{1i} \in Z_p^*$, calculates $R_i = r_iP$, $X_i = x_iP$, and $psk_i = (r_i + sh_{1i}) \bmod p$, and sets $h_{1i} = H_1(PID_i, R_i, P_{pub})$. Then, \mathcal{E}_2 transmits $vpk_i = (X_i, R_i)$ to \mathcal{A}_2 and stores $(PID_i, R_i, P_{pub}, h_{1i})$ in L_1 and $(PID_i, r_i, R_i, psk_i, x_i, X_i)$ in L_{CU} .
- (3) *Partial-Private-Key-Queries*. When \mathcal{A}_2 submits an inquiry about PID_i , \mathcal{E}_2 looks for the tuple $(PID_i, r_i, R_i, psk_i, x_i, X_i)$ in L_{CU} and transmits psk_i to \mathcal{A}_2 .
- (4) *Set-Secret-Value-Queries*. When \mathcal{A}_2 submits an inquiry about PID_i , \mathcal{E}_2 aborts if $PID_i \neq PID^*$. Otherwise, \mathcal{E}_2 finds the tuple $(PID_i, r_i, R_i, psk_i, x_i, X_i)$ in L_{CU} and transmits x_i to \mathcal{A}_2 .
- (5) *Sign-Queries* is the same as Theorem 1.
- (3) *Forgery Phase*. After initiating a limited number of the above-mentioned inquiries, \mathcal{A}_2 generates a valid single signature $\sigma^* = (U^*, s^*)$ on a message m^* under PID_i and $vpk_i = (X_i, R_i)$. If $PID_i \neq PID^*$, \mathcal{E}_2 aborts. Otherwise, according to the Forking lemma [20], \mathcal{E}_2 can obtain another valid signature $\sigma' = (U^*, s')$ with the same random tape and a different value h'_{3i} assigned by H_3 . Since $\sigma^* = (U^*, s^*)$ and $\sigma' = (U^*, s')$ are valid, the following equations hold:

$$\begin{aligned} s^*P &= U^* + h_{2i}^*(R^* + h_{1i}^*P_{pub}) + h_{3i}^*X^*, \\ s'P &= U^* + h_{2i}^*(R^* + h_{1i}^*P_{pub}) + h_{3i}^*X^*. \end{aligned} \quad (10)$$

Since $X^* = Q = yP$, $P_{pub} = sP$, and $R^* = r^*P$, the following equations can be easily derived from the above two equations:

$$\begin{aligned} s^*P &= U^* + h_{2i}^*(r^*P + h_{1i}^*sP) + h_{3i}^*yP, \\ s'P &= U^* + h_{2i}^*(r^*P + h_{1i}^*sP) + h_{3i}^*yP. \end{aligned} \quad (11)$$

Then, we have $(s^* - s')P = (h_{3i}^* - h_{3i}')yP$. Hence, \mathcal{E}_2 calculates the solution of the given ECDLP instance as $y = (h_{3i}^* - h_{3i}')^{-1}(s^* - s') \bmod p$.

Let q_{ssv} denote the total number of secret value queries initiated by \mathcal{A}_2 in the whole game. In *Set-Secret-Value-Queries*, the probability that \mathcal{E}_2 completes the game is at least $(1 - 1/q_{ssv} + 1)^{q_{ssv}}$. In *Forgery Phase*, the probability that \mathcal{E}_2 does not terminate the game is at least $1/q_{ssv} + 1$. If \mathcal{A}_2 can forge a legitimate signature with probability ϵ_2 , then \mathcal{E}_2 successfully solves the ECDLP with probability $(1 - 1/q_{ssv} + 1)^{q_{ssv}}(1/q_{ssv} + 1)\epsilon_2$. However, the ECDLP is intractable in polynomial time; hence, our enhanced scheme is resistant to Type II attacks. ■ \square

Theorem 3. *Our improved scheme can withstand the coalition attack from malicious vehicles if H_4 is a collision-resistant hash function.*

Proof. If $\sigma_{agg} = (U_1, \dots, U_n, s)$ is a valid aggregate signature, then σ_{agg} satisfies the following aggregate signature verification equation:

$$\begin{aligned} s &= H_4(x_{AS}(U_1 + h_{21}(R_1 + h_{11}P_{pub}) + h_{31}X_1), \dots, \\ & x_{AS}(U_n + h_{2n}(R_n + h_{1n}P_{pub}) + h_{3n}X_n)). \end{aligned} \quad (12)$$

From the *Aggregate Signature Generation* algorithm in the improved scheme, it is easy to get

$$s = H_4(s_1X_{AS}, \dots, s_nX_{AS}). \quad (13)$$

According to the collision resistance of H_4 , we can obtain

$$s_iX_{AS} = x_{AS}(U_i + h_{2i}(R_i + h_{1i}P_{pub}) + h_{3i}X_i), \quad i \in \{1, \dots, n\}. \quad (14)$$

Since $X_{AS} = x_{AS}P$, we have

$$x_{AS}(s_iP) = x_{AS}(U_i + h_{2i}(R_i + h_{1i}P_{pub}) + h_{3i}X_i), \quad i \in \{1, \dots, n\}. \quad (15)$$

Hence, we can obtain

$$s_iP = U_i + h_{2i}(R_i + h_{1i}P_{pub}) + h_{3i}X_i, \quad i \in \{1, \dots, n\} \quad (16)$$

This shows that each single signature $\sigma_i = (U_i, s_i)$ used to generate σ_{agg} is valid.

On the other hand, if $\sigma_i = (U_i, s_i)$ is a valid single signature, then σ_i must satisfy the following single signature verification equation:

$$s_iP = U_i + h_{2i}(R_i + h_{1i}P_{pub}) + h_{3i}X_i, \quad i \in \{1, \dots, n\}. \quad (17)$$

Then, we can obtain

$$x_{AS}(s_iP) = x_{AS}(U_i + h_{2i}(R_i + h_{1i}P_{pub}) + h_{3i}X_i), \quad i \in \{1, \dots, n\}. \quad (18)$$

From the above equations, we can easily derive

$$s_iX_{AS} = x_{AS}(U_i + h_{2i}(R_i + h_{1i}P_{pub}) + h_{3i}X_i), \quad i \in \{1, \dots, n\}. \quad (19)$$

Hence, we have

$$\begin{aligned} s &= H_4(s_1 X_{AS}, \dots, s_n X_{AS}) \\ &= H_4(x_{AS}(U_1 + h_{21}(R_1 + h_{11}P_{\text{pub}}) + h_{31}X_1), \dots, \\ &\quad x_{AS}(U_n + h_{2n}(R_n + h_{1n}P_{\text{pub}}) + h_{3n}X_n)). \end{aligned} \quad (20)$$

This shows that the aggregate signature $\sigma_{agg} = (U_1, \dots, U_n, s)$ constructed with σ_i is valid.

In our improved scheme, it can be seen from the above analysis that an aggregate signature is valid if and only if every single signature participating in the aggregation is valid. Therefore, our enhanced scheme is secure against coalition attacks from malicious vehicles. ■

We show that the improved scheme satisfies the following security requirements in VANETs:

- (1) *Resistance to Forgery Attacks.* In the *Signature Generation* algorithm of our improved scheme, the master public key P_{pub} of KGC is an input of $h_{2i} = H_2(\text{PID}_i, m_i, P_{\text{pub}}, U_i, T_i)$, and the public key $vpk_i = (X_i, R_i)$ of the vehicle is an input of $h_{3i} = H_3(\text{PID}_i, m_i, vpk_i, U_i, T_i)$. Furthermore, in $s_i = (u_i + h_{2i}psk_i + h_{3i}x_i) \bmod p$, h_{2i} is bound to the partial private key psk_i , and h_{3i} is bound to the secret value x_i . If either P_{pub} or vpk_i is changed in the *Signature Verification* algorithm, the single signature verification equation $s_i P = U_i + h_{2i}(R_i + h_{1i}P_{\text{pub}}) + h_{3i}X_i$ will fail. Hence, it is not feasible for an attacker to forge a single signature by bypassing the secret value or the partial private key of the vehicle. As a result, our improved scheme can withstand the public key replacement attack described in Section 4 as well as the forgery attack from a malicious KGC. More importantly, in the *Aggregate Signature Generation* algorithm, the collision resistance of the hash function $H_4(s_1 X_{AS}, \dots, s_n X_{AS})$ ensures that the exchange or illegal modification of any single signature value cannot pass the aggregate signature verification equation in the *Aggregate Signature Verification* algorithm. Therefore, our scheme is resistant to the coalition attack given in Section 4.
- (2) *Unlinkability.* In our improved scheme, vehicle V_i with the pseudoidentity PID_i transmits message m_i and the corresponding signature $\sigma_i = (U_i, s_i)$ to the RSU, where $U_i = u_i P$ and $s_i = (u_i + h_{2i}psk_i + h_{3i}x_i) \bmod p$. Furthermore, value u_i used in the signature σ_i is random, so the attacker is unable to associate two different messages transmitted by the same vehicle. That is, our improved scheme achieves the unlinkability of messages.
- (3) *Replay Attack.* Because of the openness of wireless networks in VANETs, the signature $\sigma_i = (U_i, s_i)$ transmitted by the vehicle to the RSU can be easily obtained by an attacker. Timestamp T_i is embedded in $h_{2i} = H_2(\text{PID}_i, m_i, P_{\text{pub}}, U_i, T_i)$ and $h_{3i} = H_3(\text{PID}_i, m_i, vpk_i, U_i, T_i)$, and $s_i = (u_i + h_{2i}psk_i + h_{3i}x_i) \bmod p$ is a component of

TABLE 2: Running time of a single cryptographic operation.

Notation	Cryptographic operation	Execution time (ms)
T_P	Bilinear pairing	4.441043
T_{PM}	Bilinear pairing scalar multiplication	4.87256
T_{sm}	ECC scalar multiplication	0.165217
T_{add}	ECC point addition	0.001404
T_H	Map to point hash function	0.142682

signature σ_i . The RSU uses T_i to check the freshness of message m_i and then verifies the validity of signature σ_i . Hence, our revised scheme is resistant to replay attacks.

- (4) *Message Authenticity, Nonrepudiation, and Integrity.* Compared with the scheme of Thumbur et al. [18], the algorithms *Pseudonym Identity Generation*, *Partial Private Key Generation*, and *Set Secret Value* in our improved scheme are the same as those in the original scheme. Furthermore, we only modify $h_{2i} = H_2(\text{PID}_i, m_i, P_{\text{pub}}, U_i, T_i)$ in the enhanced scheme, and the remainder of the *Signature Generation* algorithm is the same as that of the original scheme. The existential unforgeability of our improved scheme is guaranteed by Theorems 1 and 2. Furthermore, the identity information PID_i of the vehicle is embedded in the signature, so that the vehicle cannot deny any previous messages. Therefore, the improved scheme can guarantee authenticity, non-repudiation, and integrity of messages among the vehicles.
- (5) *Anonymity and Traceability.* To achieve the anonymity of the vehicle during communication, the vehicle uses a pseudonym $\text{PID}_i = \{\text{PID}_{i,1}, \text{PID}_{i,2}, \Delta T_i\}$ to communicate with RSU or other vehicles. When a controversial traffic-related message occurs, TA uses its master key b and the vehicle's pseudonym PID_i to obtain the real identity $\text{RID}_i = b\text{PID}_{i,1} \oplus K_i$. Thus, our improved scheme provides anonymity and traceability of the vehicle. It is generally assumed that TA is trustworthy. To further limit the rights of TA, a smart contract can be introduced to store the vehicle's pseudoidentity PID_i and relevant information on the blockchain during the vehicle's registration and traceability [21]. The data on the blockchain cannot be tampered with, so it can effectively prevent TA from abusing its rights. □

5.3. Efficiency Analysis. We evaluate the computational efficiency of our enhanced scheme by adopting the experimental data in [22]. Table 2 provides the measured value of the running time of every cryptographic operation. To evaluate the performance of the scheme based on bilinear pairing, the super singular elliptic curve $E: y^2 = x^3 + x \bmod p_1$ was picked, where the length of prime p_1 is 521 bits and the length of an element in G_1 is 1024 bits.

TABLE 3: Comparison of computational overhead.

Scheme	Single signature generation (ms)	Single signature verification (ms)	Aggregate signature verification
Kumar et al. [4]	$4T_{sm} + 2T_{add} + T_H \approx 0.806358$	$3T_{sm} + 2T_H + 4T_P + 2T_{PM} \approx 28.290307$	$3nT_{sm} + (3n-1)T_{add} + 4T_P + 2T_{PM}$
Mei et al. [13]	$4T_{sm} + 2T_{add} + 2T_H \approx 0.94904$	$2T_{sm} + 2T_H + 4T_P + 2T_{PM} \approx 28.12509$	$2nT_{sm} + (2n-2)T_{add} + 2T_H + 4T_P$
Xu et al. [14]	$3T_{sm} + T_{add} + T_H \approx 0.639737$	$2T_{sm} + T_{add} + 2T_H + 3T_P + T_{PM} \approx 18.812891$	$2nT_{sm} + (3n-2)T_{add} + (n+1)T_H$
Thumbur et al. [18]	$T_{sm} \approx 0.165217$	$3T_{sm} + 2T_{add} \approx 0.498459$	$+3T_P + T_{PM}$
Our scheme	$T_{sm} \approx 0.165217$	$4T_{sm} + 3T_{add} \approx 0.66508$	$(2n+1)T_{sm} + (3n-1)T_{add}$
			$4nT_{sm} + 3nT_{add}$

Note. n : the number of messages.

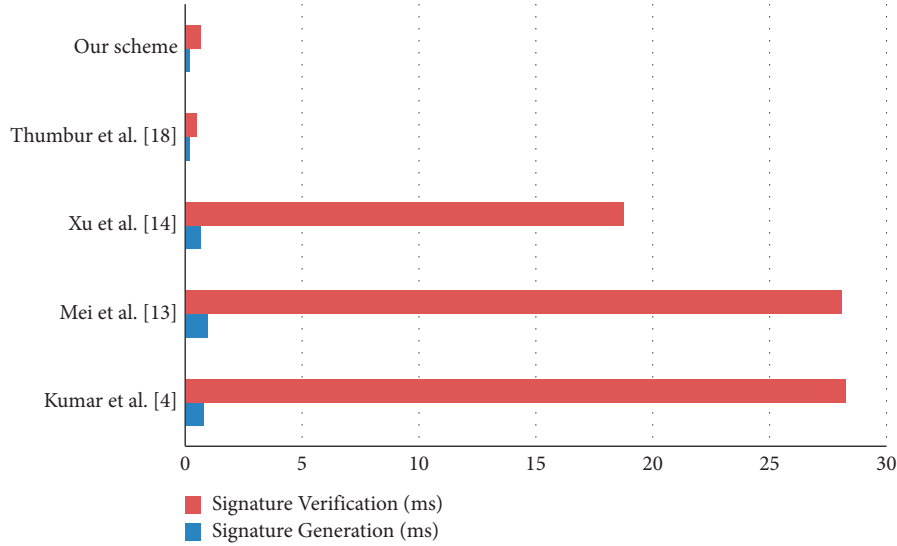


FIGURE 2: Comparison of the computational cost of a single signature.

TABLE 4: Comparison of communication overhead.

Scheme	Single signature length	Aggregate signature length	Coalition attack	Types I and II attacks
Kumar et al. [4]	$2 G_1 = 2048$ bits	$(n+1) G_1 = 1024(n+1)$ bits	No	Yes
Mei et al. [13]	$2 G_1 = 2048$ bits	$2 G_1 = 2048$ bits	No	Yes
Xu et al. [14]	$2 G_1 = 2048$ bits	$(n+1) G_1 = 1024(n+1)$ bits	No	Yes
Thumbur et al. [18]	$ G + Z_p^* = 480$ bits	$n G + Z_p^* = 320n + 160$ bits	No	No
Our scheme	$ G + Z_p^* = 480$ bits	$n G + Z_p^* = 320n + 160$ bits	Yes	Yes

Note. n : the number of messages.

In order to achieve the same security level, the Koblitz elliptic curve $E: y^2 = x^3 + ax + b \pmod{p}$ was chosen to evaluate the performance of the scheme without bilinear pairing, where the length of p is 160 bits and the length of an element in G is 320 bits.

We compare the computational overhead of the improved scheme with the other four CLAS-based authentication schemes in VAENTs [4, 13, 14, 18] in signing a message, verifying the signature of a message and verifying the aggregate signature of n messages. The comparison results are shown in Table 3. We do not consider the computational overhead of ordinary hash function, the modular addition operation, and the modular multiplication operation because their execution time is very short.

The time cost of our improved scheme and Thumbur et al.'s scheme [18] to generate a single signature is

0.168785 ms, which is the minimum time overhead among all schemes [4, 13, 14, 18]. In addition, the time cost of verifying a single signature and an aggregate signature in [18] is slightly lower than that in the proposed scheme. Nevertheless, in Section 4, we have shown that Thumbur et al.'s scheme [18] is insecure; hence, it cannot be practically applied to VANETS.

As demonstrated in Figure 2, in the generation and verification phase of a single signature, the computational cost of our scheme and Thumbur et al.'s scheme [18] is lower than those of the other three schemes [4, 13, 14]. Furthermore, our scheme requires somewhat more processing time to validate the legality of a single signature compared to Thumbur et al.'s scheme [18], but our scheme has the ability to resist all kinds of attacks given in Section 4.

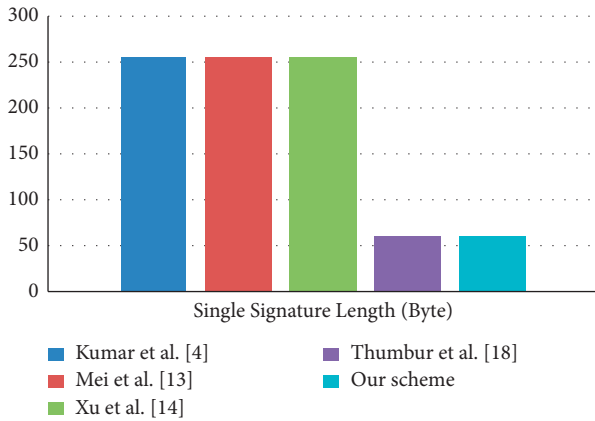


FIGURE 3: Comparison of single signature length.

Table 4 and Figure 3 provide the comparison results of the five schemes regarding communication costs. As can be seen from Table 4 and Figure 3, our improved scheme and Thumbur et al.'s scheme [18] have the shortest single signature length, so the vehicle has a lower communication overhead. However, with the exception of our improved scheme, the other four schemes [4, 13, 14, 18] are incapable of resisting the coalition attack described in Section 4.2. Hence, our improved scheme enhances the security while maintaining the communication performance of the original scheme.

6. Conclusions

In 2021, Thumbur et al. [18] presented an efficient CLAS-based authentication scheme for VANETS. In this article, we cryptanalyze their scheme and find that it is insecure against public key replacement attacks and coalition attacks. To improve the security of their scheme, we also present the corresponding improvement scheme. Furthermore, the analysis results demonstrate that the improved scheme meets a variety of security requirements. However, some security properties such as identity revocation and authentication [23–25] are not considered in the enhanced and original schemes. Hence, our future work is to design a CLAS-based authentication scheme that supports the revocation mechanism of the vehicle's identity.

Data Availability

The data used to support the findings of this study are available at <https://ieeexplore.ieee.org/document/9031715/>.

Conflicts of Interest

All authors have no conflicts of interest.

Acknowledgments

This research was supported by the China Postdoctoral Science Foundation (no. 2017M610817) and the Gansu Science and Technology Planning Project (no. 20CX9ZA076).

References

- [1] S. Mohamed Hatim, S. Jamel Elias, N. Awang, and M. Y. Darus, "VANETS and Internet of Things (IoT): a discussion," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 1, pp. 218–224, 2018.
- [2] H. Lu and J. Li, "Privacy-preserving authentication schemes for vehicular ad hoc networks: a survey," *Wireless Communications and Mobile Computing*, vol. 16, no. 6, pp. 643–655, 2016.
- [3] S. O. Ogundoyin, "An autonomous lightweight conditional privacy-preserving authentication scheme with provable security for vehicular ad-hoc networks," *International Journal of Computers and Applications*, vol. 42, no. 2, pp. 196–211, 2020.
- [4] P. Kumar, S. Kumari, V. Sharma, X. Li, A. K. Sangaiah, and S. H. Islam, "Secure CLS and CL-AS schemes designed for VANETs," *The Journal of Supercomputing*, vol. 75, no. 6, pp. 3076–3098, 2019.
- [5] F. Qu, Z. Wu, F. Wang, and W. Cho, "A security and privacy review of VANETs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 2985–2996, 2015.
- [6] A. B. S. Ahamed, N. Kanagaraj, and M. Azees, "EMBA: an efficient anonymous mutual and batch authentication schemes for vanets," in *Proceedings of the Second International Conference on Inventive Communication and Computational Technologies*, pp. 1320–1326, New Delhi, India, May 2018.
- [7] M. Azees, P. Vijayakumar, and L. J. Deboarh, "EAAP: efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2467–2476, 2017.
- [8] S. M. Pournaghi, B. Zahednejad, M. Bayat, and Y. Farjami, "NECPPA: a novel and efficient conditional privacy-preserving authentication scheme for VANET," *Computer Networks*, vol. 134, pp. 78–92, 2018.
- [9] P. Vijayakumar, M. Azees, and L. J. Deborah, "CPAV: computationally efficient privacy preserving anonymous authentication scheme for vehicular ad hoc networks," in *Proceedings of the IEEE 2nd International Conference on Cyber Security and Cloud Computing*, pp. 62–67, New York, NY, USA, November 2015.
- [10] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 416–432, Warsaw, Poland, May 2003.
- [11] H. Zhong, S. Han, J. Cui, J. Zhang, and Y. Xu, "Privacy-preserving authentication scheme with full aggregation in VANET," *Information Sciences*, vol. 476, pp. 211–221, 2019.
- [12] I. A. Kamil and S. O. Ogundoyin, "On the security of privacy-preserving authentication scheme with full aggregation in vehicular ad hoc network," *Security and Privacy*, vol. 3, no. 3, Article ID e104, 2020.
- [13] Q. Mei, H. Xiong, J. Chen, M. Yang, S. Kumari, and M. K. Khan, "Efficient certificateless aggregate signature with conditional privacy preservation in IoV," *IEEE Systems Journal*, vol. 15, no. 1, pp. 245–256, 2020.
- [14] Z. Xu, D. He, N. Kumar, and K. K. R. Choo, "Efficient certificateless aggregate signature scheme for performing secure routing in VANETs," *Security and Communication Networks*, vol. 2020, Article ID 5276813, 11 pages, 2020.
- [15] J. Cui, J. Zhang, H. Zhong, R. Shi, and Y. Xu, "An efficient certificateless aggregate signature without pairings for

- vehicular ad hoc networks,” *Information Sciences*, vol. 451–452, pp. 1–15, 2018.
- [16] I. A. Kamil and S. O. Ogundoyin, “An improved certificateless aggregate signature scheme without bilinear pairings for vehicular ad hoc networks,” *Journal of Information Security and Applications*, vol. 44, pp. 184–200, 2019.
- [17] Y. Zhao, Y. Hou, L. Wang, S. Kumari, M. K. Khan, and H. Xiong, “An efficient certificateless aggregate signature scheme for the Internet of Vehicles,” *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 5, Article ID e3708, 2020.
- [18] G. Thumbur, G. S. Rao, P. V. Reddy, N. B. Gayathri, D. V. R. K. Reddy, and M. Padmavathamma, “Efficient and secure certificateless aggregate signature-based authentication scheme for vehicular ad hoc networks,” *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1908–1920, 2021.
- [19] N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.
- [20] D. Pointcheval and J. Stern, “Security arguments for digital signatures and blind signatures,” *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.
- [21] A. Maria, V. Pandi, J. D. Lazarus, M. Karuppiah, and M. S. Christo, “BBAAS: blockchain-based anonymous authentication scheme for providing secure communication in VANETs,” *Security and Communication Networks*, vol. 2021, Article ID 6679882, 11 pages, 2021.
- [22] J. Liu, L. Wang, and Y. Yu, “Improved security of a pairing-free certificateless aggregate signature in healthcare wireless medical sensor networks,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5256–5266, 2020.
- [23] Q. Wang, D. Wang, C. Cheng, and D. He, “Quantum2fa: efficient quantum-resistant two-factor authentication scheme for mobile devices,” *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2022.
- [24] D. Wang and P. Wang, “Two birds with one stone: two-factor authentication with security beyond conventional bound,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708–722, 2018.
- [25] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, “The quest to replace passwords: a framework for comparative evaluation of web authentication schemes,” in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pp. 553–567, San Francisco Bay Area, CA, USA, May 2012.

Research Article

A New Malware Detection Method Based on VMCADR in Cloud Environments

Luxin Zheng ¹ and Jian Zhang ^{1,2,3}

¹School of Computer Science and Engineering, Tianjin University of Technology, Tianjin 300384, China

²College of Cyber Science, Nankai University, Tianjin 300350, China

³Tianjin Key Laboratory of Network and Data Security Technology, Tianjin 300350, China

Correspondence should be addressed to Jian Zhang; jeffersonzj@qq.com

Received 15 September 2021; Revised 21 December 2021; Accepted 8 March 2022; Published 27 March 2022

Academic Editor: Qi Jiang

Copyright © 2022 Luxin Zheng and Jian Zhang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the cloud computing technology developing increasingly, malware and privacy protection have become two major challenges for cloud security. At present, the detection methods based on virtualization technology are mainly in-VM and out-of-VM approaches, both of which have high detection rates. However, a lot of relevant researches at present have focused on the accuracy of malware without considering the privacy protection of cloud tenants sufficiently. In this paper, we propose a new cloud-based malware detection method that can detect malware in cloud service platforms without compromising user privacy. In order to protect the privacy of cloud tenants, this method uses relevant virtualization technologies to obtain memory snapshots of cloud tenants. Because the memory snapshot is very large, and the semantics is of low level, it needs to be processed for feature dimensionality reduction. Therefore, we propose visualized memory change area dimensionality reduction (VMCADR) method. This method directly performs malware detection on binary memory snapshots without accessing user system information and files, thereby protecting user privacy. The following are the main steps of VMCADR method. First, we propose memory difference (MDIFF) algorithm to obtain the Memory Changed Area (MCA), which is changed by the test program. Then, in order to better detect the MCA files, we use visualization technology to process it. Next, we convert these MCA files into grayscale images and RGB images, respectively. And we resize the picture pixels uniformly, so that it can be classified using convolutional neural networks. Finally, we propose a Simplified Neural Network (SNN) to classify these images. After experiments, the RGB-dataset accuracy of malware detection is 99.39%.

1. Introduction

With the cloud computing technology developing increasingly, the malware detection is getting more and more challenging in cloud. In order to avoid detection in cloud, malware programs always change their code while propagating [1, 2].

However, many relevant researches at present have focused on the accuracy of malware detection without sufficiently considering the privacy protection of cloud tenants. Traditional antivirus methods based on signature scanning are challenged by malware polymorphism and antagonism [3]. Types of malware include viruses, worms, Trojan horses, backdoors, spyware, ransomware, bots, and rootkits [4]. In

order to allow the malware to perform similar functions, attackers can share most of the code, so that the structure of the malware variants always has small differences [5].

There are some challenges with the existing malware detection methodology. First, many classification methods that extract asm functions by disassembling files have reached high accuracy. These methods usually require deploying professional disassembly software on the tenant's virtual machine, or submitting suspicious files to the server for disassembly. However, due to privacy protection, most cloud tenants may not allow antivirus service providers to access sensitive information in their files, which makes it impossible to extract asm functions. Second, most signatures are based on static analysis of executable file code, but

malware developers often evade static analysis by obfuscation specific modifications to the malware's code. Third, the antivirus software installed on the inspected machine is considered untrustworthy. The inspection mechanism is considered trusted only if the inspected instance does not know the inspection process and cannot interfere with the inspection process. Although some advanced antivirus solutions perform dynamic analysis by simulating the virtual environment and checking the behavior of files, the complex malware can still detect and evade the scanning process.

Researchers rarely share the datasets used in their work, and there is no available malware dataset that can be regarded as a reference dataset [6]. Researchers generally obtain live malware files from AV agents (such as VirusTotal, VirusShare, and VXheaven) to perform malware analysis and detection although there are some existing datasets, such as the big Challenge dataset released by Microsoft in 2015. This dataset is created based on static analysis. And it has the same advantages and limitations as the analysis technology.

In this paper, we propose a credible malware detection method. The advantages of the proposed approach are that we directly perform malware detection on binary memory snapshots without accessing user system information or files. Therefore, it can protect user privacy. The main contributions of this paper are described as follows:

- (i) We propose a malware detection method based on memory images to protect user privacy. By using out-of-VM method, the memory is obtained from the VM running test program. Therefore, this method has high reliability. And we do not need to introspect the VM memory to high-level semantics or access user-system information or files. Besides, we directly perform malware detection on binary memory snapshots. Therefore, the method can protect the privacy of cloud tenants.
- (ii) We propose an MDIFF algorithm to reduce the size of the feature data. The memory data is very large, but the effective feature data that we need to analyze only occupies a small part of the memory. The effective feature data is the changes to the memory caused by the behaviors of the test program. And we call the data MCA. Therefore, we use the MDIFF to obtain the MCA file.
- (iii) We propose an improved neural network model (Simplified Neural Network (SNN)) to perform more accurate and efficient classification. Because the obtained MCA file is a binary file, in order to effectively classify it, we combine the visualization technology to convert it into a grayscale image and an RGB image. In order to improve the detection rate and detection efficiency, we modified the VGG16 model to obtain the optimized SNN model, which has a better detection rate.

The rest of this paper is arranged as follows. Section 2 introduces related work. The third part describes the method structure. Section 4 describes the implementation of the detection method. Section 5 introduces dataset creation and

the experiment environment. Section 6 describes the evaluation of the experiment. Section 7 introduces the authors' conclusion.

2. Related Work

In this section, we review the related technologies and methods involved in malware detection in the cloud environment.

2.1. Cloud Computing and Virtualization. Cloud computing has become an indispensable part of IT infrastructure, including both resources and services (private or public) based on cloud computing. With the rapid development of processing and storage technology, computing resources have become cheaper than ever before. With more powerful functions and higher availability, these advancements have opened the door to new "cloud computing" technologies [7]. In cloud computing environment, resources (such as storage, memory, and data) are rented by users/consumers. Virtualization is the core technology within the cloud computing. Virtualization allows multiple operating systems to running together (isolated from each other) on the same physical server. The main software component that enables and monitors virtualization is the host hypervisor. The hypervisor acts as an additional layer between the physical and virtual domains. It manages the hardware resources of the system in order to efficiently allocate them among VMs.

Malware remains one of the main cyber-attack techniques against organizations. This reality has introduced a new business model in cybercrime and ransomware. Therefore, the attackers can encrypt the organization's critical data and files in the Storage Area Network (SAN) of the server and endpoints in this way. The attacker only provides the decryption key after the victim has paid the ransom. Currently, ransomware is a major cyber threat against individuals and organizations (especially against organizational VMs in the computing cloud). Nahmias et al. [7, 8] proposed a method using virtualization at the core of the cloud architecture to generate credible malware signatures based on the presence of malware processes in memory. By querying the hypervisor of the VM, they extracted the volatile memory dump of the VM. When the malware is running on the VM, these dumps (binary file) are obtained dynamically in a trusted way. These dumps include the behaviors of the malware in memory.

2.2. Convolutional Neural Networks. Nowadays, deep neural networks and deep learning have achieved outstanding performance in solving many important problems in computer vision [9], speech recognition, and natural language processing [10]. One of the main reasons why deep learning algorithms are becoming more and more popular is their ability to learn representations [11].

Regarding the use of neural networks for supervisory tasks, every layer, except the last layer, performs a kind of representation learning. The representations learned from

these layers are forwarded to the last layer, which performs linear classification (such as softmax). In principle, after the hidden layer of the network learns to transform the input data into a good representation, the representation can be used as input to any other machine learning model. According to the definition, a good representation can reflect the posterior distribution of the basic explanatory factors of the input data [12].

Convolutional Neural Networks (CNN) [13, 14] rely on the basic knowledge of the topological structure of the input dimensions. And they calculate each low-level feature based on a subset of the given input. In the context of images, each feature is calculated from a fixed-size subimage. This is due to the assumption that the patterns in the input data may appear in different locations. Therefore, a pooling layer is usually used to aggregate all the values calculated from the same feature detector. CNN can be regarded as a feature extractor. It can sequentially scan the topology of the input data and obtain a topological representation. Essentially, the malware features are usually used as input data for antivirus classification programs. Therefore, malware features can be regarded as representations of malware executable files.

2.3. Visual Analysis Method. Visual analysis technology has made a major breakthrough in malware identification. Compared with traditional static analysis methods, it contains fewer features and higher detection accuracy [15]. The method using visualization technology not only inherits the advantages of traditional malware detection technology, but also can extract high-dimensional internal features from data samples. Nataraj et al. [16] first proposed a method to visualize malware as grayscale images. In this method, a given malware binary file is read as a vector of 8-bit unsigned integers and then organized into a 2D array, which can be visualized as a grayscale image in the range [0, 255] (0: black, 255: white). Ni et al. [17] proposed a malware classification method using SimHash and CNN. They converted the disassembled malware code into grayscale images based on SimHash, and they classified the malware family through CNN. Their method successfully utilizes the features provided by the opcode sequence and achieves a very high malware classification accuracy rate. Qiao et al. [15] proposed a unique multichannel visualization method for malware classification based on deep learning. The method enhances the applicability of malware detection. Their method uses both binary byte data and malware disassembly files and combines them effectively. Naeem et al. [18] designed an architecture to detect malware attacks on Industrial Internet of Things (IIoT). In order to analyze the malware in depth, a method based on color image visualization and deep convolutional neural network is proposed. The accuracy of this method on the IIoT dataset reached 98.47% on the Windows dataset.

2.4. Malware Detection in Cloud Environments. Malware detection methods are generally divided into static detection and dynamic detection. The dynamic analysis methods are to identify malware software by analyzing information such as

application behavior in the running program. Compared with static analysis, dynamic analysis has higher accuracy, but high time overhead. Dynamic detection technology is currently mainly divided into in-VM and out-of-VM detection.

Willems et al. [19] proposed CWSandbox, a behavior-based dynamic malware analysis tool. It can execute malware samples in a simulated environment to monitor system calls and automatically generate detailed reports. The CWSandbox tool is widely used in dynamic analysis. Mosli et al. [20] used the Cuckoo sandbox to extract features such as registry activities, imported libraries, and API function calls. The linear support vector machine classifier using registered activity features achieves the highest accuracy rate of 96%. However, some malwares can easily detect the security sandbox environment to evade analysis. Sihwail et al. [21] collected datasets from the VirusTotal and Das Malwerk repositories. They used Cuckoo Sandbox and Volatility to generate two reports during each executable period. They extract the unique API call in each report as a feature and convert it into a binary vector. Finally, the result is improved by combining the two reports. The accuracy and FPR of the SVM classifier of this method are 98.5% and 1.7%, respectively.

The experimental environments of the above three methods are all carried out in a sandbox environment, and they all belong to in-VM dynamic detection. The low reliability of data is still the main challenge for in-VM detection, because the acquired data may have been tampered with by malware.

Huang and Stokes [22] proposed a new multitask, deep learning architecture for dynamic malware classification. They used data extracted from the dynamic analysis of malware and normal software to train their model. And they achieved 2.94% error rate of malware family classification. However, because dynamic malware analysis is time-consuming, effectively detecting malware attacks in suspicious files is challenging. Lin et al. [23] proposed a method based on virtual time control mechanics to overcome this challenge. This method uses an improved Xen hypervisor, in which a virtual clock source is generated according to a predefined speed ratio. It speeds up the sandbox system running on the modified hypervisor. The experimental results show that this method speeds up the running speed of the system timer and increases the record size by 41.54%. Nissim et al. [24] obtained memory snapshot of the VM running the malware program through VMM. Then, the method uses WinDbg to extract system calls. Finally, the authors use sequential mining method to analyze system calls. The authors evaluated their methods on ransomware and RATs. Their results show that the proposed method can effectively detect unknown malware. The F-measure value and FPR of the SVM classifier of this method are 100% and 1.4%, respectively.

Wang et al. [25] propose a method to detect kernel rootkits. When the executable program is running on the VM, this method captures a memory snapshot of the VM every 10 minutes, a total of 100 times. Then, they extract several types of data as features in the memory snapshot,

such as threads. The accuracy and FPR of this method for detecting unknown kernel rootkit attacks are 99.8% and 7.6%, respectively. The above four methods are out-of-VM dynamic detection. The reliability of the data they obtain is higher than that of in-VM dynamic detection. However, binary memory snapshots are unreadable data, but what we need is readable high-level semantic data. Therefore, this type of method requires semantic reconstruction of memory snapshots, which will increase time overhead.

In the above methods, most of them need to access the private information of user, such as API calls, system calls, processes, threads, and executable files. This will affect the privacy and security of user. From the perspective of privacy protection, the method we propose should perform malware detection without obtaining information about the user system.

Therefore, this paper only analyzes binary memory snapshots obtained from out-of-VM. In this way, this method can not only ensure the reliability of the acquired data, but also perform malware detection on the premise of protecting user data privacy.

3. Cloud-Based Malware Detection Method

3.1. The Architecture of the Detection Method. We will introduce the architecture of the detection method in this section. In order to eliminate the interference of other system factors on the experiment, we need to create a VM installing pure operating system without other applications as a restoring point. This can ensure that each program is running in the same environment when it is executed. Therefore, we create a pure VM at first. Then, we save the current operating state of the system, so that the virtual machine can be restored to the current system state at any time.

The steps required for one test program of this method are shown in the figure (see Figure 1). In step 1, we obtain the original memory dump file from the VM installing the operating system without other applications. Next, we use the drakvuf [26] tool to make the test program running in the VM automatically in step 2. In the step, we use python script to automatically obtain memory dump file of VM running test program within a certain period of time continuously. In the step of 45, we use the MDIFF algorithm to compare the acquired memory with the original memory dump file to get the MCA file. Immediately after that, we convert all of the MCA files into grayscale and RGB images of the same size in the step of 6. Finally, we propose a neural network model (SNN) to classify these images in the step of 78. The detection result will be compared with the current mainstream model, such as Inception-ResNet-v2 model. This method does not need to perform introspection on the user VM or obtain high-level semantic information of the current state of the user VM system. Therefore, this method protects the user privacy during the entire detection process.

3.2. Motivation Underlying Detection Method. Many studies have focused on the accuracy of malware detection

without fully considering the privacy protection of cloud tenants. They can obtain the application process, system calls, and a lot of private information currently used by the tenant through various methods. Their purpose is to obtain the state of the malware program, while it is running in the memory; besides, they can also obtain other sensitive information of the user. However, those methods may reveal user privacy; we need a method that does not affect user privacy to obtain similar features. The state of the malware program in the memory can be seen as its specific modification to the memory. Therefore, on the premise of protecting user privacy, we can use an algorithm to directly obtain the memory changed area caused by test program. We expect that the MDIFF algorithm can find out all the memory changes made by test programs. When malware programs or normal programs run in system memory, their memory allocation patterns might be different. We hope that the differences or changes obtained by MDIFF algorithm can represent these patterns.

4. Methods

4.1. Out-of-VM Memory Snapshot Acquisition. The virtualization platform we use is based on the Xen environment. Xen is an open source virtual machine monitor developed by the University of Cambridge. It has good isolation and can effectively isolate virtual machines. In the Xen environment, there are two main components. One is the virtual machine monitor (VMM) or called hypervisor. The hypervisor layer is between the hardware and the virtual machine. And it is the first layer that must be loaded into the hardware. After the hypervisor is loaded, the virtual machine can be deployed. The virtual machine is called "domain." Among these virtual machines, one of them plays a very important role, that is, domain0, which has high privileges. Domain0 is responsible for some specialized work. Since the hypervisor does not contain any drivers that talk to the hardware, and there is no interface to talk to the administrator, these drivers are provided by domain0. In domain0 virtual machine, administrators can use some Xen tools to create other virtual machines or acquire memory snapshot.

We used shell language to write a script that can automatically complete the entire process of taking a snapshot of the VM running test program. Once the malware sample starts running on the VM, the script will take a snapshot of the VM with a given number of snapshots at a given time interval. Because Xen has good isolation, the virtual machines are transparent to each other. Besides, the malware inside the virtual machine is completely ignorant of the snapshot process; therefore, it cannot interfere with it. However, during the snapshot process, the virtual machine needs to be paused.

4.2. Visualized Memory Change Area Dimensionality Reduction (VMCADR) Methods. In order to effectively detect malware programs, we need to obtain relevant features from memory. However, because the memory snapshot data is

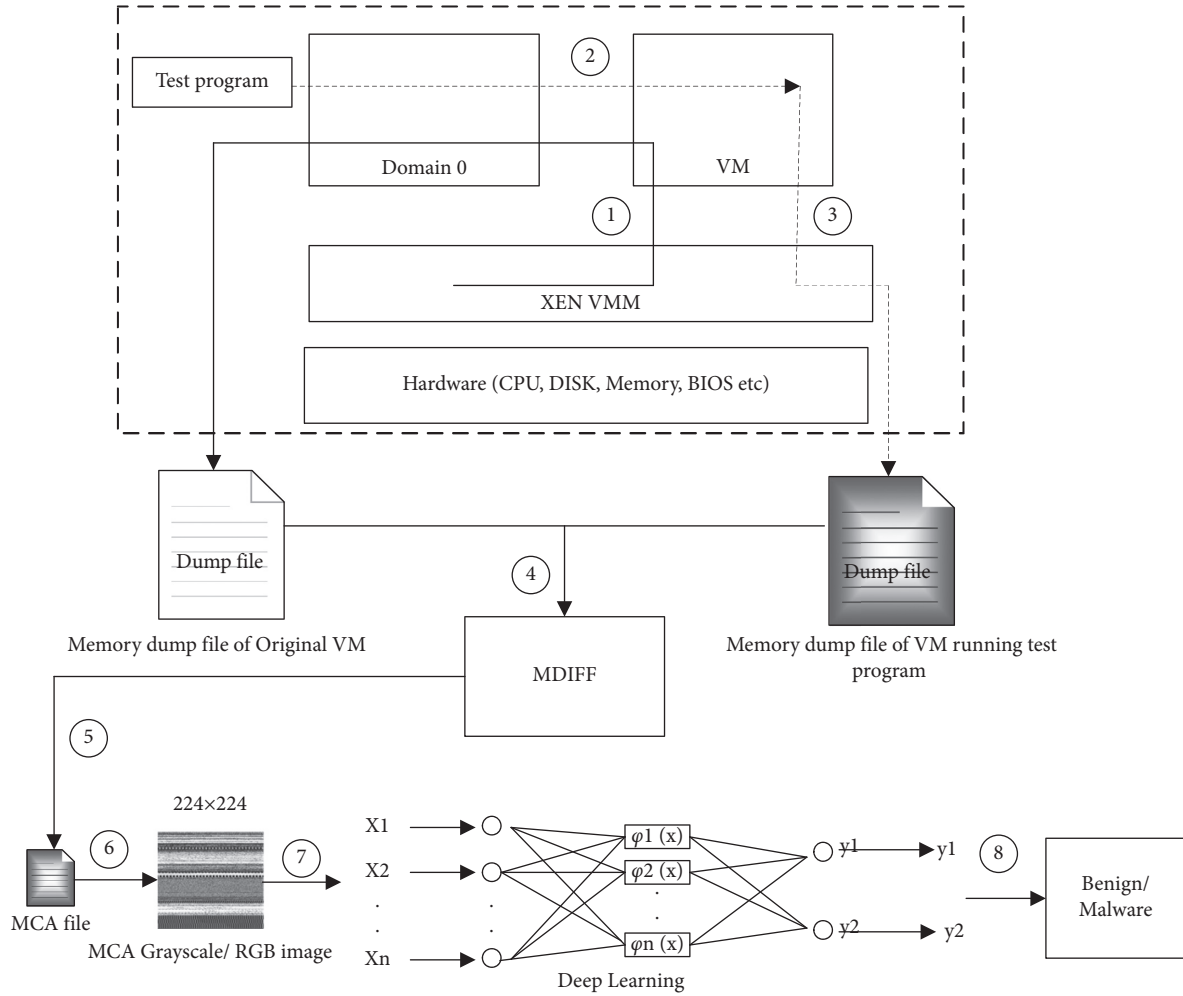


FIGURE 1: The overview of the detection method architecture.

very large, we need to obtain the effective features of the memory by using a series of techniques.

Jian et al. [27] research verifies the superiority of three-channel RGB images compared to grayscale images in malware detection. It compares the contributions of different channels and shows that data augmentation technology can use visualization technology to make a contribution to malware identification.

Therefore, we propose a visualized memory change area dimensionality reduction (VMCADR) method to achieve memory feature dimensionality reduction and transformation. The main technology of this method is mainly divided into three parts. They are described as follows.

4.2.1. MDIFF Algorithm. When the test program is loaded into the memory, it will make changes to the memory. After we obtain the memory snapshot of the program, the way how to find the memory change area of the program is key to our research. Since the memory snapshot is stored in a binary format, we need to compare the binary files to find out the change area accurately. It is obviously unreliable to update the difference file obtained through a simple binary

comparison. Because the impact of the memory stack may not be considered. In order to obtain the memory change area of the program effectively, we propose an algorithm—memory different (MDIFF) algorithm.

MDIFF is an excellent difference algorithm, which draws on a current mainstream update algorithm [28]. And we modify and optimize the algorithm. The basic flow of the MDIFF algorithm is shown in Algorithm 1. Next, we will introduce the main steps of the MDIFF algorithm.

MDIFF can be divided into three parts. First, we sort the contents of the old (original memory snapshot) by sorting technology to form a lexicographic order I . The sorting method we use is faster suffix sorting method with a complexity of $n \log n$ and a space complexity of $O(n)$.

The second step is the core of the algorithm; we will find the longest matching len between old and new (memory snapshot of running test program) through dichotomy. Then, we get the MCA through len. We need to introduce several parameter variables in this step. The scan represents the character to be queried in new, pos represents the matched character in old and the lastscan = scan - lenb, lastpos = pos - lenb, lastoffset = scan - pos. Lastoffset is the offset between new and old. If the area in old can be found in

```

Input: original memory snapshot, memory snapshot of running test program
Output: MCA file
The old represents the original memory snapshot, the new represents memory snapshot of running test program;
Faster suffix sorting (old)
    Return lexicographic order I;
By using the lexicographic order I, find a position pos in old. The pos maximizes the  $k$  of  $\text{new}[\text{scan}, \text{scan} + k] = \text{old}[\text{pos}, \text{pos} + k]$ 
Return  $\text{len} = k + 1$ ,  $\text{offset} = \text{pos} - \text{scan}$ .
While ( $\text{scan} < \text{newsize}$ )
{
    If the length of  $\text{old}[\text{scan}, \text{scan} + \text{lastoffset}]$  and  $\text{new}[\text{scan}, \text{scan} + \text{len}]$  does not match with more than 8 bytes then
        Divides the forward-extension ( $\text{lenf}$ ) of the former completely match area and the backward-extension ( $\text{lenb}$ ) of the latter completely match area. The remaining part between the two completely match areas is used as MCA ( $i$ ) ( $i = 1, 2, \dots, n$ )
        Return  $\text{lenf}$ ,  $\text{lenb}$ , MCA ( $i$ ).
    else
        continue.
    Integrate all MCA ( $i$ ) areas into one MCA file. And the MCA file obtained is the final output.
}

```

ALGORITHM 1: The basic flow of the MDIFF algorithm.

new ($\text{area} + \text{lastoffset} = \text{area}$) in new, it is considered that the area in old and new is completely match area, such as area 2 and area 4 (see Figure 2). The scsc represents the starting position of the comparison between new and old. And the starting position in old is $\text{scsc} + \text{lastoffset}$. lenf stands for forward-extension, and lenb stands for backward-extension. By using the lexicographic order I , we find a position pos in old. The pos maximizes the k of $\text{new}[\text{scan}, \text{scan} + k] = \text{old}[\text{pos}, \text{pos} + k]$. And then, the $\text{len} = k + 1$, $\text{offset} = \text{pos} - \text{scan}$. If the length of $\text{old}[\text{scantoscan} + \text{lastoffset}]$ and $\text{new}[\text{scantoscan} + \text{len}]$ does not match with more than 8 bytes, the MDIFF algorithm will divide the forward-extension (lenf) of the former completely match area and the backward-extension (lenb) of the latter completely match area (see Figure 2). The remaining part between the two completely match areas is used as MCA (i). Finally, we obtain the MCA file as the integration of all MCA (i).

4.2.2. Image Conversion. Because the MCA file we obtained is an unreadable binary file, we combine it with visualization technology. In terms of image visualization technology, we used the visualization method of Pinhero [29]. First of all, we read the MCA file byte by byte and map the binary content of each byte to a decimal value in the range of 0 to 255. By doing this, each MCA file has been converted into a 1D vector of decimal numbers.

We need two consecutive decimal values from a 1D array to draw pixels. By using a random number generator [29], a pixel in the grayscale image is plotted through using a 2D color map. We use the decimal values as the row index and column index, respectively. The generated 1D pixel array is reshaped into a 2D matrix and visualized as a grayscale image. The process of converting an MCA file into a grayscale image is shown in the figure (see Figure 3). The pixels in an RGB image are specified by the number of red, green, and blue. In order to obtain the contribution of red, green, and blue in the pixels, three different colormap of

red, green, and blue are used. To draw the pixels in an RGB image, two consecutive values in a 1D vector array are used as row and column indexes [29]. The generated pixel array is reshaped into a 2D matrix and visualized as an RGB image ultimately. The process of converting an MCA file into an RGB image is shown in Figure 4.

The width of the generated image can be set manually. And the length will automatically change according to the size of the MCA file. In our experiment, the relationship between the width of the image and the size of the MCA file is shown in Table 1. The size of the MCA file determines the width of the image it generates. For example, when the size of a MCA file is 1 MB, the width of the image it generates is 1024.

4.2.3. Image Processing. The pixels of the acquired picture are different due to the size of the MCA file. The purpose of memory image processing is to resize the image while preserving the features of the original image as much as possible. In this way, the images can be classified using a convolutional neural network. We used image interpolation to reduce the size of the original image. The principle of the interpolation method is to calculate the value of the target point by using parameters such as the pixel values of several points around the target point. We used interpolation calculation methods to include bilinear interpolation, bicubic interpolation, and Lanczos interpolation [30].

The principle of bilinear interpolation is to use the interpolation algorithm to calculate the value of the target point according to the values of the nearest 4 points around the target point. The image quality obtained by the bilinear interpolation algorithm is the worst, but it has the fastest image processing speed.

The bicubic interpolation algorithm is improved on the basis of the bilinear interpolation algorithm, but the complexity is also increased. The interpolation algorithm not only uses the values of the four directly adjacent pixel points around the target point to participate in the interpolation

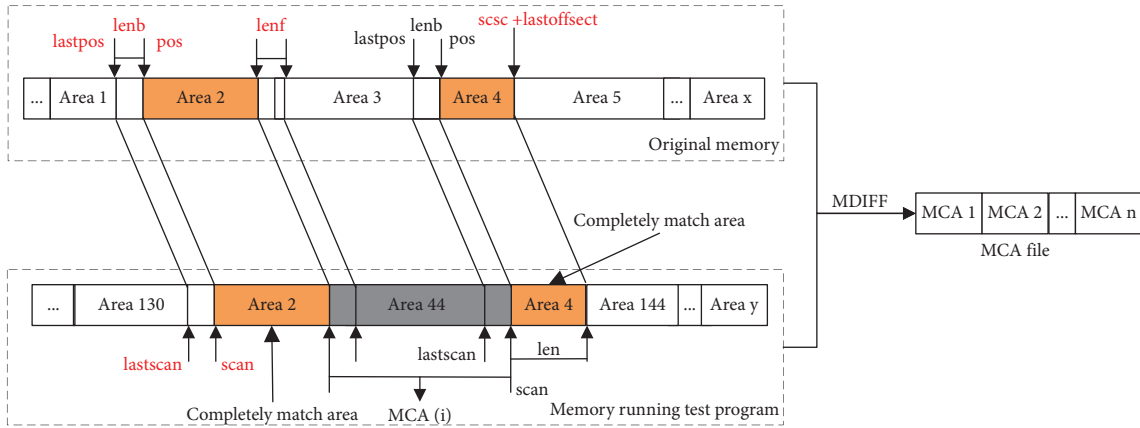


FIGURE 2: The core principle of MDIFF algorithm.

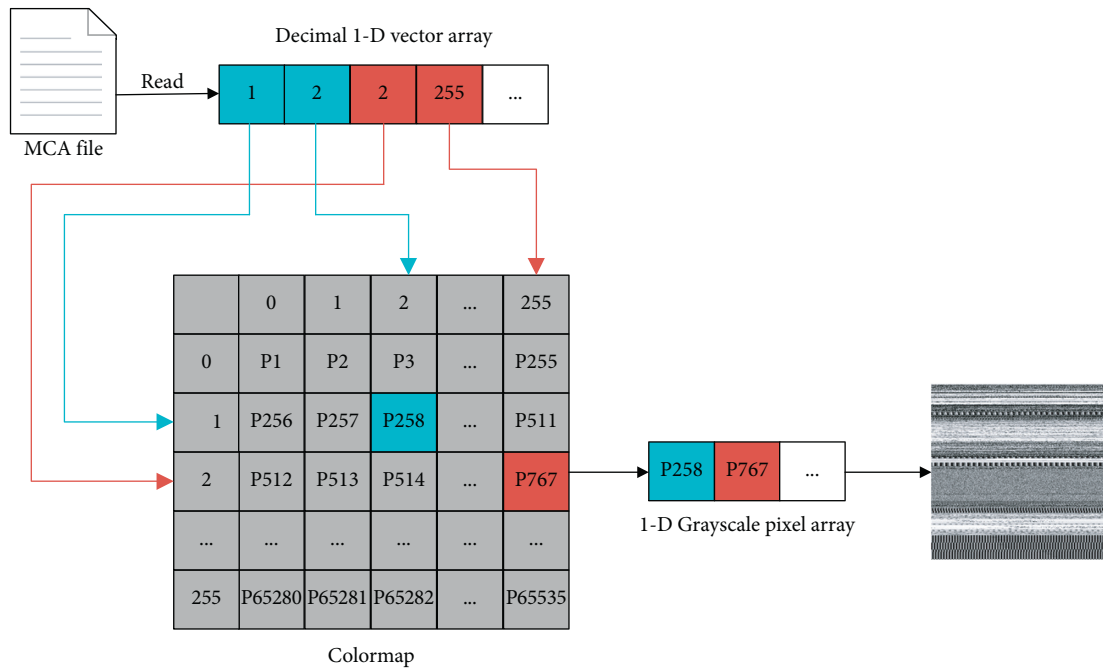


FIGURE 3: Principles of grayscale image generation.

calculation, but also uses the rate of change of the values of these four points to perform the interpolation calculation. The interpolation algorithm uses the values of 16 pixels near the target point for cubic interpolation calculation, and the cubic polynomial $s(x)$ is also used in the calculation (see (1)). The interpolation algorithm processes images faster than bilinear interpolation. And the image quality obtained is higher than bilinear interpolation, but lower than the Lanczos interpolation algorithm.

The one-dimensional Lanczos interpolation algorithm mainly selects four points from the left and right of the sampling point for interpolation calculation and calculates the weights of these eight points through a high-order interpolation function. The two-dimensional Lanczos interpolation algorithm performs interpolation calculations on the adjacent eight points in the x -axis and y -axis directions,

respectively. The principle of Lanczos interpolation algorithm is as follows.

For a one-dimensional vector, the input point set is X , the window size is $2a$, and the weight of each point in the window is calculated. The weight calculation formula is as follows:

$$L(x) = \begin{cases} 1 & \text{if } x = 0 \\ \frac{\text{asin}(\pi x)\text{sin}(\pi x/a)}{\pi^2 x^2} & \text{if } 0 < |x| < a \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

After the weight of each point in the window is obtained, the weighted summation of the points in the window s_i can

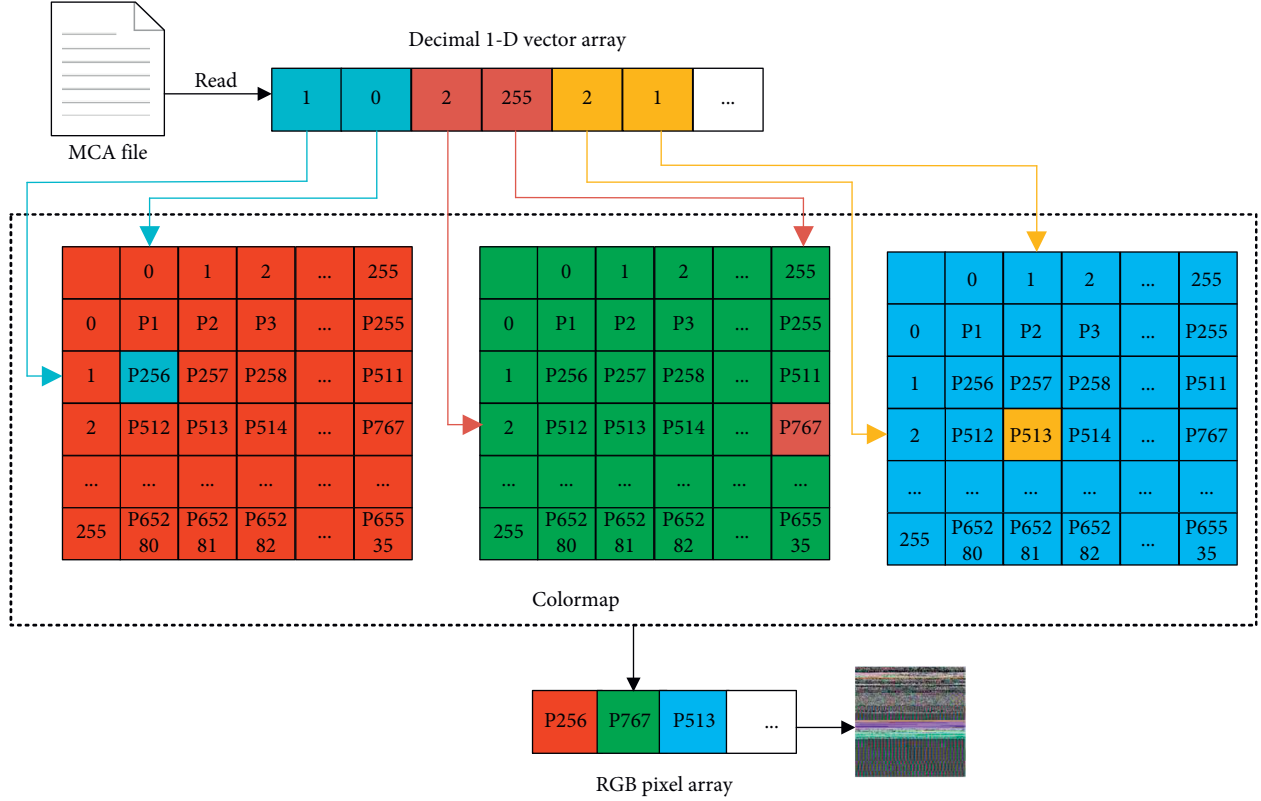


FIGURE 4: Principles of RGB image generation.

TABLE 1: The relationship between the width of the image and the size of the MCA file.

MCA file size (KB)	Image width
Between 0 and 10	32
Between 11 and 30	64
Between 31 and 60	128
Between 61 and 100	256
Between 101 and 200	384
Between 201 and 1000	512
Between 1001 and 1500	1024
Greater than 1500	2048

calculate the interpolation value $s(x)$ at the window, as shown in the following formula:

$$S(x) = \sum_{i=x-a+1}^{x+a} s_i L(x-i). \quad (2)$$

From one-dimensional interpolation to two-dimensional interpolation, the weight calculation formula is shown in the following formula:

$$L(x, y) = L(x)L(y). \quad (3)$$

When interpolating the image, we set $a=4$, and the window size is 8CE8, which means that 64 pixels are used to represent the target pixel. The interpolation formula is shown in the following formula:

$$S(x) = \sum_{i=x-a+1}^{x+4} \sum_{j=y-a+1}^{y+4} s_{ij} L(x-i)(y-j). \quad (4)$$

In our experiments, we perform image processing on two datasets with 3 interpolation algorithms. The specific results are shown in Table 2. The time overhead of these three interpolation algorithms is not very different in our experiments. The Lanczos interpolation algorithm is the slowest to process images, but the image quality is the best. And the time overhead of the least expensive bilinear interpolation algorithm and the most expensive Lanczos interpolation algorithm is less than 2 minutes. Therefore, in order to preserve the relevant features of the original image as much as possible to achieve better detection results, this paper uses the Lanczos interpolation algorithm to resize the image.

4.3. Deep Learning. In order to confirm whether the MCA file effectively contains the performance characteristics of the malware in the memory, we need to find a specific method to classify the obtained grayscale image and RGB image data sets, respectively. In recent years, due to the elimination of a lot of feature engineering work, there have been more and more malware classification methods based on malware images and deep learning [31]. The current mainstream neural network models (such as VGG16, VGG19, and Inception-ResNet-v2) have high detection rates

TABLE 2: Interpolation algorithm time cost comparison.

	Bilinear interpolation (second)	bicubic interpolation (second)	Lanczos interpolation (second)
Grayscale	400.2181	442.9408	478.7009
RGB	381.6803	392.2938	417.8827

on the ImageNet dataset [32], which consists of 14 million labeled images.

We need to choose a suitable mainstream deep learning model to classify the dataset. The purpose of this step is to confirm whether the dataset is valid in the malicious detection process. We found that most models have a high detection rate for our dataset. In order to reduce the time overhead, we choose the VGG16 network model with a small number of layers as the basis. And we make adaptive improvements to it. In order to highlight the advantages of our proposed model, we choose the V2 model with the highest detection rate as the comparison model.

4.4. Simplified Neural Network (SNN). We have designed a deep learning model (SNN) that is suitable for memory image classification. The model is improved based on the vgg16 model. The specific architecture of SNN is shown in Figure 5.

The size of the convolution kernel in the SNN network structure is 3CE3, and the step size is 1. And we use multiple continuous 3CE3 small convolution kernels instead of large convolution kernels. The dashed box in the figure is a convolution block. The pooling layer uses the maximum pooling with a window size of 2CE2 and a step size of 2. Because the memory image does not have a clear object or target, the depth of the network model is too deep to easily cause overfitting, and we reduce the depth of the network to improve the accuracy of detection. And it can also reduce the time overhead. In addition, we use global maximum pooling instead of the fully connected layer using the activation function. There are some benefits of using global maximum pooling. Global maximum pooling is to take the global maximum value of the feature map as the output. It does not take the maximum value in the form of a window but takes the maximum value in the unit of the feature map. It can reduce the parameters, avoid overfitting, and improve the generalization ability of the network model.

5. Dataset Creation and Experiment

5.1. Dataset Creation. One challenge of malware research is the lack of reference malware datasets [33]. Currently, some malware datasets are created based on static or dynamic analysis. However, the dataset is subject to the same limitations of analytical techniques. Datasets containing weak and irrelevant features may negatively affect the training of machine learning classifiers. In addition, it makes the method easily manipulated by malware evasion techniques.

In order to verify the effectiveness and feasibility of the detection method, we used a large number of real malware samples. In order to ensure the real-time and diversity of the

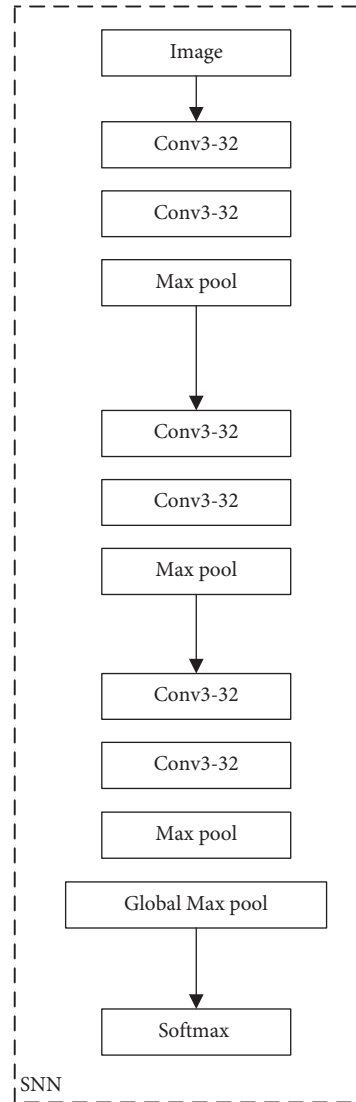


FIGURE 5: SNN architecture.

training samples, we directly connected the customer's virtual machine to the Internet. We have collected 220 normal executable software including browsers, video players, compression software, music players, Office software such as Word, novel accessing software, and other common software. When a normal program is running, we will perform related operations on it, such as opening a word document, playing music, and playing video. In addition, we also collected 440 malware executable files samples from VirusShare. It includes viruses, Trojan, worms, adware, backdoor software, and spyware. The sample set is shown in Table 3.

We create a VM installing pure operating system without other applications as a restore point. Then, each of the above samples will run in the restored VM. While the program is running, the time intervals at which we obtain memory snapshots in order are 10 seconds, 20 seconds, 30 seconds, 1 minute, and 1 minute. Each sample program can obtain 5 memory snapshots after running in the memory. It represents the manifestation of a program at a specific time in the

TABLE 3: Executable program dataset.

Type	Instances
Malware	440
Normal	220

memory. Therefore, we create a dataset containing 3300 memory snapshots, which includes 1,100 memory snapshots of running normal programs and 2,200 memory snapshots of running malware programs.

Due to the large memory data, we use the MDIFF algorithm to reduce the dimensionality of the memory data and obtain a smaller MCA file. Therefore, we got 3300 MCA file datasets, which contains 1100 MCA files of normal programs and 2200 MCA files of malware programs. However, the MCA file is a low-level binary file; therefore, we use visualization technology to process it. Through visualization technology, we convert MCA files into grayscale images and RGB images, respectively. Then, this paper adjusts the converted picture pixels to $224 * 224$. Finally, we obtained 3300-grayscale-image dataset and 3300-RGB-image dataset. It contains 2,200 malware MCA files images and 1,100 normal MCA file images, respectively, as shown in Table 4.

In this paper, the grayscale image dataset and RGB dataset are divided into training set, validation set, and test set according to 8:1:1.

5.2. Experimental Environment. The method is deployed on the hardware environment of GeForce Titan XP GPU (video memory: 12 GB), Intel Xeon E5-2600 CPU, 64G RAM, and 4TB HDD. We use Xen 4.9.1 to build a virtualized environment. The Ubuntu 16.04 64-bit operating system acts as a secure virtual machine (Domain0), and the Windows 7 Professional 64-bit operating system, 2G RAM, and 200G hard disk acts as a VM. We use the Xen tools to directly dump the memory of the VM.

5.3. Experimental Design. Our experimental design aims to evaluate the classification effect of our proposed classification algorithm on the dataset we generated. In order to evaluate the effect of the MDIFF algorithm, we use the mainstream deep learning model (Inception-ResNet-v2) to classify the grayscale image and RGB image converted from the memory modification file obtained by the MDIFF algorithm.

In order to improve the accuracy and efficiency of detection, we tested the above-mentioned datasets separately for our proposed SNN model. Then, we compare them with the results of Inception-ResNet-v2 model.

6. Evaluation

6.1. Evaluation Metrics. In our evaluation, we evaluate our proposed image dataset and our framework’s detection capabilities. In addition to using the metric Accuracy (see (5)) to judge the classification performance of the classifier, the TPR (see (6)) and FPR (see (7)) measure the proportion

TABLE 4: Image dataset.

Type	Instances grayscale	Instances RGB
Malware	2200	2200
Normal	1100	1100

of correctly identified positives and measure the proportion of negatives incorrectly classified as positives of the total number of negatives, respectively.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}. \quad (5)$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (6)$$

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}}. \quad (7)$$

In the experiment of this paper, the malware memory image is a positive sample, and the normal memory image is a negative sample. The confusion matrix is shown in Table 5.

6.2. Results. In order to prove the feasibility of the MDIFF method and SNN model proposed in this paper, the mainstream deep learning model is tested on the processed image dataset. We chose Inception-ResNet-v2 convolutional neural network models to compare them with the SNN model proposed in this paper. The reason why we chose the Inception-ResNet-v2 model for comparison experiments is worth noting. The Inception-ResNet-v2 model is a pre-trained model provided in Keras. This model is one of the best performing models in the field of image classification in recent years.

First of all, we set up a comparative experiment of two neural network models. When using a convolutional neural network for classification, the training time cost of the model has a greater impact on the overall time cost. Therefore, we compared the training costs of several models. Before comparing the time cost, we must first introduce a hyper-parameter (batch_size) that affects the training time. For the same model, the larger the batch_size setting, the shorter the training time. But it has higher requirements for the graphics card. Therefore, when setting the parameters, we set the largest batch_size much as possible. The specific values of this parameter for different models are shown in Table 6. It can be seen from the table that the batch_size of the Inception-ResNet-v2 model is set to 30. This is because the model has many parameters, which can be set to a maximum of 4 in our experimental environment. The time in Table 6 is the average time for training an epoch. It can be seen that the more complex the network model structure, the longer the time consumed.

We use the TensorBoard tool to directly visualize the changes in the accuracy of the model training process (see Figure 6). The vertical axis of the graph is the accuracy, and the horizontal axis is the number of batches passed. The effects of the two models will stabilize after being trained for 16 epochs. And the Inception-ResNet-v2 model will have a

TABLE 5: Confusion matrix.

	Predicted as malware	Predicted as normal
Malware	TP	FN
Normal	FP	TN

TABLE 6: Different model hyperparameter settings and training time.

	batch_size	Training time (s/epoch)
Inception-ResNet-v2 grayscale	30	39.8
Inception-ResNet-v2 RGB	30	35.3
SNN grayscale	30	9.6
SNN RGB	30	9.4

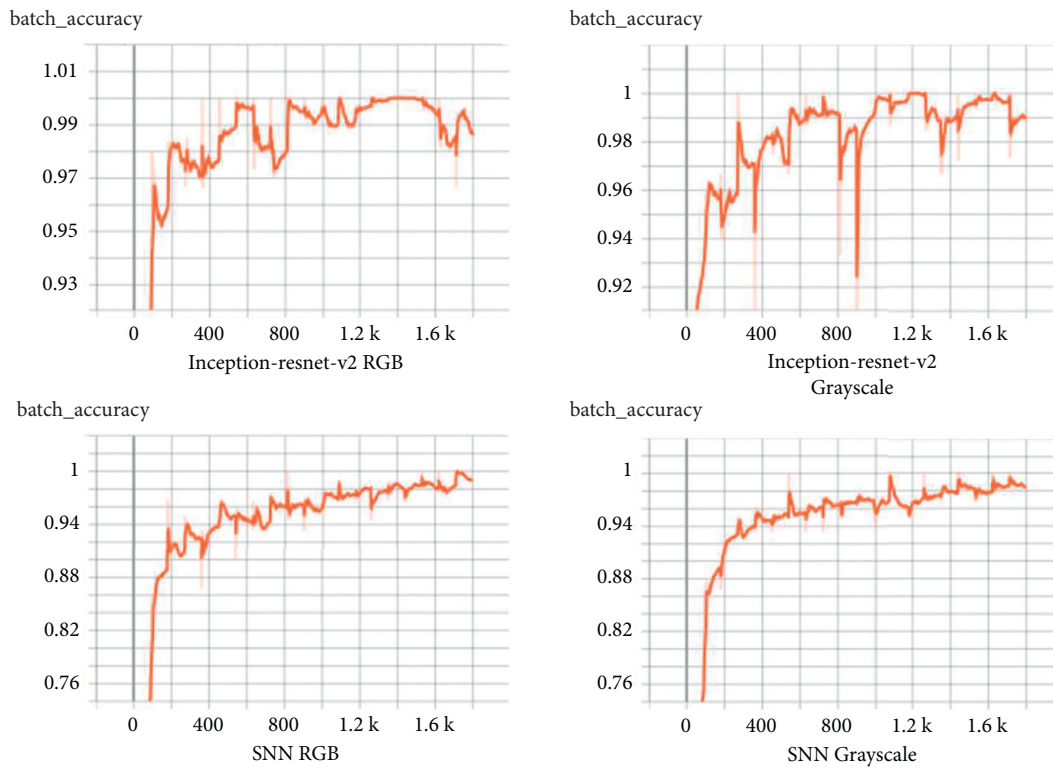


FIGURE 6: The accuracy rate change graph of the model training process.

certain degree of overfitting as the training epoch increases. The overall time cost and accuracy of the SNN model are better than those of the Inception-ResNet-v2 model. And the performance of RGB is better than that of grayscale images.

The model proposed in this paper has achieved 97.63% and 99.96% accuracy on the grayscale image and RGB verification set, respectively, which is better than the 94.35% and 98.96% of Inception-ResNet-v2.

It can be seen from the above that the feature set extracted by the MDIFF algorithm has outstanding performance in malware detection. The structure of the Inception-ResNet-v2 model is more complicated than that of the classification model proposed in this paper, but the classification effect is worse. The main reason for this result

is that the memory image has no clear objects or targets. In the learning process of complex network structure, it is possible to learn the unique characteristics of a certain picture rather than the common characteristics of such samples. Because of the huge network parameter, the Inception-ResNet-v2 model is easy to overfit. In summary, the MDIFF algorithm proposed in this paper has outstanding performance in extracting memory features. The model SNN performs better than traditional neural networks in classification. In addition, the detection effect of RGB images is better than the effect of grayscale images.

After the experiment, the specific data of the results are shown in Table 7. In summary, the SNN model architecture proposed in this paper has a higher detection rate than the Inception-ResNet-v2 model in terms of detection rate. And

TABLE 7: TPR and FPR of different classifiers on different feature type.

	TPR	FPR
Inception-ResNet-v2 grayscale	0.9484	0.1938
Inception-ResNet-v2 RGB	0.9787	0.0568
SNN grayscale	0.9606	0.1266
SNN RGB	0.9939	0.0259

the RGB picture has a higher detection rate than the grayscale picture.

7. Conclusion

In this paper, in order to achieve credible detection, we propose a credible malware detection method based on the VMCADR method that does not affect user privacy. In this work, when the malware process is active, we use the Xen tools to obtain a continuous memory snapshot of the VM running test program after a specific time interval. The whole process is credible, because we obtain the VM memory dump from outside of the VM running test program. And it is not interfered by malware running in the VM.

Because the acquired memory data is very large, we use VMCADR to process it. First, we propose an algorithm MDIFF to reduce the dimensionality of memory feature data. And the obtained data MCA file has the features that play an important role in subsequent malware detection. Because the obtained MCA file is a low-level binary file, we use visualization technology to process it. After converting the MCA file into grayscale and RGB images, we proposed our own neural network SNN model. The SNN model has the advantages of higher detection rate and lower time cost than the traditional Inception-ResNet-v2 model in detection.

7.1. Limitations. While VMCADR has an excellent detection rate on malware detection, it still has some limitations. First, the memory snapshot we get from the VM is 2 GB in our experiments. However, most of the VM memory on the cloud is larger than 2 GB. When taking a VM memory snapshot, we need to suspend the VM about 5 seconds, which may affect users. In addition, obtaining MCA files and visualization processing also require additional time overhead.

7.2. Future Work. The testing of multiple programs will be carried out in our future work. Besides, we will consider more reliable features into malware detection methods, such as hardware features of hardware counters. Because the data features generated by the underlying hardware have high reliability, it may be possible to further improve the accuracy of detection.

Data Availability

Some or all data, models, or code that support the findings of this study are available from the author Luxin Zheng upon reasonable request via e-mail.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank all of the team members and those who have helped this work. This work is supported by the National Key R&D Program of China (2016YFB0800805) and the Major Projects of Science and Technology Service Industry in Tianjin (16ZXFWGX00140).

References

- [1] J. D. Watson and F. H. C. Crick, "Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid," *Nature*, vol. 171, no. 4356, pp. 737-738, 1953.
- [2] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: a survey," *Journal of Information Security*, vol. 5, no. 2, pp. 56-64, 2014.
- [3] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of malicious code by applying machine learning classifiers on static features: a state-of-the-art survey," *Information Security Technical Report*, vol. 14, no. 1, pp. 16-29, 2009.
- [4] Y. Ye, T. Li, D. Adjeroh, and S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys*, vol. 50, no. 3, pp. 1-40, 2017.
- [5] L. Nataraj, S. Karthikeyan, and B. S. Manjunath, "SATTVA: spArsiTy inspired classification of malware VArants," in *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*, pp. 135-140, ACM, Portland Oregon, USA, 17 June 2015.
- [6] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123-147, 2019.
- [7] D. Nahmias, A. Cohen, N. Nissim, and Y. Elovici, "Trustsign: trusted malware signature generation in private clouds using deep feature transfer learning," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8, IEEE, Budapest, Hungary, 14 July 2019.
- [8] D. Nahmias, A. Cohen, N. Nissim, and Y. Elovici, "Deep feature transfer learning for trusted and automated malware signature generation in private cloud environments," *Neural Networks*, vol. 124, pp. 243-257, 2020.
- [9] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Ng, "Manning Convolutional-recursive deep learning for 3D object classification," *Advances in Neural Information Processing Systems*, vol. 25, pp. 656-664, 2012.
- [10] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 160-167, ACM, Helsinki, Finland, 5 July 2008.
- [11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, The MIT Press, Cambridge, MA, 2016.
- [12] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, 2013.
- [13] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541-551, 1989.

- [14] H. Yang, C. Yuan, B. Li et al., "Asymmetric 3D convolutional neural networks for action recognition," *Pattern Recognition*, vol. 85, pp. 1–12, 2018.
- [15] Y. Qiao, Q. Jiang, Z. Jiang, and L. Gu, "A multi-channel visualization method for malware classification based on deep learning," in *Proceedings of the 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 757–762, IEEE, Rotorua, New Zealand, 5 August 2019.
- [16] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th international symposium on visualization for cyber security*, pp. 1–7, ACM, Pittsburgh Pennsylvania, USA, 20 July 2011.
- [17] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Computers & Security*, vol. 77, pp. 871–885, 2018.
- [18] H. Naeem, F. Ullah, M. R. Naeem et al., "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, Article ID 102154, 2020.
- [19] C. Willems, T. Holz, and F. Freiling, "Toward automated dynamic malware analysis using cwsandbox," *IEEE Security and Privacy Magazine*, vol. 5, no. 2, pp. 32–39, 2007.
- [20] R. Mosli, R. Li, B. Yuan, and Y. Pan, "Automated malware detection using artifacts in forensic memory images," in *Proceedings of the 2016 IEEE Symposium on Technologies for Homeland Security (HST)*, pp. 1–6, IEEE, Waltham, MA, USA, 10 May 2016.
- [21] R. Sihwail, K. Omar, K. Zainol Ariffin, and S. Al Afghani, "Malware detection approach based on artifacts in memory image and dynamic analysis," *Applied Sciences*, vol. 9, no. 18, p. 3680, 2019.
- [22] W. Huang and J. W. Stokes, "MtNet: a multi-task neural network for dynamic malware classification," in *Proceedings of the International conference on detection of intrusions and malware, and vulnerability assessment*, pp. 399–418, Springer, San Sebastián, Spain, 7 July 2016.
- [23] C.-H. Lin, H.-K. Pao, and J.-W. Liao, "Efficient dynamic malware analysis using virtual time control mechanics," *Computers & Security*, vol. 73, pp. 359–373, 2018.
- [24] N. Nissim, Y. Lapidot, A. Cohen, and Y. Elovici, "Trusted system-calls analysis methodology aimed at detection of compromised virtual machines using sequential mining," *Knowledge-Based Systems*, vol. 153, pp. 147–175, 2018.
- [25] X. Wang, J. Zhang, J. Zhang, A. Zhang, and J. Ren, "TKRD: trusted kernel rootkit detection for cybersecurity of VMs based on machine learning and memory forensic analysis," *Mathematical Biosciences and Engineering*, vol. 16, no. 4, pp. 2650–2667, 2019.
- [26] "DRAKVUF eb/ol," <https://drakvuf.com/>.
- [27] Y. Jian, H. Kuang, C. Ren, Z. Ma, and H. Wang, "A novel framework for image-based malware detection with a deep neural network," *Computers & Security*, vol. 109, Article ID 102400, 2021.
- [28] P. Colin, "Naive differences of executable code," 2003, <http://www.daemonology.net/bsdif/>.
- [29] A. Pinhero, M. L. Anupama, P. Vinod et al., "Malware detection employed by visualization and deep neural network," *Computers & Security*, vol. 105, Article ID 102247, 2021.
- [30] S. Fadnavis, "Image interpolation techniques in digital image processing: an overview," *International Journal of Engineering Research in Africa*, vol. 4, no. 10, pp. 70–73, 2014.
- [31] B. Yuan, J. Wang, D. Liu, W. Guo, P. Wu, and X. Bao, "Byte-level malware classification based on Markov images and deep learning," *Computers & Security*, vol. 92, Article ID 101740, 2020.
- [32] L. Fei-Fei, J. Deng, and K. Li, "ImageNet: constructing a large-scale image database," *Journal of Vision*, vol. 9, no. 8, p. 1037, 2010.
- [33] R. Sihwail, K. Omar, and K. A. Z. Ariffin, "A survey on malware analysis techniques: static, dynamic, hybrid and memory analysis," *International Journal of Advanced Science, Engineering and Information Technology*, vol. 8, no. 4-2, p. 1662, 2018.

Research Article

Cryptocurrency Mining Malware Detection Based on Behavior Pattern and Graph Neural Network

Rui Zheng ^{1,2}, Qiuyun Wang ², Jia He ¹, Jianming Fu ¹, Guga Suri ¹,
and Zhengwei Jiang ^{2,3}

¹Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

²Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

³School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Correspondence should be addressed to Jianming Fu; jmfu@whu.edu.cn

Received 17 September 2021; Revised 12 November 2021; Accepted 9 March 2022; Published 26 March 2022

Academic Editor: Chunhua Su

Copyright © 2022 Rui Zheng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Miner malware has been steadily increasing in recent years as the value of cryptocurrency rises, which poses a considerable threat to users' device security. Miner malware has obvious behavior patterns in order to participate in blockchain computing. However, most miner malware detection methods use raw bytes feature and sequential opcode as detection features. It is difficult for these methods to obtain better detection results due to not modeling robust features. In this paper, a miner malware identification method based on graph classification network is designed by analyzing the features of function call graph and control flow graph of miner malware, called MBGINet. MBGINet can model the behavior graph relationship of miner malware by extracting the connection features of critical nodes in the behavior graph. Finally, MBGINet transforms these node features into the feature vectors of the graph for miner malware identification. In the test experiments, datasets with different volumes are used for simulating real-world scenarios. The experimental results show that the MBGINet method achieves a leading and stable performance compared to the dedicated opcode detection method and obtains an accuracy improvement of 3.08% on the simulated in-the-wild dataset. Meanwhile, MBGINet gains an advantage over the general malware detection method Malconv. These experimental results demonstrate the superiority of the MBGINet method, which has excellent characteristics in adapting to realistic scenarios.

1. Introduction

Miner malware, also known as cryptocurrency mining hijacking attacks, performs cryptocurrency mining by stealing computing resources from the victim's computing devices [1]. Miner malware has continued to be active in recent years due to the rising value of cryptocurrencies [2]. The continued activity of miner malware poses a significant security threat to user devices. Miner malware stealing computing resources will consume a large number of electric power resources and reduce the lifetime of the victim device [3]. In addition to that, some miner malware can seriously occupy CPU resources and cause system crash to bring substantial business losses to users [4]. Therefore, the detection of miner malware in the wild becomes an important malware detection issue.

Different from other common malware, miner malware has its distinct characteristics. First of all, miner malware has noticeable blockchain computing features compared to common malware, and it needs to constantly interact with the blockchain during the running of the malware. These characteristics will expose a large number of behavior features of miner malware. In local, cryptocurrency computing requires a large number of computing resources and access to the victim's computing devices. Network interaction and local computing features together composed the cryptocurrency mining behavior features of miner malware. In addition, miner malware has unique ecological characteristics, such as reliance on XMR's open-source software to mine Monero cryptocurrencies, which bring distinct characteristics to the behavior pattern of miner malware.

Traditional malware detection techniques mainly rely on signature matching techniques and heuristic detection, but this type of rule matching has the low capability for generalization and finding unknown malware [5]. Machine learning techniques have evolved in recent years and have become another important way to solve malware detection. Furthermore, deep learning has a clear advantage in the large space search problem [6], being able to accurately predict labels in high-dimensional spaces [7]. However, in the malware detection domain, feature selection is still related to the upper limit of the capability of the final training model. Some malware detection studies used a data-driven type of feature to design malware detectors, such as byte-level features [8] and PE format features [9]. Such a feature design idea runs counter to the idea of the traditional malware analysis community, which often determines the class of malware based on its behavior pattern. In addition, some malware research heavily relies on research methods from the traditional field of machine learning, using random splitting and small datasets as test datasets production standards. Such testing methods do not fit the real-world malware detection circumstance, where malware detectors are trained in small datasets and deployed against the scenario of massive malware samples.

In this paper, we use function call graphs and basic block jump graphs within miner malware binaries as features for miner malware detection. The principle of malware detector based on behavior features follows the malware analysis methods of the malware analysis community, while the behavior features are also more difficult to fabricate. In order to explore the role of the behavior features for miner malware detection, we convert the call relationship and instructions jump graph as the feature vector. This approach can highlight the nodes with distinct connectivity properties in the behavior graph. For the evaluation, the large dataset is used as validation samples, and small dataset samples are used as training samples, which follows the paradigm of malware detection. Finally, the popular detection methods for the miner malware and general malware are taken as baselines to represent the effectiveness of the proposed methods.

The contributions of this paper are as follows:

- (1) This paper explores the usefulness of behavior pattern on function level and basic block level for miner malware classification under the static analysis. By constructing different graph isomorphic networks to build miner malware classifiers, experiments show that the proposed method in this paper outperforms other baseline methods.
- (2) Representative miner malware detection methods and a general malware detection method are chosen for the experiments as baseline methods. The experimental results show that the proposed method achieves advantages over these methods, proving its superiority.
- (3) The experiment uses a simulated in-the-wild dataset as the test environment. The experimental results better reflect the effectiveness of the proposed method in the real world.

This paper systematically described a graph-based method for miner malware detection. Section 2 reviews the research on machine learning-based methods for general and miner malware detection. Section 3 introduces the framework of MBGINet, and Section 4 shows the performance evaluation results of all methods on laboratory dataset and simulated in-the-wild dataset. Section 5 summarizes the work of this paper.

2. Related Work

This section first discusses popular malware detection methods based on static analysis features and machine learning. Then, the research of miner malware is described, which contains miner malware ecological development and their detection methods.

2.1. Malware Detection. Static analysis features are important features for malware detection. Compared with dynamic analysis features, static analysis features have the characteristics of fast extraction and variety. Taking the parsing of the PE file as the criterion, we can divide the static features of PE files into three layers, namely, raw bytes, PE structure, and disassembly code. The complexity of parsing rules used for these three feature layers goes from shallow to deep and parsing time from short to long.

Raw bytes are the most accessible malware feature, which does not require format parsing. Raw bytes can be directly taken as the basis for malware identification, but raw bytes often face the challenge of enormous feature dimensions. Therefore, many dimensionality reduction methods are applied to the feature preprocessing of raw bytes, for example, the frequency method [10], information compression method [11], sampling method [12], etc. With the help of these methods, raw bytes are processed into feature vectors suitable for machine learning models to train malware classifiers. Furthermore, some deep learning models [13, 14] can even directly process the original high-dimensional byte features. However, raw bytes have a notorious drawback in that no clear semantic information can be obtained internally. The lack of interpretable malware determination results makes the robustness of this class of methods questionable.

PE structure information is another feature for malware detection. PE structure information needs to be parsed in a fixed format with byte codes to extract feature information compared to raw bytes. The PE header [15] and sections information [9] are useful for malware detection. The extracted PE meta information and sections information describe the basic parsing information of the malware, which has similarities in different malware. However, this information is easy to fabricate, leading to malware detector failure.

The assembly code needs to be further parsed based on the PE structure parsing. By scanning the code sections of the PE file, the disassembly tool can convert binary code to assembly code, even C language pseudocode. The assembly code obtained in this process can represent the behavior of

the malware, including system-level action features and instruction-level action features involved in the malware execution process. From the perspective of program analysis, these features directly determine the maliciousness determination of malware. Moreover, these features are much less likely to be fabricated. Therefore, disassembly code features are one of the desirable types of features for malware detection.

An important feature extracted through disassembly tools is the opcode, which characterizes the sequence of instructions executed by the CPU when the malware is running. Thus, opcode could be taken as the feature to detect malware. Jeon et al. [16] apply a convolutional autoencoder to extract and compress the opcode feature vector. Then, these feature vectors are fed into a recurrent neural network to train a malware detector. However, although the opcode feature can be helpful for malware classification, its inability to describe invocation relationships between instructions limits its usefulness for malware detection.

Among the features obtained from disassembly, graph features constructed by functions or basic blocks can represent the behavior pattern of malware, and such features are used as the basis for malware classification. Yan et al. [17] extract control flow graph from disassembly analysis as classification features and converted code blocks to feature vector using its instructions statistical features. These features are fed into the graph convolutional network to train a malware detector. In addition to the above methods, graph matching algorithms are also used to identify malware. Ammar et al. [18] extract static function call graphs of malware as feature vectors for malware recognition. Graph matching algorithms are used to calculate the similarity between the suspicious samples and the known samples. The similarity is used to determine whether the suspicious sample is malware.

In the tasks of malware detection based on graph feature extracted by disassembly tool, the connections between elements represent that malware's actions can describe the behavior patterns of malware. Such features are robust compared to the raw bytes feature and PE structure feature. Moreover, behavior pattern recognition is an important way to solve malware detection problems.

2.2. Miner Malware Detection. Due to the great danger of miner malware, many studies have been conducted to analyze miner malware activities. Within this process, many features of miner malware have been revealed. For example, Pastrana et al. [19] study the campaigns of miner malware in a decade. The authors find that the cryptocurrency mined by miner malware is concentrated in several currencies such as Monero and Bitcoin. It is not the only case. Tekiner et al. [3] find miner malware focused on XMR or Bitcoin to mine. In addition, miner malware also needs to perform a large number of cryptographic operations when performing tasks. Some standard anti-antivirus measures are also often used before cryptographic operations [20]. These behaviors indicate that the miner malware has a clear pattern of behavior.

For the detection of miner malware, the feature of network traffic [21] and CPU usage [22] have been taken as indicators for detecting miner malware. In terms of static analysis, the opcode is used as an important feature for miner malware detection. Yazdinejad et al. [23] trained a malware classifier using biLSTM for modeling the sequence characteristics of opcode and achieved good experimental results. Naseem et al. [24] introduced a detection method for browser cryptocurrency mining attacks by converting bytecode files into grayscale images and then training CNN models to identify grayscale image features.

Although opcode and grayscale image features can be used as feature vectors for miner malware identification, it is difficult for these features to represent the pattern of miner malware behavior. Since miner malware has obvious behavior characteristics, this paper designs a graph network classification method called MBGINet based on behavior patterns to detect miner malware.

3. MBGINet Architecture

In this paper, we design a Miner Behavior Graph Isomorphism Network (MBGINet) for miner detection. Compared with traditional opcode feature methods and data-driven methods, this approach enables more efficient detection of miner malware and has more robust performance in the real world. In this section, the design of MBGINet will be described in detail. Firstly, the miner malware behavior pattern at the basic block and function level will be described. Secondly, the MBGINet structure and parameters corresponding to the two levels will be presented.

3.1. Miner Behavior Graph Modeling. In the static analysis setting, the behaviors of malware cannot be executed precisely in chronological order. Although there exists a sequence of instructions execution, it is difficult to describe a perfect sequential action due to branching judgment and other issues. Therefore, a graph network becomes the most suitable model to describe the invocation relationship at levels of functions and basic blocks. From the perspective of the malware analysis community, the calling relationships between disassembly codes are the essential source for malware analysis. In the operation of miner malware, the ultimate purpose of all actions performed is to load the cryptocurrency computing module smoothly. Before calling the cryptocurrency mining module, a series of OS information collection and persistence operations need to be performed. This process does not contain much files operation and data exfiltration, which is different from other types of malware. Therefore, miner malware has obvious behavioral differences compared to ordinary software.

The behavior semantic of miner malware can be represented by functions and instructions. The former is the functions call, which is the concrete execution module for behaviors in Ring 3. The latter is the instructions sequence in the assembly code obtained by parsing the binary code,

which is the CPU-level actions. In these two behavior levels, miner malware implements the corresponding malicious capability through function execution and code block execution, respectively.

Function objects are responsible for the execution of specific behaviors, and the boundary of the function is almost the boundary of the behavior. The invocations between functions form the basis of the binary’s operation. So the calling relationship between functions represents the entire malware behavior implementation. The calling relationship of functions is clear in miner malware, where not only does the miner malware need to perform the cryptocurrency mining behavior, but some discovery and defense evasion behaviors are also critical to the implementation of the malicious behavior. The combination of these behaviors or the alone blockchain computing brings a distinct behavioral signature representation for cryptocurrency mining malware.

The basic code blocks are a group of instructions with jump instructions as the dividing point. Within the basic code block, instructions are executed sequentially. Among the blocks of instructions, they are executed according to jump conditions. Thus, the instruction-level execution behavior reflects the action sequence of the CPU. Furthermore, the code block generally represents a small execution operation, and the jump operation is related to the conditional execution of the software. This execution paradigm describes the basic malware CPU behavior pattern. CPU behavior pattern represents the behavior of cryptocurrency mining and other malicious operators. CPU behavior patterns are suitable for discovering miner malware, as most miner malware relies on the arithmetic power provided by the CPU to perform blockchain computing [3].

We define the graph classification task model for miner malware identification based on the graph features. Given a graph $G = (V, E)$, node $v_i \in V$, v_i represents a function body or a basic code block, and $E = (v_i, v_j)$ represents the direct call or jump relationship of the nodes. Such a graph network consisting of nodes and edges can represent the behavior pattern of a malware binary. Connectivity relations are used to describe the graph isomorphism between miner malware samples, where all nodes are taken as the same vector. We design MBGINet for modeling the behavior pattern of the control flow relationship in miner malware.

3.2. MBGINet Model Structure. In order to extract the connection relationship between nodes, we choose graph isomorphism network [25] as the classification network. Inspired by the design of the GIN model, we construct graph feature aggregation to collect aggregate features to important nodes and convert the nodes vector to graph embedding for classification. Node feature aggregation layer is shown in equation (1). Each layer further aggregates the node’s degree information to the critical nodes and finally converges the structural information of the graph into a vector. In equation (1), k denotes the index of layers and ϵ denotes the merge coefficient between neighborhood nodes and center node, which is assigned a fixed scalar. h_v^k denotes the message passing operator of graph features. Message passing of graph

data in MLP guarantees that the entire layer can be used as an inflective function to extract aggregated features of nodes.

$$h_v^{(k)} = \text{MLP}^{(k)} \left((1 + \epsilon^{(k)}) * h_v^{(k-1)} + \sum_{u \in N(v)} h_u^{(k-1)} \right). \quad (1)$$

In the behavior graph, the most remarkable information is the connectivity of nodes, which are aggregated into a vector to describe the graph structure in the task of malware classification. In the further compression of the nodes feature, we extract the maximum value as an input (as shown in equation (2)) to the next fully connected layer, unlike the original GIN model. The operation of global max-pooling is partly due to the max nodes features, leading to much differentiation between miner malware and others. Moreover, the global max-pooling enables highlighting the information describing the calls or jumps in the behavior pattern of miner malware. Behavior graphs at the basic block level have more nodes and more edges than that at the function level; this implies a more sparse connection pattern at the function level. Therefore, we use global max-pooling for graph isomorphism network at the basic block level and use summation pooling for graph isomorphism network at the function level.

$$\text{inter_layer} = \text{Global Max Pooling}(h_v^{(k)}). \quad (2)$$

Inspired by the GIN model and after parameter tuning, we designed the framework structure of MBGINet. We separately designed different graph neural network structures for the function level and the code basic block level because of the huge difference between their node sizes. As shown in Figure 1, the structure and hyperparameter of MBGINet-FCG are present. MBGINet-FCG uses a hidden layer of nodes aggregator layer and the nodes are extended to 8-dimensional channels in the MLP layer. As mentioned earlier, the network structure is designed to use sum pooling to handle the convergence features due to the sparse connectivity of the function layer.

Aiming to process the structure of the basic block nodes, four hidden layers with Multilayer Perceptron (MLP) are used to extract aggregated features of the nodes. Global max-pooling is used to extract the largest features that are output as graph classification results through the fully connected layer and activation function. The global max-pooling facilitates the extraction of connectivity features between basic block nodes, and thus the feature vector of the entire invoked graph features is fed to the fully connected layer for classification. As shown in Figure 2, MBGINet-CFG contains four hidden layers with aggregation computation and a global max-pooling layer, where k represents the index of the layer and d represents the number of channels of the MLP in the hidden layer. As depicted in Figure 2, the four hidden layers of the model use a recursive structure to output the final results for the global max-pooling layer instead of using the concatenating output of every hidden layer in the original paper. As a result, this allows for better extraction of

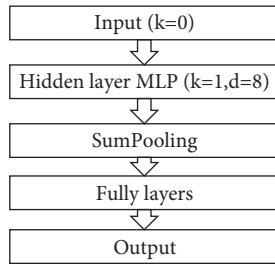


FIGURE 1: The structure of MBGNet at level of function call graph.

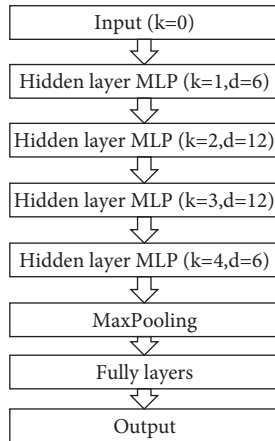


FIGURE 2: The structure of MBGNet at level of control flow graph.

the most significant node connectivity features through multilayers message passing.

We used the MBGNet structure to model two-level graph pattern of miner malware. In the next section, the performance of these two graph neural networks with different feature levels is evaluated in the same experiments framework.

4. Evaluation

To verify the performance of MBGNet in behavior levels of FCG and CFG, we conducted experiments on two datasets, which simulated the developments environment of the miner malware detector in the real world. Through comparing the performance of MBGNet with baseline methods, we found the MBGNet outperforms opcode sequence model [23], grayscale image feature model [24], and the excellent Malconv model [8]. In this section, we first introduce the datasets used for the experiments and the evaluation metrics of the results. Secondly, the performance of the MBGNet and the baseline methods on the laboratory dataset and simulate-wild dataset are present, respectively. Finally, the time consumption of all methods involved in this paper is described.

4.1. Dataset and Experiments Setup. In order to understand the effectiveness of MBGNet from the perspective of practical applications, this paper uses two miner malware datasets with different data volumes as the benchmark. This

is different from some malware detection works [26, 27] that randomly split the training and test samples from one dataset, and the test dataset is always undersize. These two miner malware datasets are published on the malicious code analysis direction of the Big Data Security Analysis Competition *. We call the smaller dataset lab dataset, and call the larger one the simulated in-the-wild (Sim-Wild) dataset.

The number distribution of samples in the lab dataset and the simulated in-the-wild dataset is shown in Table 1. The two datasets consist of miner and not-miner samples, where not-miner one includes other types of malware and benign software. The lab dataset and the simulated in-the-wild dataset have almost the same data distribution ratio; the ratio of miner samples to not-miner samples is about 1 : 2. At the same time, the simulated in-the-wild dataset has about three times the volume of the lab dataset, i.e., 6000 vs. 17,657. Such a data distribution is consistent with developing and deploying machine learning malware detectors in the real world. These malware detectors are always developed on a small dataset and deployed against the massive malware samples in the wild. In this paper, opcode, FCG, and CFG features are extracted using IDA pro †. As shown in Table 1, a small number of samples suffer parsing failure in the feature extraction.

Typical performance metrics are used as evaluation indicators for experiments, such as accuracy, precision, recall, F1-score, and AUC. Since these metrics are usual information retrieval metrics, we do not describe the formulas of these metrics in detail. For the evaluation of time consumption, we compare by calculating the sum of feature extraction, training time, and inference time.

4.2. Performance Comparison on Lab Dataset. In the development of machine learning malware detectors, small datasets are often used for cross-validation experiments. As described previously, we use the lab dataset as a small training and testing dataset to simulate the development of miner malware detectors in a real scenario. The lab dataset is divided into five pieces for performing cross-validation experiments. Four pieces of the dataset are treated as a training dataset, and the remaining one piece is used to test trained model in each experiment. The average of the results of the five experiments is taken as the performance representation. The results of various methods are shown in Table 2.

Table 2 shows the detection results of five machine learning methods, where MBGNet-FCG and MBGNet-CFG denote the effects of MBGNet on two levels of graph features, and the remaining three models are baseline methods. The grayscale image (GI) method is derived from [24], which detects cryptocurrency mining attacks in browsers by converting bytecode files to grayscale images to perform the classification task. The OP denotes a dedicated miner malware detection method [23] that implements classifier construction by using opcode as a feature vector using a bidirectional LSTM method. Malconv [8] is a representative approach for general malware detection. In this experiment, Malconv takes raw bytes of binary files as

TABLE 1: Number of samples in the lab dataset and the Sim-Wild dataset for different features.

Experiments	Lab dataset		Sim-Wild dataset	
	Miner	Not miner	Miner	Not miner
Original	2000	4000	5898	11759
FCG	1885	3773	5167	11120
CFG	1885	3773	5167	11120
Opcode	1885	3699	5823	11163

TABLE 2: Detection performance of machine learning models on lab dataset.

Feature	Accuracy	Precision	Recall	F1-score	AUC
GI [24]	0.9634	0.9802	0.9091	0.9431	0.9499
OP [23]	0.9660	0.9666	0.9321	0.9486	0.9577
Malconv [8]	0.9776	0.9957	0.9371	0.9653	0.9675
MBGINet-FCG	0.9732	0.9921	0.9312	0.9607	0.9636
MBGINet-CFG	0.9803	0.9954	0.9453	0.9697	0.9716

input to build a deep learning model for miner malware detection. These three baseline methods are representative research for miner malware and general malware detection in the static analysis.

Table 2 presents the performances of our proposed method and the baseline methods, where the miner malware detectors trained by the MBGINet method on two-level graph features are denoted as MBGINet-FCG and MBGINet-CFG, respectively. The bold values in Table 2 represent the best performance in this one column. It can be seen that MBGINet-CFG obtains the best results in every performance metric among all the detectors and achieves a significant advantage. For example, MBGINet-CFG achieves a result of 0.9803 in the accuracy metric, a 0.3 percentage point improvement over Malconv that is the best method among the baseline methods. Moreover, for a composite metric of F1-score, MBGINet-CFG achieves a 0.9 percentage point improvement over the Malconv method. On the other hand, MBGINet-FCG also achieves better results relative to the GI and OP methods. This illustrates the effectiveness of the MBGINet. Although the methods listed in Table 2 all obtained good detection results, the weakest GI method also obtained an accuracy of more than 0.96. However, such results do not directly reflect the test results of real-world data environments. To better approximate the performance of the miner detectors in the real world, we test the models' performances using a larger dataset to compare their generalization ability.

4.3. Performance Comparison on Simulated In-The-Wild Dataset. Once a miner detector is deployed in the wild, it often needs to face a large amount of data. This causes the range of feature variations of miner malware to increase significantly due to the artificial nature of malware. In this section, we choose Sim-Wild dataset as the performance test benchmark. Its data size is ten times larger than the test samples of lab dataset. By inspecting various miner detectors' performance on the Sim-Wild dataset and their changes relative to the lab dataset, we present the detection

performance and generalization performance of different miner malware detectors. In the experiments, detectors trained in five cross-validation experiments were used as test subjects to examine corresponding methods performance on the simulated in-the-wild dataset. Moreover, the average of results in the five experiments was used to represent the performance of the tested method. Such a paradigm ensures machine learning methods are tested on the same models and datasets.

As in Table 2, the bold values in Table 3 represent the optimal performance in this column. In Table 3, MBGINet also achieves the leading position. The precision rate of MBGINet-CFG is higher than the second place of Malconv with 1.33 percentage points. Similar to Table 2, MBGINet-CFG still outperforms other methods and MBGINet-FCG is weaker than the Malconv model. Except for the Malconv model, the MBGINet is far superior to the other two specialized miner malware detection baselines. Opcode (OP) method [23] and grayscale image (GI) method [24] suffer a significant drop in all metrics. OP has the largest drop in recall rate, reaching 13.58 percentage points. This shows that both machine learning methods do not learn universal features on the lab dataset, resulting in local optimization. MBGINet-CFG still outperforms the Malconv model in all metrics except the recall rate. MBGINet-CFG has a better precision rate but a weaker recall rate than Malconv, which indicates that MBGINet-CFG captures more essential features of miner malware and obtains a clearer picture of miner malware. The recall loss shows MBGINet-CFG does not cover nonmainstream miner malware behavior. Nevertheless, MBGINet-CFG outperforms Malconv in the other metrics. MBGINet-CFG's two comprehensive metrics are ahead of Malconv by 0.3 percentage points and 0.15 percentage points, respectively. This illustrates that MBGINet-CFG has comprehensive capabilities to detect miner malware.

The results in Tables 2 and 3 show that the MBGINet proposed in this paper has excellent performance on the task of miner malware detection. In the MBGINet, the behavior feature at the CFG level is significantly better than the behavior feature at the FCG level for the classification task. Both the GI features and opcode features achieve poor results for multiple reasons. The GI [24] may be because the browser bytecode file format is simpler than PE file format, so it is difficult for the CNN to model the characteristics of miner malware in PE format. The opcode's poor results may be because its sequential nature is inconsistent with the actual behavior sequence of miner malware, which contains a large number of jump actions.

In the experiments, Malconv obtained stronger performance than MBGINet-FCG. Such results are due to at least two reasons: on the one hand, the FCG features are still

TABLE 3: Detection performance of machine learning models on Sim-Wild dataset.

Feature	Accuracy	Precision	Recall	F1-score	AUC
GI [24]	0.9248	0.9561	0.8121	0.8783	0.8967
OP [23]	0.9146	0.9459	0.7963	0.8647	0.8863
Malconv [8]	0.9503	0.9720	0.8766	0.9218	0.9320
MBGINet-FCG	0.9411	0.9566	0.8533	0.9020	0.9176
MBGINet-CFG	0.9556	0.9853	0.8730	0.9258	0.9335

too sparse with low feature dimensionality, resulting in the behavior pattern of miner malware not being well represented, and on the other hand, Malconv, as a variant of a traditional CNN model, has excellent learning performance and has been proven to have good modeling capabilities for static features of malware [13, 28]. For the MBGINet model, the model performance at the CFG level outperforms that at the FCG level, which is in line with our expectation because the underlying jumps at the basic block level are more diverse and have more call-and-call relationships. This allows the model’s graph isomorphism learning capability to be better exploited.

However, although the method proposed in this paper achieves better results compared to the baseline methods, this paper also has the common drawbacks of static analysis-based malware detection methods, for example, packer obfuscation and other obfuscations against control flow analysis. These techniques can severely disrupt the behavior pattern under static analysis, causing MBGINet and other methods based on static features to fail. Fortunately, according to public reports [19], less than 30% of miner malware applied packer obfuscation, which shows that packer obfuscation has a limited impact on static analysis methods for miner malware detection.

4.4. Timely Consumption. We present the time consumption analysis of miner malware detection methods involved in this paper for feature preprocessing, model training, and sample inference. All experiments in this paper were performed on a computer with a NVIDIA 1080ti GPU and an Intel i7-7700 CPU. The experiments code was written in Python 3.7, and the Pytorch machine learning library with CUDA 10.1 is used as the deep learning framework.

As shown in Table 4, the “Preprocess” column indicates the computation time required for one sample. The “Training stage” column consists of epoch time and the number of epochs required. The time required for one sample inference is displayed in the “Inference stage” column.

From Table 4, it can be seen that our proposed MBGINet method has better time efficiency for both levels of behavior features. Although using the least time, GI feature’s detection performance is poor. Malconv and biLSTM-OP methods need a mass of training time due to the huge number of parameters. In the MBGINet method, the computation speed of FCG features is better than that of CFG, because the structure of MBGINet-FCG applied is

TABLE 4: Time consumption of methods for miner malware detection (in seconds).

Methods	Preprocess	Training stage	Inference stage
GI	0.008	4.9810 * 18	0.00150
OP	3.74	626.08 * 16	0.42089
Malconv	0	205 * 7	0.08858
MBGINet-FCG	3.63	2.68 * 30	0.00067
MBGINet-CFG	3.63	7.8 * 10	0.00326

simpler. Putting the above analysis together, the MBGINet method still has advantages in time efficiency for training and testing, especially compared with the Malconv and OP, where the training speed is greatly reduced.

5. Conclusion

The behavior pattern of miner malware is distinctly different from other malware and benign software due to the apparent CPU computing and auxiliary malicious behavior characteristics. In this paper, a miner malware detection method MBGINet is designed based on behavior pattern recognition. MBGINet is designed with aggregator layers and fully connected layers to model the connectivity characteristics between nodes in function call graph or control flow graph. For FCG and CFG features, MBGINet has different network structures to better model node convergence structures. Experimental verification shows the MBGINet can accurately describe the behavior pattern of miner malware compared to the opcode feature [23]. Although MBGINet uses similar static behavioral features as opcode, it achieves a 3.08% accuracy improvement over the opcode method. Moreover, experiment results show that the MBGINet method achieves 4.1% accuracy gains relative to the grayscale image feature [12], which is used to detect browser cryptohijack byte files [24]. Besides, MBGINet-CFG is also better than the popular malware detection method of Malconv [8] in the comprehensive metrics. These experimental results show that the MBGINet is an effective behavioral pattern analysis method for miner malware detection in static analysis. Experiments under simulated in-the-wild scenarios also demonstrate the performance stability of MBGINet, indicating that MBGINet is more practical than other baseline methods.

Data Availability

The datasets used in this paper are publicly available and fully desensitized, without any privacy or security risks. The public datasets are available at <https://datacon.qianxin.com/opendata/maliciouscode>. Note: Following the policy of data provider, these datasets are currently open for research needs/teaching needs. Therefore, it is necessary to submit a request file with information about the research organization to the owner of the datasets copyright. For more information, please refer to the abovementioned webpage.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

First of all, the authors would like to thank QI-ANXIN Technology Group Inc. for organizing the Big Data Security Analysis Contest 2020 (DataCon2020) and sharing the cryptocurrency mining malware dataset. This work was supported by the National Natural Science Foundation of China (Grant nos. 61972297, 62172308, and 62172144) and National Key Research and Development Program of China (Grant no. 2018YFB0805005).

References

- [1] Randi Eitzman BWJK Kimberly Goody, "How the rise of cryptocurrencies is shaping the cyber crime landscape: the growth of miners," 2021, <https://www.fireeye.com/blog/threat-research/2018/07/cryptocurrencies-cyber-crime-growth-of-miners.html>.
- [2] McAfee, "McAfee labs threats report," 2021, <https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-apr-2021.pdf>.
- [3] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirida, and A. A. Selcuk, "Sok: cryptojacking malware," 2021, <https://arxiv.org/abs/2103.03851>.
- [4] "360TS. Cryptominer, winstarnssminer, has made a fortune by brutally hijacking computers," 2021, <https://blog.360totalsecurity.com/en/cryptominer-winstarnssminer-made-fortune-brutally-hijacking-computer/>.
- [5] R. A. Bridges, S. Oesch, M. E. Verma et al., "Beyond the hype: a real-world evaluation of the impact and cost of machine learning-based malware detection," 2020, <https://arxiv.org/abs/2012.09214>.
- [6] Q. Dong, C. Jia, F. Duan, and D. Wang, "RLS-PSM: a robust and accurate password strength meter based on reuse, Leet and separation," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4988–5002, 2021.
- [7] W. Melicher, B. Ur, S. M. Segreti et al., "Fast, lean, and accurate: modeling password guessability using neural networks," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, pp. 175–191, Austin, TX, USA, August 2016.
- [8] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. Nicholas, "Malware detection by eating a whole exe," 2017, <https://arxiv.org/abs/1710.09435>.
- [9] H. S. Anderson and P. Roth, "Ember: an open dataset for training static pe malware machine learning models," 2018, <https://arxiv.org/abs/1804.04637>.
- [10] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-KDD'04*, Seattle, WA, USA, August 2004.
- [11] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proceedings of the 2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 11–20, IEEE, Fajardo, PR, USA, October 2015.
- [12] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security-VizSec'11*, pp. 1–7, Pittsburgh, PA, USA, July 2011.
- [13] S. E. Coull and C. Gardner, "Activation analysis of a byte-based deep neural network for malware classification," in *Proceedings of the 2019 IEEE Security and Privacy Workshops (SPW)*, pp. 21–27, IEEE, San Francisco, CA, USA, May 2019.
- [14] E. Raff, W. Fleschman, R. Zak, H. S. Anderson, B. Filar, and M. McLean, "Classifying sequences of extreme length with constant memory applied to malware detection," 2020, <https://arxiv.org/abs/2012.09390>.
- [15] R. Vyas, X. Luo, N. McFarland, and C. Justice, "Investigation of malicious portable executable file detection on the network using supervised learning techniques," in *Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 941–946, IEEE, Lisbon, Portugal, May 2017.
- [16] S. Jeon and J. Moon, "Malware-detection method with a convolutional recurrent neural network using opcode sequences," *Information Sciences*, vol. 535, pp. 1–15, 2020.
- [17] J. Yan, G. Yan, and D. Jin, "Classifying malware represented as control flow graphs using deep graph convolutional neural network," in *Proceedings of the 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019*, pp. 52–63, Portland, OR, USA, June 2019.
- [18] A. A. E. Elhadi, M. A. Maarof, B. I. A. Barry, and H. Hamza, "Enhancing the detection of metamorphic malware using call graphs," *Computers & Security*, vol. 46, pp. 62–78, 2014.
- [19] S. Pastrana and G. Suarez-Tangil, "A first look at the cryptomining malware ecosystem," in *Proceedings of the Internet Measurement Conference*, pp. 73–86, Amsterdam, Netherlands, October 2019.
- [20] A. Zimba, Z. Wang, M. Mulenga, and N. H. Odongo, "Cryptomining attacks in information systems: an emerging threat to cyber security," *Journal of Computer Information Systems*, vol. 60, no. 4, pp. 297–308, 2020.
- [21] A. Pastor, A. Mozo, S. Vakaruk et al., "Detection of encrypted cryptomining malware connections with machine and deep learning," *IEEE Access*, vol. 8, pp. 158036–158055, 2020.
- [22] F. Gomes and M. Correia, "Cryptojacking detection with cpu usage metrics," in *Proceedings of the 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pp. 1–10, IEEE, Cambridge, MA, USA, November 2020.
- [23] A. Yazdinejad, H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, G. Srivastava, and M.-Y. Chen, "Cryptocurrency malware hunting: a deep recurrent neural network approach," *Applied Soft Computing*, vol. 96, Article ID 106.630, 2020.
- [24] F. Naseem, A. Aris, L. Babun, E. Tekiner, and A. S. Uluagac, "MINOS: a lightweight real-time cryptojacking detection system," in *Proceedings of the 2021 Network and Distributed System Security Symposium*, pp. 21–25, San Diego, CA, USA, November 2021.
- [25] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," 2018, <https://arxiv.org/abs/1810.00826>.
- [26] S. Yue, "Imbalanced malware images classification: a CNN based approach," 2017, <https://arxiv.org/abs/1708.08042>.
- [27] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences*, vol. 460–461, pp. 83–102, 2018.
- [28] S. Bose, T. Barao, and X. Liu, "Explaining ai for malware detection: analysis of mechanisms of malconv," in *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, Glasgow, UK, July 2020.

Research Article

An Efficient Outlier Detection Approach for Streaming Sensor Data Based on Neighbor Difference and Clustering

Saihua Cai ^{1,2}, Jinfu Chen ¹, Baoquan Yin,³ Ruizhi Sun,⁴ Chi Zhang,¹ Haibo Chen,¹ Jingyi Chen,¹ and Min Lin¹

¹School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang 212013, China

²Jiangsu Key Laboratory of Security Technology for Industrial Cyberspace, Jiangsu University, Zhenjiang 212013, China

³Yantai Academy of China Agricultural University, Yantai 264670, China

⁴College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China

Correspondence should be addressed to Jinfu Chen; jinfuchen@ujs.edu.cn

Received 14 September 2021; Revised 15 January 2022; Accepted 29 January 2022; Published 27 February 2022

Academic Editor: Shifeng Sun

Copyright © 2022 Saihua Cai et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In wireless sensor networks (WSNs), the widely distributed sensors make the real-time processing of data face severe challenges, which prompts the use of edge computing. However, some problems that occur during the operation of sensors will cause unreliability of the collected data, which can result in inaccurate results of edge computing-based processing; thus, it is necessary to detect potential abnormal data (also known as outliers) in the sensor data to ensure their quality. Although the clustering-based outlier detection approaches can detect outliers from the static data, the feature of streaming sensor data requires the detection operation in a one-pass fashion; in addition, the clustering-based approaches also do not consider the time correlation among the streaming sensor data, which leads to its low detection accuracy. To solve these problems, we propose an efficient outlier detection approach based on neighbor difference and clustering, namely, ODNDC, which not only quickly and accurately detects outliers but also identifies the source of outliers in the streaming sensor data. Experiments on a synthetic dataset and a real dataset show that the proposed ODNDC approach achieves great performance in detecting outliers and identifying their sources, as well as the low time consumption.

1. Introduction

In recent years, wireless sensor networks (WSNs) have been widely used in a variety of applications, such as object positioning [1], health management [2], industrial safety control [3], and social media [4–6]. WSNs usually consist of lots of low-cost sensor nodes distributed over a wide area, which leads to the data in WSNs being generated in the form of streams. That is, the data in WSNs are arriving instantly and continuously, and the generating speed of the data is very quick. The use of a large number of sensors makes the real-time processing of data face more severe challenges, while edge computing [7, 8] provides flexible and on-demand processing power to quickly process the data in WSNs. Nevertheless, the limitations in terms of memory, communication bandwidth, battery power, and computational capacity existing in sensor nodes cause the unreliability of collected data, which further results in the

inaccurate results of edge computing-based processing. Therefore, it is necessary to perform some measures to guarantee the quality of collected data.

As an efficient technology to ensure accurate and reliable data for edge computing, outlier detection [9–11] (also known as anomaly detection [12, 13]) plays a critical role in data security; it detects those data in each edge of WSNs that deviate from the rest based on a certain measured approach of data pattern [14, 15], thereby providing high-quality data for edge computing. In addition, to detect abnormal data, outlier detection can also be used to discover some abnormal events [16]. Because of the importance of data security in all walks of life, massive outlier detection approaches have been proposed and used in many applications, such as intrusion detection [17, 18], health diagnosis [19], and social network detection [20, 21]. However, most outlier detection approaches require the collected data to be scanned several times, but the characteristics of streaming

sensor data do not allow the multiple scans of data since the time, cost, and computational complexity are very high [22], which leads to these approaches not being effectively used in WSNs.

Compared with distance-based [10, 23], density-based [24], clustering-based [25, 26] outlier detection approaches, clustering-based approaches are often used in WSNs for the advantages of simple, efficient, low time complexity, and low space complexity. However, the clustering-based outlier detection approaches have some issues as follows: (1) easy to determine normal data as an outlier when the size of the sliding window is not suitable; (2) cannot effectively process high-dimensional data for the calculation of distance or density, as it is very time-consuming; (3) the detection accuracy is easily affected by noise or outliers. In addition to the above problems, clustering-based approaches usually consider the spatial correlation between data to determine the outliers but rarely consider the time correlation of the data, while the data in WSNs have a typical time correlation. Furthermore, the sources of outliers that occurred in WSNs could be caused by errors, events, or malicious attacks [27], and different types of outliers can result in different influences; therefore, accurately analyzing the sources of outliers can help to provide some hints about how to handle the detected outliers. For example, if the detected outlier is not an event, it should be removed from the streaming sensor data to ensure the high quality; otherwise, it could be caused by some unexpected reasons and must be paid attention to, such as fire [28]. Unfortunately, most existing clustering-based outlier detection approaches cannot effectively distinguish the sources of outliers [29, 30].

To solve the problems existing in clustering-based outlier detection approaches, with the use of w-k-means algorithm (which assigns different weights to each attribute according to the correlation between attributes to reduce the impact of irrelevant attributes on the clustering results and thus solving the problem of other k-means algorithms for their high false alarms caused by the irrelevant data), this paper proposes an efficient outlier detection approach based on neighbor difference and clustering, namely, ODNDC, to accurately detect the outliers and identify the source of outliers in the streaming sensor data collected from WSNs. The major contributions of this work are concluded as follows:

- (1) With the use of neighbor difference, we propose an efficient outlier detection algorithm to accurately detect potential outliers with the consideration of time correlation of the data, and then use the w-k-means algorithm with the consideration of correlation between each attribute to reduce the impact of irrelevant attributes on the clustering results, thereby improving the detecting performance.
- (2) With the consideration of spatial correlation, we propose an efficient outlier source identification algorithm to improve the identifying accuracy.
- (3) Based on a synthetic dataset and a real dataset, we conduct extensive experiments to evaluate the efficiency of the ODNDC approach, and the experimental result verifies that the proposed ODNDC approach can accurately detect potential outliers

from streaming sensor data and identify the sources of outliers as well as cost in a short time.

The remaining of this paper can be organized as follows. Section 2 presents the related works on outlier detection approaches for WSNs and clustering-based outlier detection approaches. Section 3 gives some definitions and presents the classification of outliers. Section 4 first describes the framework of the proposed approach, and then presents the details of the proposed approach. Section 5 demonstrates our experimental results. Finally, we conclude our paper with an outline of future work in Section 6.

2. Related Works

In this section, we first review the related outlier detection approaches for WSNs, and then review the clustering-based outlier detection approaches.

2.1. Outlier Detection Approaches for WSNs. The outlier detection is an essential and challenging task, which aims to identify the data instances that significantly differ from other observations, while outlier detection for WSNs is more difficult because data distribution may change accompanied with the environment. To accurately detect the outliers existing in WSNs, researchers proposed many efficient outlier detection approaches in recent years. Exploiting the time-series analysis, Zhang et al. [31] defined the normal behaviors based on the spatial and temporal correlations of the data, and then proposed an efficient temporal and spatial real data-based outlier detection technique (called TSOD) to identify the outliers in WSNs. However, the communication overhead and spatial consumption of TSOD are very high. Around the concept of edge computing, Bharti et al. [32] proposed an in-network contextual outlier detection on edge (called INCODE) to detect the contextual outliers and estimate the abnormal degree; the experimental result verified by the INCODE can accurately detect the outliers as well as minimize the WSNs resource consumption. Poornima and Paramasivan [33] first reduced the dimensionalities online with the principal component analysis to handle the irrelevant and redundant data, and then proposed an online locally weighted projection regression (called OLWPR) to identify the outliers in WSNs; however, the AUC of OLWPR is not ideal. By using the probabilistic inference to adapt the Bayesian networks distributed over wireless sensor nodes, De Paola et al. [34] proposed an adaptive distributed outlier detection approach (called ADOD) to detect outliers in the data collected in WSNs with a high detection accuracy. In addition to ADOD, the local outlier detection algorithm (called LODA) [35] was also based on the adaptive Bayesian network; it detected the potential outliers through time-series modeling on each sensor node locally without collaboration with neighbors. Although the detection efficiency of ADOD and LODA is obviously improved, their time complexity and communication complexity are still very high.

2.2. Clustering-Based Outlier Detection Approaches. In the clustering-based outlier detection approaches [36, 37], it is necessary to define and calculate the distance or similarity metric between two data instances; then, based on the metric, the data instances that are far away from their closest cluster centroid or where their density is below a threshold are declared as outliers. The k-means is one of the most well-known clustering-based algorithms; it has been widely used in outlier detection since its simplicity and efficiency. Elahi et al. [38] presented a strategy of outlier detection in streaming data: as a data instance is detected as abnormal at the first time, it would be declared as a candidate rather than an outlier; and once the times that a candidate outlier is declared abnormal exceeds a threshold in the fixed time, it would be declared as a real outlier; otherwise, it would be regarded as normal. Furthermore, as the dimension of data becomes higher, the space of data becomes sparser, which results in every point in high dimensional space becoming an almost equally good outlier. As a result, the definition of distance becomes meaningless and the clustering-based approaches will fail, known as “curse of dimensionality.” To reduce the number of attributes, many approaches have been proposed. One common way is to execute the feature extraction before clustering [39]; another way is to use weighted k-means algorithm to reduce the effect of irrelevant attributes and noisy attributes by weighting attributes based on their relevance [40]. Common clustering-based approaches use spatial relationship among data to detect outliers regardless of the temporal relationship among data. But, in reality, there are several typical temporal relationships among streaming sensor data. Therefore, it is necessary to detect outliers with the consideration of temporal relationships between the data collected from WSNs, thereby providing accurate and reliable data for edge computing.

Compared with these existing outlier detection approaches for WSNs and clustering-based outlier detection approaches, the proposed approach innovatively uses neighbor difference to detect candidate outliers before performing clustering operations to eliminate the influence of outliers on clustering efficiency, as well as solves the problem of clustering-based approaches not considering time correlation of data. In addition, it also uses spatial correlation of data generated by distributed adjacent sensor to effectively identify the source of outliers, thereby improving the identifying efficiency.

3. Preliminaries

In this section, we provide the necessary background and definitions.

3.1. Sources of Outliers. Outliers may be caused by measurement errors, system errors, or inherent characteristics of the data. Therefore, the handling measures for outliers from different sources are different. The sources of outliers in WSNs can be classified into errors, events, and malicious attacks [27]. Considering that there are fewer outliers caused by malicious attacks in WSNs, and this category of outliers is

not concerned with data, our work mainly focuses on errors and events.

An error normally represents a large change in the reading of a sensor or a large deviation between the data and other surrounding data samples [41]. In WSNs, errors usually originate from unstable or inaccurate sensor readings caused by sensor noise, time drift, human error, etc., or originate from the frame loss or transient network transmission errors caused by weak communication signals. Because the handling of outliers caused by errors will not cause the loss of important information, these outliers can be discarded directly.

An event normally originates from the sudden changes in real-world conditions [41]—rainfall, fire, etc. The sources of events in WSNs can be further divided into internal events or external events. Internal events are usually caused by battery exhaustion, sensor failure, communication interruption, etc., while external events are usually caused by forest fires, air pollution, or environmental changes. The outliers caused by events should be paid sufficient attention, and simply discarding of these outliers may lead to the loss of important information.

In general, the outliers caused by errors appear more frequently than that caused by events, and the probability of outliers caused by errors is more frequent.

3.2. Classification of Outliers. The classification of outliers is very important for the identification of outliers because the sources of outliers are usually caused by different types of outliers. In general, after the outliers are detected by the detection algorithm, domain experts need to analyze and explain the cause and source of outliers, and then the user finally determines whether the suspicious data is a real outlier, while outlier detection algorithm only presents suspicious data to the user from the perspective of data distribution. According to literature [41], outliers can be classified into point outliers, contextual outliers, and collective outliers, but this classification cannot accurately distinguish the sources of outliers in WSNs. To improve the identification accuracy, we assume that the phenomena in real world represented by sensor data is continuous and with a sort of regularity. With this assumption and the characteristics of sensor data, we classify the outliers of sensor data in WSNs into following four types rather than three types.

3.2.1. Point Outliers. If an individual data instance is diverged from normal pattern of the sensor data, it is termed as a point outlier (e.g., t_1 in Figure 1). Point outliers are the simplest type of outliers, and the detection of point outliers is the base of other outlier detections. In general, point outliers are usually caused by errors.

3.2.2. Collective Outliers. If a group of linked data instances are different from entire patterns in the dataset, these data instances are termed as collective outliers (e.g., t_2 in Figure 1).

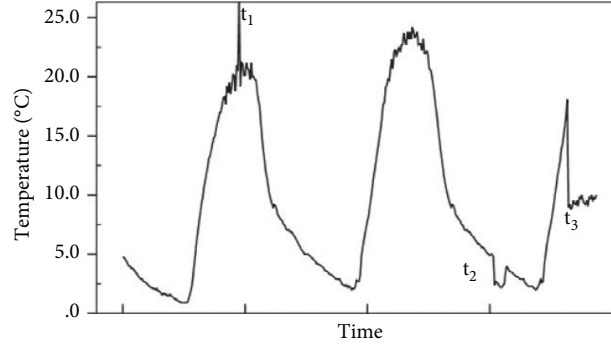


FIGURE 1: A case for demonstrating types of outliers.

3.2.3. Jump Outliers. If a data instance has much difference from its previous data at a specific moment and the subsequent data instances vary within a small range, these data instances are termed as jump outliers (e.g., t_3 in Figure 1).

The way of distinguishing collective outliers and jump outliers can be performed by determining whether the number of data samples after the jump exceeds a preset threshold. In other words, the collective outliers mean that the sensor data will revert to the previous mode after a short period when the outliers appear, while the jump outliers mean that the data will reconstruct a new distribution model after the appearance of the outlier and no longer return to the previous distribution mode. The causes of collective outliers and jump outliers are different. In general, the collective outliers may come from multiple continuous measurement errors of the sensor or an occasional short-term event, and the jump outliers usually come from an event rather than an error.

3.2.4. Contextual Outliers. Unlike the above three types of outliers, there is no obvious difference between contextual outliers and normal data. In general, contextual outliers are abnormal data in a specific context, which are also termed as conditional outliers.

4. Proposed Approach

Based on the above definitions and the sources of outliers, we proposed an efficient outlier detection approach based on neighbor difference and clustering, namely ODND, which can accurately detect the outliers and identify the sources of outliers. Overall, the primary features of ODND can be concluded as follows: (1) can detect outliers in an unsupervised way, (2) can detect outliers through one scan of the collected streaming sensor data, (3) can handle concept evolution, (4) has low time complexity, and (5) can identify the source of outliers. And then, the proposed ODND approach is described in detail.

4.1. The Framework of ODND Approach. In order to realize the detection and identification of outliers, drawing on the idea of CluStream algorithm [42], we divide the framework of ODND approach into online detection phase and offline identification phase, and the specific framework of ODND

approach is shown in Figure 2. The online detection phase aims to realize the real-time detection of outliers in the streaming sensor data, which requires the algorithm to have a faster running speed to catch up with the generating of streaming sensor data. The offline phase aims to identify the source of outliers detected at the online detection phase. Since the amounts of outliers are much smaller than that of normal data, it is not necessary to consider more the time consumption in offline identification phase.

The ODND approach contains three algorithms, including outlier detection based on neighbor difference (ODND), outlier detection based on clustering (ODC), and outlier sources identification based on correlation (OSIC). The ODND algorithm and ODC algorithm belong to online detection phase and they are used to detect the potential outliers from the streaming sensor data; these two algorithms use neighbor difference to each streaming sensor data on the base of the clustering algorithm to solve the problem of traditional clustering-based outlier detection approach that does not consider the time correlation of the data. The OSIC algorithm belongs to offline identification phase, and it is used to identify the sources of outliers.

4.2. ODND Algorithm. The clustering-based outlier detection approaches are very sensitive to noise and outliers, which results in their detection accuracy not being stable when processing streaming sensor data. In order to eliminate the influence caused by noise and outliers, with the consideration of time correlation in the streaming sensor data, we propose an outlier detection algorithm based on the neighbor difference, namely ODND; it detects the outliers before performing clustering-based outlier detection algorithm. The proposed ODND algorithm not only solves the problem of the clustering-based outlier detection algorithm not considering the time correlation but also eliminates the point outliers, thereby improving the performance of subsequent ODC algorithm.

In the ODND algorithm, the neighbor difference between two adjacent data instances in the streaming sensor data is calculated, and the neighbor difference of sensor data (shown in in Figure 1) is the lower curve in Figure 3. It can be seen from Figure 3 that once outliers appear, the neighbor difference values will have significant changes, and different types of outliers have their own specific features, that is, with

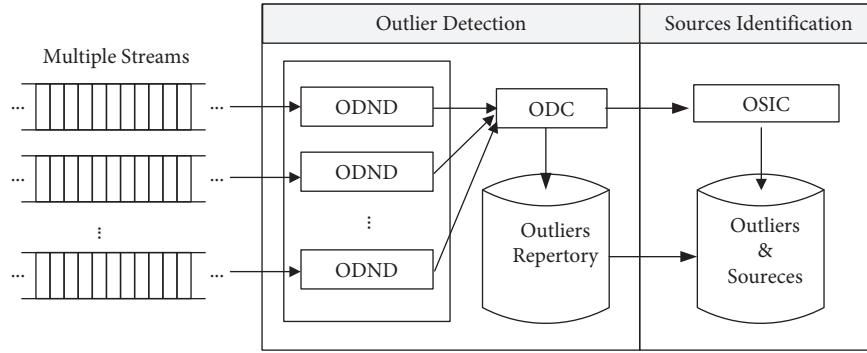


FIGURE 2: The framework of ODND approach.

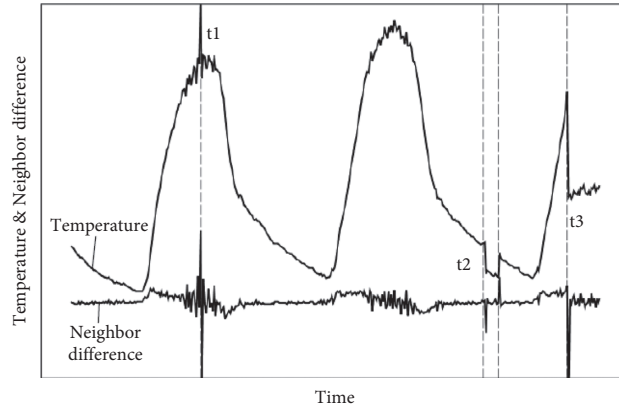


FIGURE 3: Contrast between sensor data and its neighbor difference, where the sensor data source is from temperature values in Figure 1.

a different neighbor difference value. When a point outlier is appearing (t_1 in Figure 3), the neighbor difference value has consecutive significant changes with opposite signs. When a collective outlier appears (the interval of t_2 in Figure 3), there are several small differences in the adjacent difference value with obvious opposite signs, and the number of these small differences is equal to the number of collective outliers according to its definition. Different from collective outliers, when the number of small differences exceeds a given threshold, the outliers are jump outliers (t_3 in Figure 3). Note that the set of threshold can directly influence the final result of the ODND algorithm. In the WSNs, the sensor data meet the normal distribution in general, so the outliers are mostly distributed in the areas outside 3-standard deviation (marked as 3δ) of average value (marked as μ); therefore, we adopt 3δ -principle [43] in the setting of the threshold, where the threshold is set to $(\mu - 3\delta \leq \text{threshold} \leq \mu + 3\delta)$ of the collected streaming data.

According to the above analysis, the ODND algorithm can use the change of neighbor difference to distinguish point outliers, collective outliers, and jump outliers, but it cannot distinguish contextual points (the contextual points will be detected by ODC algorithm). To reduce the false alarms, the detected collective outliers and jump outliers will be regarded as candidate outliers rather than true outliers, and these two types of outliers will be determined as true outliers when they are also declared as outliers in the ODC algorithm.

Before presenting the specifics of the ODND algorithm, we give some definitions of variables:

- (i) curDiff: the difference between current sensor data and previous one
- (ii) abDiff: the abnormal difference detected for the first time
- (iii) count: a counter used to distinguish collective outliers and jump outliers
- (iv) isAbnormal: a flag representing whether the data are being marked as outliers

The specific operation of the proposed ODND algorithm is shown in Algorithm 1.

Furthermore, we use different processing ways for different types of outliers to keep the original distribution of data as much as possible. Normally, the point outliers will be regarded as true outliers and each point's value will be replaced by mean value of its adjacent points. In contrast, the values of other types of outliers will not be modified in the ODND algorithm.

4.3. ODC Algorithm. The proposed ODND algorithm has the advantages of being simple, effective, and fast, and it can improve the performance of the ODC algorithm. However, it only can be used for detecting potential outliers from single streaming sensor data and cannot

```

Input: streaming sensor data, threshold
Output: outliers and their types
(01) calculate the curDiff
(02) if isAbnormal = true then
(03)   count = count + 1
(04) end if
(05) if curDiff outside threshold then
(06)   if isAbnormal = true the
(07)     if curAbnormal and abDiff has opposite sign then
(08)       if count = 1 then
(09)         current data are labeled as true point outlier
(10)       else
(11)         if count in threshold then
(12)           all data between current data and the data corresponding to abDiff are labeled as candidate collective outliers
(13)         else if count outside threshold then
(14)           the data corresponding to abDiff is labeled as a candidate jump outlier
(15)         end if
(16)       end if
(17)       reset the temporary variables
(18)     end if
(19)   else
(20)     if count in threshold then
(21)       label the data corresponding to abDiff as a true jump outlier
(22)       reset the temporary variables
(23)     end if
(24)   end if
(25) end if

```

ALGORITHM 1: Outlier detection based on neighbor difference (ODND).

```

Input: the data labeled by ODND
Output: outliers and their types
(01) use a sliding window model to receive data from all sensors
(02) for each point outlier detected by ODND do
(03)   replace them by their mean values calculated with adjacent data instances
(04) end for
(05) detect outliers using w-k-means algorithm
(06) Reassign the labels of all outliers detected by ODND and w-k-means algorithm

```

ALGORITHM 2: Outlier Detection based on Clustering (ODC).

detect contextual outliers. To effectively detect the outliers in multiple streaming sensor data and detect contextual outliers, an outlier detection algorithm based on clustering, namely ODC, is added after ODND algorithm. In the WSNs, there is no obvious correlation between most streaming sensor data. However, the existence of irrelevant data interferes and conceals the true clustering structure of k-means algorithm, which reduces the efficiency of outlier detection and leads to high false alarms. Different from k-means clustering algorithm, w-k-means algorithm [40] assigns different weights to each attribute according to the correlation between attributes, which can reduce the impact of irrelevant attributes on the clustering results and thus improving the detection performance. Therefore, the w-k-means algorithm is used in ODC algorithm to improve the detection efficiency to a greater extent.

The proposed ODC algorithm contains three blocks, including updating block, clustering block, and outlier detection block. (1) In the update block, k samples are randomly selected as initial cluster center firstly and each object in the data block is assigned to the nearest cluster. And then, the objects in small clusters are regarded as candidate outliers. In order to avoid the influence of outliers on the clustering effect, candidate outliers are not used but the remaining objects are used to calculate the cluster centers and weights. This processing can effectively solve the evolution problem of streaming sensor data. (2) In the clustering block, the w-k-means algorithm [40] is adopted, which reduces the influence of noise attributes on the clustering performance by calculating and assigning a weight to each attribute. (3) In the outlier detection block, the outlier score of objects that are identified as candidate outliers for the first time is set to 1, and then enter to the next

cycle to determine whether this candidate outlier would be detected as an outlier. If the candidate outlier is detected as an outlier again, the outlier score is updated. Within the set period, when the outlier score of a candidate outlier is equal to L , it will be regarded as a true outlier; otherwise, it will be regarded as a normal sample.

The specific operation of the proposed ODC algorithm is shown in Algorithm 2.

The difference between ODC and ODND algorithms is that ODND uses time correlation between the historical data to realize outlier detection of single streaming sensor data, which runs fast but does not consider the spatial correlation between multiple sensors; The ODC algorithm is based on a w - k -means clustering algorithm and uses the spatial correlation between multiple sensors to realize outlier detection. The difference between ODC algorithm and w - k -means clustering algorithm is that ODC algorithm not only needs to detect outliers based on w - k -means clustering algorithm but also needs to jointly determine the outliers detected in the ODND algorithm.

The rules for re-identification of outliers are as follows:

- (1) The point outlier detected in the ODND algorithm usually means that the point is different from isolated data points around it. It is usually caused by operating errors or instant errors of sensor, thus, the point outlier marked by the ODND algorithm is still maintained in the ODC algorithm, and the point outlier is processed before the ODC algorithm, while the processing method is usually to take the average or median value of adjacent data.
- (2) The collective outlier detected in the ODND algorithm may be caused by the sensor error or physical quantities in the objective world in which appear short-term temporary changes, and it is unable to determine the cause of this type of outliers on a single sensor. In order to reduce the false alarms, the collective outliers need to be cleverly detected by the ODND algorithm combined with other sensor variables. Therefore, if most of the data in the ODND algorithm marked as continuous collective outliers are still marked as outliers in the ODC algorithm, then, these collective outliers are determined as true outlier; otherwise, they are considered as normal data.
- (3) The data that are detected as outliers by the ODC algorithm but detected as normal data by the ODND algorithm usually mean that the data on a single sensor is not significantly different, but in the data appear changes when the data of other sensors is considered comprehensively. Therefore, this type of outliers is determined as contextual outliers in the ODC algorithm.
- (4) The jump outlier detected in the ODND algorithm usually means the data generated by a certain sensor have a sudden change as well as other sensors. If the candidate jump outliers detected in the ODND algorithm are not regarded as outliers in the ODC algorithm, the candidate outliers are marked as jump outliers; otherwise, they will be marked as collective outliers.

4.4. OSIC Algorithm. In the ODND approach, the sources of detected outliers will be identified offline using an outlier sources identification algorithm based on correlation, namely OSIC.

In the WSNs, multiple types of sensor nodes are usually deployed at a high density in a certain space to achieve the coverage. Due to high-density distribution of sensor nodes, the environment perceived by sensor nodes deployed in adjacent locations of the space is also similar, which results in the data having a certain spatial correlation. Therefore, the recognition of events can be effectively realized with the use of mutual cooperation of spatially correlated sensor nodes, where the rules for identifying the sources of outliers are defined as follows:

- (1) The source of point outliers is usually marked as an error. Because the point outlier is a data point isolated from other surrounding samples, and it is deviating much from data samples surrounding it, the point outlier is usually caused by operating errors or sensor noise.
- (2) The source of jump outliers is usually marked as an event. The occurrence of a jump outlier usually means that the physical quantity of the objective world appears as a real change, and it also may be caused by a long-term failure of sensors. These two situations require user's attention. The former requires the user to use other domain knowledge to further determine what is happening at the monitoring site, and the latter requires the user to further determine whether the sensor needs maintenance. The long-term sensor failure is regarded as an event in the OSIC algorithm. On the contrary, the short-term sensor failure can be quickly recovered by itself and usually without special treatment, as it is regarded as an error in OSIC algorithm.
- (3) The source of collective outlier may be an error or an event, which requires further judgment. In the WSNs, sensors usually adopt a redundant design to adapt complex environment and improve its stability and reliability, that is, the measurement of the same physical quantity in the same area usually uses multiple sensors with the same type. Because the occurrences of collective outliers can be caused by either errors or events, it is necessary to identify them by significant correlations among sensors variables. There are two categories of significant correlations [44]: (1) One is the spatial neighbor sensors with the same type. For example, the temperature variable of adjacent sensors is usually very similar. If the streaming data of one sensor changes and is quite different from other sensors, this sensor may have functional failures, and it should be marked as an error. If the error remains longer than a given time, then it should be marked as an event. In this manner, the sensor with long-time error should be maintained, while the sensor with short-time error will recover soon automatically. If all streaming data of

temperature variables of adjacent sensors change a lot, an event may occur. (2) The other category of significant correlate sensor variables is several spatial neighbor sensors with different types. For example, the readings of several temperature sensors and several humidity sensors within a certain geographical space have a good correlation. When the temperature in the real world rises, the humidity usually falls correspondingly, and vice versa. Compared with the second category of correlation, the first correlation is easier to judge the cause of outliers. Therefore, in the OSIC algorithm, the first selection for identifying the source of outliers is the first category of correlation.

The criterion for distinguishing the source of outliers is the correlation coefficient of sensor variables. In the ODNDC approach, if the value of correlation coefficient exceeds a threshold of th_1 , it means that all sensors belong to neighbor nodes with the same type; then, a majority voting method is adopted to identify the sources of outliers. If the absolute value of correlation coefficient exceeds a threshold of th_2 , we use a predictive algorithm to predict the mean and the variance of normal data. If a data instance exceeds the range, it will be declared as an error; otherwise, it will be declared as a suspected event, which means that an event may have occurred. In the OSIC algorithm, a support vector regression algorithm with 10-folder cross validation is used to predict the mean and variance.

Note that the choice of thresholds th_1 and th_2 can be done in two ways. One way is to manually set the fix values of th_1 and th_2 , which are based on experts' advice and the implementation of sensors in real life. The advantage of this method is very simple and effective, while the disadvantage is that the accuracy is quite sensitive to preset values. For example, if the thresholds of th_1 and th_2 have been set improperly, the result of this algorithm is inaccurate. Another way is to choose the correlation value of sensors that are in top rank as th_1 and th_2 to automatically determine them. However, the second method is inapplicable if the number of correlation sensors is very less. Therefore, the thresholds of th_1 and th_2 should be chosen based on the actual situation of the sensor deployment.

The design of the OSIC algorithm is shown in Algorithm 3.

5. Experimental Results

In the experiments, we use two datasets to evaluate the efficiency of the proposed ODNDC approach.

5.1. The Description of Datasets. The first dataset is a synthetic dataset, as shown in Figure 1. The second dataset is a real dataset that is collected at intervals of 1 minute during the period from November 1 to November 30 in 2020 at an experimental jujube yard; we only selected 43154 data instances (shown in Figure 4) in the collected data. The sensor variables in the second dataset include temperature,

humidity, and atmospheric pressure, and the specific information of outliers marked by the experts is shown in Table 1.

5.2. Experimental Results. Firstly, we use the synthetic dataset to test the proposed ODNDC approach. The results in Figure 5 show that the point outlier, collective outlier, and jump outlier are correctly detected, where the source of point outlier is labeled as an error, and the source of jump outlier is labeled as an event. Because the dataset only has temperature variable, there are no relative variables that can be used to further distinguish the source of collective variables; thus, the source of collective outlier is labeled as unknown.

And then, we use the real dataset to test the ODNDC approach, and the experimental results are shown in Figure 6, where the specific detection accuracy of the ODNDC approach is listed in Table 2.

It can be seen from Figure 6 that through the use of our proposed ODNDC approach, there are 5 point outliers, 284 collective outliers, 7 jump outliers, and 27 contextual outliers that can be detected in the temperature variable of the sensor data; 15 point outliers, 240 collective outliers, 14 jump outliers, and 27 contextual outliers can be detected in the humidity variable of the sensor data; and 146 collective outliers, 2 jump outliers, and 27 contextual outliers can be detected in the atmospheric pressure variable of the sensor data. We can see from Table 2 that the detection accuracy of the proposed ODNDC approach is relatively high, and it also can accurately distinguish different types of outliers with the use of neighbor difference, w-k-means algorithm, and re-identification rules.

To further evaluate the proposed ODNDC approach, we use four outlier detection approaches, including LODA [35], ROCF [37], CORM [38], and Heuristic algorithm [44], to test the detection efficiency, where the real dataset collected at the experimental jujube yard is used in the experiment, and the experimental result is shown in Table 3.

It can be seen from Table 3 that the detection accuracy of the proposed ODNDC approach is the highest, while the detection accuracy of the CORM is the lowest. In these five compared approaches, the proposed ODNDC and compared Heuristic can identify the sources of outliers, and the identifying accuracy of the ODNDC is higher than that of the Heuristic. The reason for the high detection accuracy of the ODNDC approach is that it first uses neighbor difference to detect candidate outliers to reduce the influence of outliers on clustering results, and then uses spatial correlation and re-identification rules to detect different types of outliers. In contrast, the LODA, ROCF, and CORM approaches can only determine whether the data instances are outlier but cannot effectively identify the sources of outliers.

And then, the identification of the sources of outliers by the ODNDC approach on the temperature of the collected real dataset in two time windows (on November 23 and November 26) is conducted in the experiment, and the experimental result is shown in Figure 7, where the blue line represents the data collected by sensors.

Input: the streaming sensors data after outlier detection
Output: outliers and their sources

```

(01) if there is no outlier in current window then
(02)   exit
(03) else
(04)   if the data instance is detected as point outlier then
(05)     label its source as error
(06)   end if
(07)   if the data instance is detected as jump outlier then
(08)     label its source as event
(09)   end if
(10)   if the data instance is detected as collective outlier or contextual outlier then
(11)     calculate the correlation coefficient
(12)     if correlation coefficient >  $th_1$  then
(13)       if more than half of attributes are labeled as outliers then
(14)         label its source as event
(15)       else
(16)         label its source as error
(17)       end if
(18)     else if |correlation coefficient| >  $th_2$  then
(19)       read these correlative variables of data instances
(20)       predict the mean and variance of normal values with 10-folder cross validation
(21)       if the outlier is out of the predicted range then
(22)         label its source as error
(23)       else
(24)         label its source as suspected event
(25)       end if
(26)     end if
(27)   else
(28)     label the source of collective outliers as unknown
(29)     label the source of contextual outliers as normal
(30)   end if
(31) end if

```

ALGORITHM 3: Outlier sources identification based on correlation (OSIC).

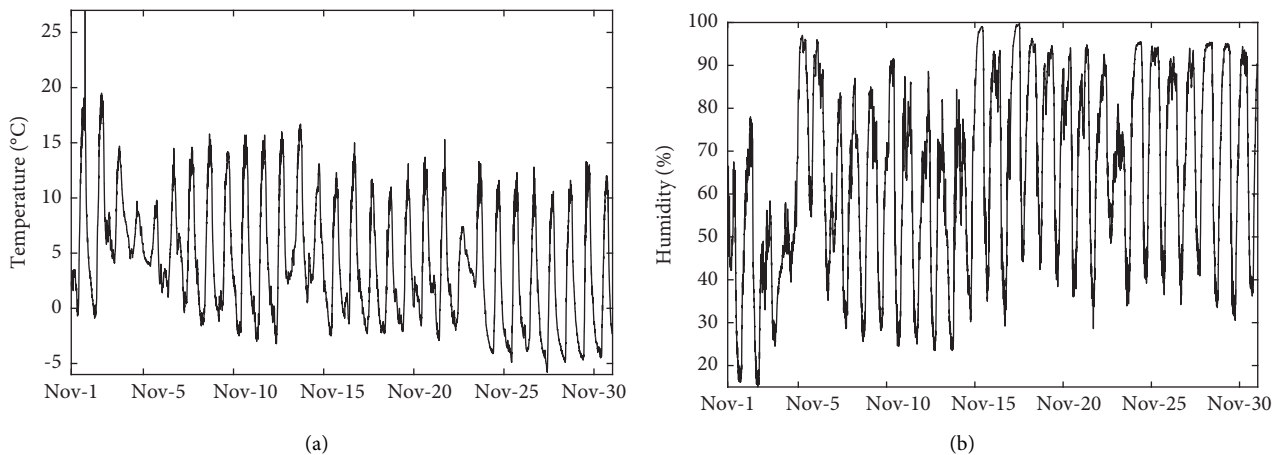


FIGURE 4: Continued.

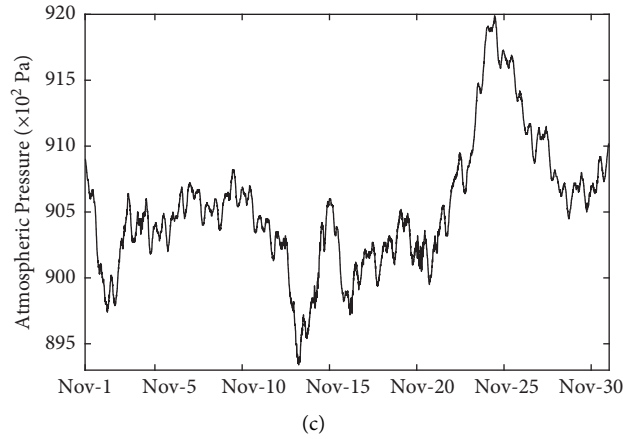


FIGURE 4: The data collected at intervals of 1 minute during November 2020 at an experimental jujube yard. (a) Temperature, (b) humidity, (c) atmospheric pressure.

TABLE 1: The specific information of contained outliers.

Number	Variables		
	Temperature	Humidity	Atmospheric pressure
Point outliers	6	16	0
Collective outliers	303	253	154
Jump outliers	7	15	2
Contextual outliers	28	29	29

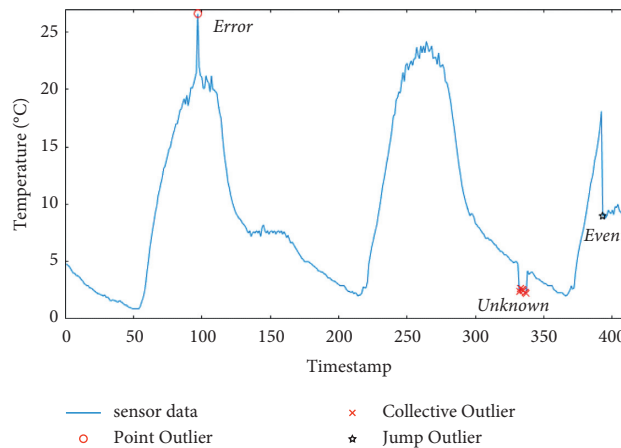


FIGURE 5: The result of outlier detection for the synthetic dataset.

It can be seen from Figure 7 that with the use of our proposed ODND C approach, the sources of outliers, including error and event, are effectively identified. In addition, the contexture outliers can be identified as “suspected event” by our proposed ODND C approach, which can prevent the identification error. The experimental result verifies that the use of correlation coefficient of sensor variables is beneficial for identifying the sources of outliers, and the designed identifying rule also improves the identifying efficiency.

Because the OSIC algorithm runs in the off-line mode, and the number of outliers is far less than that of normal instances, it has little effect on the time complexity of the ODND C approach. To evaluate this aspect, we test the running time of the

ODND C with the familiar w-k-means algorithm, the LODA, ROCF, CPRM, and Heuristic approaches on the real dataset, and the experimental results are shown in Figure 8.

It can be seen from Figure 8 that in the six approaches, the time cost of the LODA is the longest, and the time cost of the ROCF is the second longest, while the time cost of the w-k-means algorithm is the shortest. For the proposed ODND C approach, its time cost is slightly longer than that of the w-k-means algorithm. This is because the ODND C approach needs to calculate the neighbor difference before performing clustering operation to reduce the influence of outliers on clustering results. Compared with other four outlier detection approaches, the time cost of the ODND C is much shorter; it indicates that

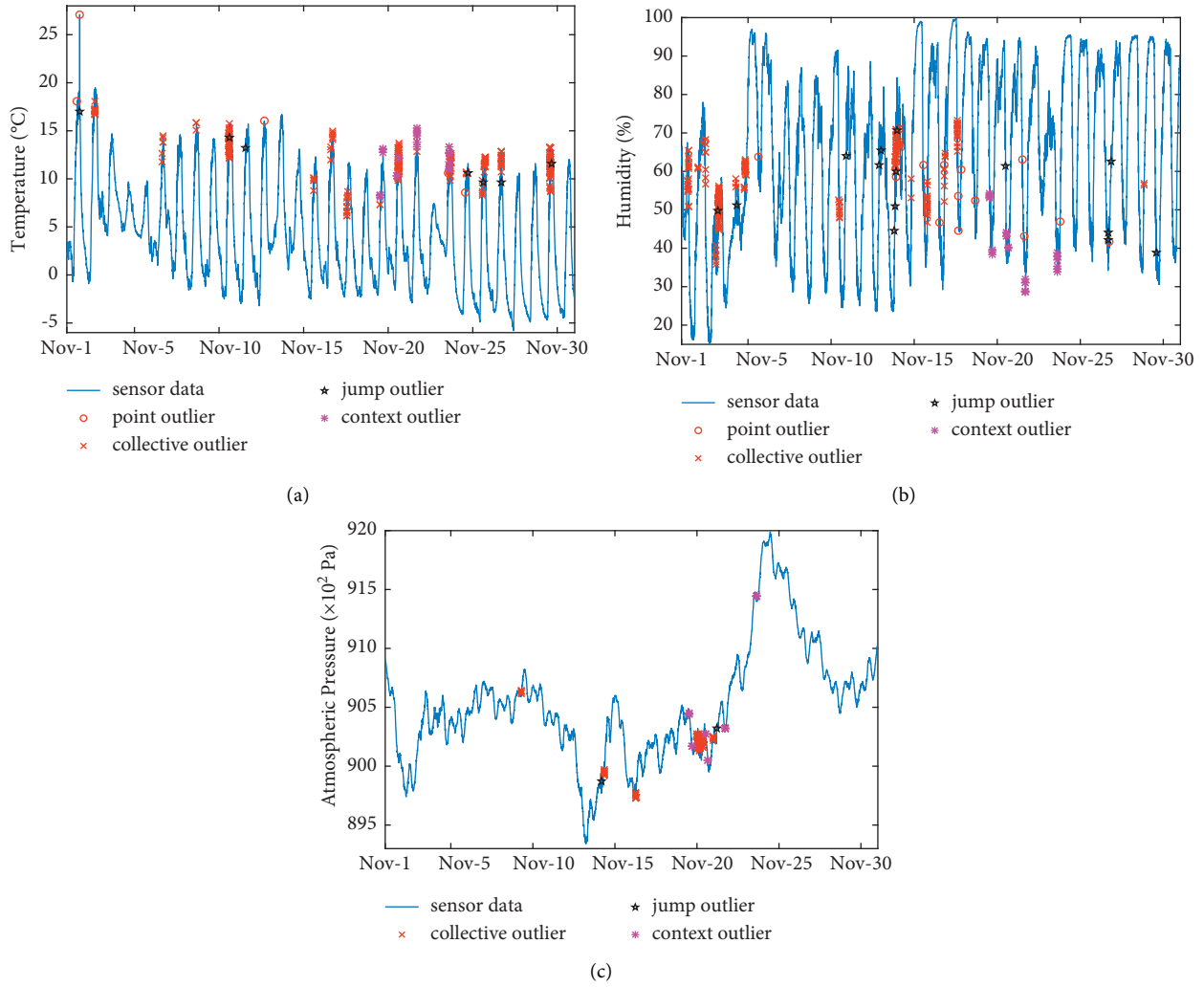


FIGURE 6: The results of outlier detection by ODNDC approach. (a) Temperature, (b) humidity, (c) atmospheric pressure.

TABLE 2: The detection accuracy of the ODNDC approach.

Types	Variables		
	Temperature (%)	Humidity (%)	Atmospheric pressure
Point outliers	83.3	93.8	—
Collective outliers	93.7	94.9	94.8%
Jump outliers	100	93.3	100%
Contextual outliers	96.4	93.1	93.1%

TABLE 3: The comparisons of detection accuracy

Types	ODNDC (%)	LODA	Variables		
			ROCF	CORM	Heuristic (%)
Point outliers	90.9	—	—	—	86.4
Collective outliers	94.4	—	—	—	92.5
Jump outliers	95.8	—	—	—	87.5
Contextual outliers	94.2	—	—	—	90.7
Overall	94.3	93.6%	92.8%	89.5%	92

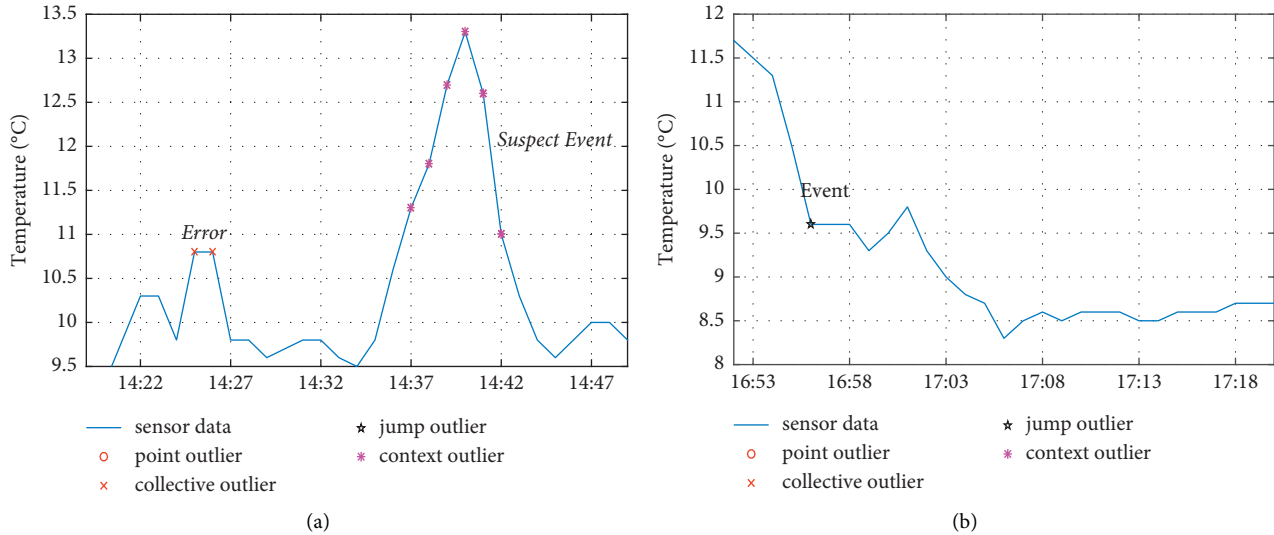


FIGURE 7: Two cases of outlier source identification in two time windows. (a) Nov-23, and (b) Nov-26.

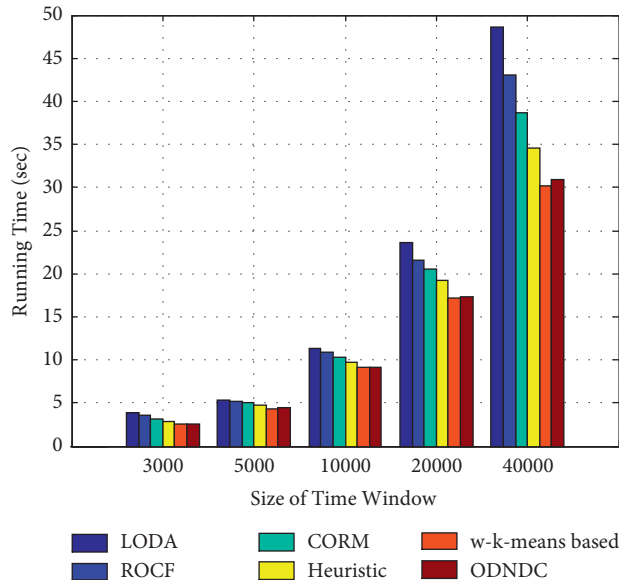


FIGURE 8: Runtime of the compared outlier detection algorithm and the ODND approach with different sizes of the time window.

the time efficiency of the ODND approach is very competitive. Compared with the w-k-means algorithm, although the time cost is slightly longer, the proposed ODND approach can accurately detect the outliers and distinguish the sources of outliers; thus, we can ignore the small amount of extra time consumption. The experimental result verifies whether the ODND is a time-efficient approach, which can apply to outlier detection for streaming sensor data.

6. Conclusions

In this paper, we propose a new approach of the ODND, which can detect the outliers in streaming sensor data and identify the sources of outliers based on the analysis of the data pattern. The ODND approach is composed of ODND, ODC,

and OSIC, where the outliers are detected and labeled by the ODND algorithm and ODC algorithm, and the sources of outliers are identified by the OSIC algorithm. In addition, the ODND algorithm also focuses on the temporal relationship of each streaming sensor data, and the ODC algorithm focuses on spatial relationship of all streaming sensor data, which solves the issue of the common clustering-based outlier detection algorithms neglecting temporal relationship of streaming sensor data. The experimental results show that the proposed ODND approach can accurately detect potential outliers from the collected data from WSNs, as well as with high time efficiency. Therefore, the proposed ODND approach can be effectively used in outlier detection in the environment of WSNs, thereby providing an accurate and reliable data for edge computing.

Although the use of the w-k-means algorithm can reduce the impact of irrelevant attributes on the clustering results, its clustering results depend on the initial cluster centers and the initial weights. In the future, we would like to use the coefficient of variation or entropy to select suitable initial weights to further improve the clustering efficiency; in addition, we also would like to study the patterns of abnormal sensors in specific domain and try to use the approach in early warning for sensors.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partly supported by National Natural Science Foundation of China (Grant numbers: U1836116 and 62172194), the China Postdoctoral Science Foundation (Grant number: 2021M691310), the Future Network Scientific Research Fund Project (Grant number: FNSRFP-2021-YB-50), the Postdoctoral Science Foundation of Jiangsu Province (Grant number: 2021K636C), the Natural Science Foundation of the Jiangsu Higher Education Institutions (Grant number: 21KJB520031), the Graduate Research Innovation Project of Jiangsu Province (Grant numbers: KYCX21_3375 and SJCX21_1692), and the College Student Innovation and Entrepreneurship Training Program (Grant number: 202110299178Y).

References

- [1] H. Liu and K. Y. Ki, "Application of wireless sensor network based improved immune gene algorithm in airport floating personnel positioning," *Computer Communications*, vol. 160, pp. 494–501, 2020.
- [2] A. Farhat, A. Jaber, R. Tawil, C. Guyeux, and A. Makhoul, "On the coverage effects in wireless sensor networks based prognostic and health management," *International Journal of Sensor Networks*, vol. 28, no. 2, pp. 125–138, 2018.
- [3] B. Cao, J. Zhao, Y. Gu, S. Fan, and P. Yang, "Security-aware industrial wireless sensor network deployment optimization," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5309–5316, 2020.
- [4] L.-L. Shi, L. Liu, Y. Wu et al., "Human-centric cyber social computing model for hot-event detection and propagation," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1042–1050, 2019.
- [5] L.-L. Shi, L. Liu, Y. Wu, L. Jiang, J. Panneerselvam, and R. Crole, "A social sensing model for event detection and user influence discovering in social media data streams," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 1, pp. 141–150, 2020.
- [6] L.-L. Shi, L. Liu, Y. Wu, L. Jiang, and A. Ayorinde, "Event detection and multi-source propagation for online social network management," *Journal of Network and Systems Management*, vol. 28, no. 1, pp. 1–20, 2020.
- [7] X. Li, T. Chen, Q. Cheng, S. Ma, and J. Ma, "Smart applications in edge computing: overview on authentication and data security," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4063–4080, 2021.
- [8] L. Ullah, M. S. Khan, M. St-Hilaire, and M. Faisal, "Task priority-based cached-data prefetching and eviction mechanisms for performance optimization of edge computing clusters," *Security and Communication Networks*, vol. 2021, Article ID 5541974, 10 pages, 2021.
- [9] S. Cai, R. Huang, J. Chen et al., "An efficient outlier detection method for data streams based on closed frequent patterns by considering anti-monotonic constraints," *Information Sciences*, vol. 555, pp. 125–146, 2021.
- [10] M. Radovanovic, A. Nanopoulos, and M. Ivanovic, "Reverse nearest neighbors in unsupervised distance-based outlier detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1369–1382, 2015.
- [11] S. Cai, J. Chen, H. Chen et al., "An efficient anomaly detection method for uncertain data based on minimal rare patterns with the consideration of anti-monotonic constraints," *Information Sciences*, vol. 580, pp. 620–642, 2021.
- [12] J. Fan, Q. Zhang, J. Zhu, M. Zhang, Z. Yang, and H. Cao, "Robust deep auto-encoding Gaussian process regression for unsupervised anomaly detection," *Neurocomputing*, vol. 376, pp. 180–190, 2020.
- [13] E. K. Boahen, B. E. Bouya-Moko, and C. Wang, "Network anomaly detection in a controlled environment based on an enhanced PSO/SARFC," *Computers & Security*, vol. 104, p. 102225, 2021.
- [14] C. Titouna, F. Nait-Abdesselam, and A. Khokhar, "DODS: a distributed outlier detection scheme for wireless sensor networks," *Computer Networks*, vol. 161, pp. 93–101, 2019.
- [15] Ł. Saganowski, T. Andrysiak, R. Kozik, and M. Choraś, "DWT-based anomaly detection method for cyber security of wireless sensor networks," *Security and Communication Networks*, vol. 9, no. 15, pp. 2911–2922, 2016.
- [16] N. Nesa, T. Ghosh, and I. Banerjee, "Non-parametric sequence-based learning approach for outlier detection in IoT," *Future Generation Computer Systems*, vol. 82, pp. 412–421, 2018.
- [17] O. Iraqi and H. El Bakkali, "Application-level unsupervised outlier-based intrusion detection and prevention," *Security and Communication Networks*, vol. 2019, pp. 1–13, 2019.
- [18] E. X. Min, J. Long, Q. Liu, J. J. Cui, and W. Chen, "Tr-I. D. S.: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Security and Communication Networks*, vol. 2018, Article ID 4943509, 9 pages, 2018.
- [19] B. Saneja and R. Rani, "An efficient approach for outlier detection in big sensor data of health care," *International Journal of Communication Systems*, vol. 30, no. 17, pp. 1–10, 2017.
- [20] K. Edward, C. D. Wang, and B. Bouya-Moko, "Detection of compromised online social network account with an enhanced knn," *Applied Artificial Intelligence*, vol. 34, no. 11, pp. 777–791, 2020.
- [21] X. Sun, "Similarity detection method of abnormal data in network based on data mining," *Journal of Intelligent and Fuzzy Systems*, vol. 38, no. 1, pp. 155–162, 2020.
- [22] K. Thangaramya, K. Kulothungan, S. Indira Gandhi, M. Selvi, S. V. N. Santhosh Kumar, and K. Arputharaj, "Intelligent fuzzy rule-based approach with outlier detection for secured routing in WSN," *Soft Computing*, vol. 24, no. 21, pp. 16483–16497, 2020.

- [23] F. Angiulli, S. Basta, S. Lodi, and C. Sartori, "GPU strategies for distance-based outlier detection," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 11, pp. 3256–3268, 2016.
- [24] B. Tang and H. He, "A local density-based approach for outlier detection," *Neurocomputing*, vol. 241, pp. 171–180, 2017.
- [25] E. Bigdeli, M. Mohammadi, B. Raahemi, and S. Matwin, "Incremental anomaly detection using two-layer cluster-based structure," *Information Sciences*, vol. 429, pp. 315–331, 2018.
- [26] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Hyperspherical cluster based distributed anomaly detection in wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, pp. 1833–1847, 2014.
- [27] O. Ghorbel, M. W. Jmal, W. Ayedi, H. Snoussi, and M. Abid, "An overview of outlier detection technique developed for wireless sensor networks," in *Proceedings of the International 10th Multi-Conference on Systems, Signals & Devices (SSD)*, pp. 1–6, IEEE, Hammamet, Tunisia, March 2013.
- [28] A. Alkhatib and Q. Abed-Al, "Multivariate outlier detection for forest fire data aggregation accuracy," *Intelligent Automation & Soft Computing*, vol. 31, no. 2, pp. 1071–1087, 2022.
- [29] S. A. N. Nozad, M. A. Haeri, and G. Folino, "SDCOR: scalable density-based clustering for local outlier detection in massive-scale datasets," *Knowledge-Based Systems*, vol. 228, p. 2021, 107256.
- [30] D. Krleza, B. Vrdoljak, and M. Brcic, "Statistical hierarchical clustering algorithm for outlier detection in evolving data streams," *Machine Learning*, vol. 110, no. 1, pp. 139–184, 2021.
- [31] Y. Zhang, N. A. S. Hamm, N. Meratnia, A. Stein, M. van de Voort, and P. J. M. Havinga, "Statistics-based outlier detection for wireless sensor networks," *International Journal of Geographical Information Science*, vol. 26, no. 8, pp. 1373–1392, 2012.
- [32] S. Bharti, K. K. Pattanaik, and A. Pandey, "Contextual outlier detection for wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 4, pp. 1511–1530, 2020.
- [33] I. G. A. Poornima and B. Paramasivan, "Anomaly detection in wireless sensor network using machine learning algorithm," *Computer Communications*, vol. 151, pp. 331–337, 2020.
- [34] A. De Paola, S. Gaglio, G. Lo Re, F. Milazzo, and M. Ortolani, "Adaptive distributed outlier detection for WSNs," *IEEE Transactions on Cybernetics*, vol. 45, no. 5, pp. 888–899, 2015.
- [35] M. Safaei, A. S. Ismail, H. Chizari et al., "Standalone noise and anomaly detection in wireless sensor networks: a novel time-series and adaptive Bayesian-network-based approach," *Software: Practice and Experience*, vol. 50, no. 4, pp. 428–446, 2020.
- [36] J. A. Lara, D. Lizcano, V. Ramperez, and J. Soriano, "A method for outlier detection based on cluster analysis and visual expert criteria," *Expert Systems*, e12473, vol. 37, no. 5, , 2019.
- [37] J. Huang, Q. Zhu, L. Yang, D. Cheng, and Q. Wu, "A novel outlier cluster detection algorithm without top-n parameter," *Knowledge-Based Systems*, vol. 121, pp. 32–40, 2017.
- [38] M. Elahi, K. Li, W. Nisar, X. Lv, and H. Wang, "Efficient clustering-based outlier detection algorithm for dynamic data stream," in *Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 298–304, Spring, Xi'an, China, 19 december 2008.
- [39] N. Randive, "Sneha. Hybrid approach for outlier detection in high dimensional dataset," *International Journal of Science and Research*, vol. 3, no. 7, pp. 1743–1746, 2014.
- [40] J. Z. Huang, M. K. Ng, H. Hongqiang Rong, and Z. Zichen Li, "Automated variable weighting in k-means type clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 657–668, 2005.
- [41] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.
- [42] C. C. Aggarwal, P. S. Yu, J. Han, J. Wang, and R. Ctr, "A framework for clustering evolving data streams," in *Proceedings of the 2003 VLDB Conference. In: Proceedings of VLDB Conference*, pp. 81–92, Elsevier, 18 September 2003.
- [43] E. W. Grafarend, *Linear and Nonlinear Models: Fixed Effects, Random Effects, and Mixed Models*, Walter de Gruyter, Berlin, New York, 2006.
- [44] A. F. Hassan, H. M. O. Mokhtar, O. Hegazy, A. F. Hassan, and O. Hegazy, "A heuristic approach for sensor network outlier detection," *International Journal of Research and Reviews in Wireless Sensor Networks*, vol. 1, no. 4, pp. 66–72, 2011.

Research Article

Blockchain-Empowered Secure and Privacy-Preserving Health Data Sharing in Edge-Based IoMT

Xueli Nie ¹, Aiqing Zhang ^{1,2}, Jindou Chen ^{1,2}, Youyang Qu ³ and Shui Yu ⁴

¹School of Physics and Electronic Information, Anhui Normal University, Wuhu 241002, China

²Anhui Provincial Engineering Laboratory on Information Fusion, Wuhu 241002, China

³Deakin Blockchain Innovation Lab, School of Information Technology, Deakin University, Geelong, Australia

⁴School of Computer Science, University of Technology Sydney, Ultimo, Australia

Correspondence should be addressed to Aiqing Zhang; aqzhang2006@163.com

Received 12 September 2021; Revised 8 December 2021; Accepted 13 January 2022; Published 21 February 2022

Academic Editor: Ding Wang

Copyright © 2022 Xueli Nie et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Health data sharing, as a booming demand, enables the patients with similar symptoms to connect with each other and doctors to obtain the medical history of patients. Health data are usually collected from edge-based Internet of medical things (IoMT) with devices such as smart wearable devices, smart watches, and smartphones. Since health data are highly private and have great financial value, adversaries ceaselessly launch diverse attacks to obtain private information. All these issues pose great challenges to health data sharing in edge-based IoMT scenarios. Existing research either lacks comprehensive consideration of privacy and security protection or fails to provide a proper incentive mechanism, which expels users from sharing data. In this study, we propose a novel blockchain-assisted data sharing scheme, which allows secure and privacy-preserving profile matching. A bloom filter with hash functions is designed to verify the authenticity of keyword ciphertext. Key-policy attribute-based encryption (KP-ABE) algorithm and smart contracts are employed to achieve secure profile matching. To incentivize users actively participating in profile matching, we devise an incentive mechanism and construct a two-phase Stackelberg game to address pricing problems for data owners and accessing problems of data requesters. The optimal pricing mechanism is specially designed for encouraging more users to participate in health data sharing and maximizing users' profit. Moreover, security analysis illustrates that the proposed protocol is capable of satisfying various security goals, while performance evaluation shows high scalability and feasibility of the proposed scheme in edge-based IoMT scenarios.

1. Introduction

Internet of medical things (IoMT) can improve healthcare service quality by sharing health data among users and realize remote health care. Generally, profile matching is the prerequisite of health data sharing. For example, patients with similar symptoms may want to connect with others, exchange their experiences, and broaden the understanding of the symptoms helping them to get early diagnosis or better treatments in time [1, 2].

Since IoMT devices are usually resource constraint, edge computing is introduced into the system to improve the efficiency [3, 4]. Due to privacy-sensitive nature of health data, data security and privacy preservation are crucial in

edge-based IoMT system. Although the existing schemes use cryptographic algorithms to achieve security, there are still some threats because the edge is a semi-trusted center and also faces a single point of failure. In addition, these works failed to take incentive problem into consideration for users. Even though incentive mechanism is considered in [5, 6], the security issues in profile matching have hardly been solved.

Fortunately, emerging blockchain technology is tamper-proof, decentralized, transparent, anonymous, and autonomous, which is a promising solution for the aforementioned problems [7]. Although blockchain is conducive to profile matching and health data sharing, there are still three following challenges: (1) how to realize privacy preservation through profile matching in blockchain environment? (2)

How to ensure that the authenticity of the keyword is selected from a valid keyword set while not revealing users' private information? (3) How to design an incentive mechanism for motivating users to take an active part in the system?

To solve the above challenges, we put forward a secure and privacy-preserving profile matching scheme by employing key-policy attribute-based (KP-ABE) keyword search and bloom filter in edge-based IoMT. Blockchain is adopted to store the keyword ciphertext. It ensures that requesters access the desired data and protect data privacy. Moreover, the bloom filter verifies the authenticity of keyword ciphertext using various hash functions without disclosing any sensitive information. We design an optimal pricing mechanism to incentivize data owners and data requesters to participate in this system.

The major contributions of the proposed scheme are summarized as follows.

We construct a new blockchain-based profile matching framework in edge-based IoMT with security and privacy preservation. We design a bloom filter based on many hash functions to verify whether the keyword is selected from a valid keyword set before storing the keyword ciphertext in blockchain. In this way, the computational cost is significantly reduced.

We present a key-policy attribute-based keyword search and bloom filter profile matching protocol in a blockchain-enabled edge-based IoMT. Only when data users' attribute set satisfies the access structure set, users are able to access the desired data. Data owners' original data are stored in local server, while the corresponding keyword ciphertext is kept in blockchain verified by a bloom filter. It greatly improves search efficiency.

We devise an optimal pricing mechanism to motivate users to actively participate in the system. The data owners take part in pricing and setting payment for matching data according to pricing list based on an optimal pricing mechanism. In particular, it provides an economic approach to analyze the incentive mechanism. By utilizing game-based optimal pricing mechanism, data owners and data requesters can obtain their maximum benefits. Furthermore, we implement smart contracts on the Ethereum platform and conduct extensive evaluations to demonstrate the superiority of the proposed scheme.

The structure of the study is organized as follows. Related works about health data sharing with edge computing blockchain-based profile matching are elaborated in Section 2. Section 3 introduces preliminaries of the work. Section 4 presents system architecture, workflow, security threats, and design goals. Section 5 designs the detailed profile matching protocol for the proposed blockchain, proposes an optimal pricing mechanism to incentive data owner and data requester actively participating in data sharing, and presents the smart contracts. Then, Section 6 analyzes how our scheme achieves the security goals. Section 7 carries out the

proposed scheme on Ethereum platform to estimate the performance of protocol and smart contracts. Finally, Section 8 concludes this work.

2. Related Work

In this section, we introduce existing relevant researches on health data sharing with edge computing, profile matching, and incentive mechanisms.

2.1. Health Data Sharing with Edge Computing. Privacy issues emerge with the fast development of edge computing and IoMT [8, 9]. The edge node was the closest to the restricted resource devices and had strong computing power. Therefore, some edge-based healthcare systems were proposed [10]. Pustokhina et al. [11] presented an effective scheme in edge computing-enabled IoMT system. The IoMT devices sensed patient's data and transferred the captured data to edge computing. Aiming at guaranteeing data security and flexible right control simultaneously, an efficient attribute-based encryption (ABE) scheme was proposed to outsource part of encryption and decryption to edge nodes [12]. It supported attribute updates and reduced the computing cost of resource-constrained devices for edge-enabled smart health care. To rely on a trusted center, Akkaoui et al. [13] proposed a blockchain-based secure health data sharing framework in edge computing, named "EdgeMediChain." The work promoted the necessary requirements for scalability, security, and privacy of medical ecosystem. Similarly, Nguyen et al. [14] proposed a new decentralized health architecture that integrates mobile edge computing and blockchain for data offloading and data sharing in distributed hospital networks. They utilized a decentralized authentication mechanism associated with a distributed IPFS storage to improve service quality. The ongoing study [15] proposed a ubiquitous healthcare framework that leveraged edge computing, big data, deep learning, high-performance computing, and the IoMT to solve the above challenges. The framework made use of four layers. Three main components improved network service quality. However, these works did not provide profile matching and incentive mechanisms for these health data.

2.2. Profile Matching. Some cloud-assisted solutions have been proposed to address the problems of data security and privacy preservation for profile matching [16–18]. Li et al. [17] put forward a privacy-preserving and scalable matching protocol without disclosing the users' personal data to the cloud. To achieve secure communication between users, He et al. [18] designed an efficient Cross-Domain HandShake (CDHS) scheme with symptom matching in hierarchical identity-based cryptosystem. Compared with the existing profile matching schemes, their schemes used cloud computing to conduct most of the computation, which effectively reduced user's computation complexity. However, the cloud is a semi-trusted center, which may collude with other malicious users to obtain the sensitive data.

Blockchain has drawn considerable interests in research and industrial communities [19–22] due to its advantages. Yang et al. [23] designed a distributed matching mechanism based on blockchain. Furthermore, their scheme transformed the social welfare maximization problem into a matching game of bilateral matching and unilateral preference. Meng et al. [24] proposed a blockchain-enabled signature matching while achieving data security and privacy preservation. In [25], a matching scheme based on hierarchical blockchain was proposed to protect users' privacy. The scheme combined ciphertext-policy attribute-based (CP-ABE) encryption and bloom filter to meet users' requirements to search friends.

The aforementioned works proposed heuristic blockchain-based profile matching schemes with data security and privacy preservation. In fact, some researches presented a structure or concept for profile matching based on blockchain without technically proposing a solution in detail for a specific application scenario. In this work, we present a novel blockchain-based profile matching framework by employing (key-policy attribute-based encryption) KP-ABE and bloom filter to achieve data security and privacy preservation. In addition, we design a detailed profile matching protocol and implement the devised smart contracts on Ethereum test platform.

2.3. Incentive Mechanisms. To encourage more users to join the system, some incentive mechanisms have been designed. Jiao et al. [26] proposed a profit maximization mechanism to maximize the profit under different users' valuation distributions. The pricing mechanism effectively solved the profit maximization problem and provided useful strategies for users. Game-theoretic methods had been extensively used in pricing mechanism [27, 28]. The uncertainty of future prices with a single-leader, multiple-follower Stackelberg game was proposed to maximize profits by setting optimal pricing strategy [27]. It provided a simple distributed algorithm for finding the unique Stackelberg equilibrium point. Chen et al. [28] proposed a pricing approach for incentive mechanisms and considered a two-stage game in a three-layer architecture. They formulated a Markov decision process (MDP)-based social welfare maximization model and studied a convex optimization pricing problem.

Some works adopted blockchain technology to design incentive mechanisms. Liu et al. [29] designed an optimal pricing mechanism and adopted a two-phase Stackelberg game to address pricing and buying problem of the users. Furthermore, they used backward induction to investigate the quantity of purchased data and pricing strategies. Li et al. [30] put forward an incentive mechanism to encourage users to participate in sharing data through price-aware top-k matching queries. Xiong et al. [31] proposed multi-leader multi-follower game-based alternating direction method of multipliers algorithm for pricing to accomplish optimum results in a distributed manner. Zhang et al. [32] designed a dynamic, hierarchical pricing mechanism based on economic modeling methods and heterogeneous agent theory. Nie et al. [33] proposed an optimal incentive mechanism of

users using backward induction and applied a Stackelberg game to analyze users' participation level.

The existing work failed to apply the incentive mechanism to encourage data owners and data requesters to share health data. In this scheme, we combine blockchain technology and smart contract to design an optimal pricing mechanism with the guarantee of data security and privacy preservation for motivating users to actively join the system and maximizing their profits.

3. Preliminaries

In this section, we provide preliminaries required in this work.

3.1. Cryptographic Assumptions

Definition 1. Decisional Bilinear Diffie–Hellman Problem (DBDH). We denote an elliptic curve as E and consider a cycle group G of prime order q . Let P be a generator in G_1 and $a, b, c, z \in Z_q^*$. The DBDH problem is defined as follows: given an input tuple $(P \in G_1, aP \in G_2, bP \in G_2, cP \in G_2)$, $e(P, P)^{abc}$ is computed. Assume that an attacker \mathcal{A} can successfully distinguish between $e(P, P)^{abc}$ and $e(P, P)^z$ with the advantage $\text{Adv}_{\mathcal{A}}^{\text{DBDH}}(\lambda)$. If the DBDH assumption holds, the advantage $\text{Adv}_{\mathcal{A}}^{\text{DBDH}}(\lambda) \leq \epsilon$ must be ignored.

3.2. Linear Secret-Sharing Schemes (LSSS).

$$\text{Adv}_{\mathcal{A}}^{\text{DBDH}}(\lambda) = \left| \frac{\Pr[\mathcal{A}(e, P, P, P, P, e(P, P)^{abc}) = 1]}{-\Pr[\mathcal{A}(e, P, P, P, P, e(P, P)^z) = 1]} \right| \leq \epsilon. \quad (1)$$

An LSSS access structure is denoted as (N, ρ) , where N is an $l \times n$ matrix and ρ is an injective function from $\{1, 2, \dots, l\}$ to attributes. Let $S \in N$ be an authorized set, defined as $I = \{i \mid \rho(i) \in S\}$. Let T_r be the set of distinct attributes in N . Here, $T_r = \{b \mid \exists i \in [1, l], \rho(i) = b\}$. There exist constants $\{\pi_i \in Z_p, i \in I\}$ such that $\sum_{i \in I} \pi_i N_i = (1, 0, \dots, 0)$, where N_i is i th row of N . It randomly generates a vector $(\vec{v}) = \{\mu_1, r_2, \dots, r_n\}$, where $\mu_1 \in Z_q^*$ is the secret to be shared and $r_2, \dots, r_n \in Z_q^*$ is randomly chosen. For $1 \leq i \leq l$, it will compute $\lambda_i = (\vec{v})N_i$, where N_i is i th row of N . Given an attribute set S and N_i , $\{\pi_i \in Z_q, i \in I\}$ is found and $\sum_{i \in I} \pi_i N_i = (1, 0, \dots, 0)$ is formulated.

3.3. Bloom Filter. A bloom filter [34] is a data structure for validating whether an element comes from a set. An m -bit bloom filter can be represented by a set of hash functions $H = \{H'_1, H'_2, \dots, H'_k\}, i \in [1, k]$. All the positions in the array are all initially 0. To query an input $w \in W = \{w_1, \dots, w_n\}$ into the bloom filter, its corresponding position is set as 1; i.e., $BF[H'_i(w)] = 1, i \in [1, k]$ is set. The bloom filter checks whether an element w is selected from a set W . w is hashed by the k hash functions $H = \{H'_1, H'_2, \dots, H'_k\}$ to obtain k hashed array positions. If any of the bits in these positions

are set to 0, it represents that the element w does not belong to the set. Otherwise, all the bits in the positions are set to 1, which means the element is in the set. There exists a possibility that $w \notin W$, and all bits in the corresponding positions are all 1, termed as the false-positive error. Its probability is $(1 - (1 - 1/m)^{kn})^k$. Here, $k = m/n \ln 2$. Hash functions are able to minimize false-positive probability $(0.6185)^{m/n}$. There are two algorithms, shown as follows:

$BF \leftarrow BF\text{Build}(\{H'_1, H'_2, \dots, H'_k\}, \{w_1, w_2, \dots, w_n\})$. $BF\text{Build}$, and algorithm puts all w into BF .

$c_w \leftarrow \text{Verify}BF(\{H'_1, H'_2, \dots, H'_k\}, BF, w)$. If $w \notin W$, it returns 0. Otherwise, it returns 1.

3.4. Blockchain and Smart Contract. Blockchain is a distributed ledger, which records numerous transactions [35]. All nodes in the network have the same copy of the ledger. Blockchain has the features of decentralization, privacy preservation, immutability, fault tolerance, and the ability to implement smart contracts. In blockchain, consensus algorithm is used to address the consistency issue of untrusted systems [36]. Current proposed consensus mechanisms include proof of work (PoW), proof of stake (PoS), and delegated proof of stake (DPoS) [37–39]. In our system, we design a proof of existence consensus mechanism for the proposed system.

Smart contract is a computer program that can be self-executed, self-verified, and tamper-resistant without trusted parties. The credibility of smart contract drives from its unforgeability. It cannot be modified or altered once it is deployed on blockchain. Ethereum is a decentralized application platform of smart contract, which supports application of Turing complete. It allows users to create, deploy, and run smart contract on blockchain. In our scheme, we design and deploy the smart contracts on Ethereum to test its availability and evaluate its performance.

4. System Model

In this section, we first propose the system architecture built upon edge-based IoMT. Then, we analyze the security threats and set the security goals.

4.1. System Architecture. The proposed model is made up of different entities: attribute authority (AA), IoMT devices, data owners (DOs), data requesters (DRs), and edge nodes, as shown in Figure 1. The major symbols used in the system architecture are listed in Table 1.

4.1.1. Attribute Authority. AA takes charge of system initialization and registration for DO and DR. It is a trustworthy center in this system. Furthermore, AA generates attribute keys for authorized entities.

4.1.2. IoMT Devices. IoMT devices contain smart devices, such as smartphone, smart watch, and other wearable

devices. They are used to collect the data, especially health data in this context. Furthermore, the collected data are uploaded to DOs who can share their data with others within the edge-based IoMT infrastructure.

4.1.3. Data Owners. Data owners refer to patients who construct a community to share their medical experiences and symptoms with others for broadening healthcare information. These DOs can form a social Internet. Additionally, they can earn fees by sharing their symptoms and experiences. They must register at the AA for obtaining attribute keys and joining the system. Moreover, DOs encrypt the keywords of their symptoms using attribute keys. They participate in pricing smart contract to derive an optimal price according to their own pricing strategies.

4.1.4. Data Requesters. Data requesters refer to patients who want to seek similar patients. Similar patients can share medical experiences, so that data requesters can better understand their health states. The DR should firstly register at the AA for attribute keys to take part in the system. The DR can generate search tokens using attribute keys and search relevant symptoms on blockchain. Once DRs receive the matching result from pricing and payment smart contract, they will select the intending data according to the matching results and pay fees to the corresponding DO by pricing and payment smart contract. Furthermore, they earn the fee by providing feedback to feedback smart contract.

4.1.5. Edge Nodes. Edge nodes are used to maintain the blockchain. Edge nodes are also responsible for packaging transactions into blocks in the blockchain.

There are three smart contracts deployed in our proposed blockchain: verification smart contract (VSC), pricing and payment smart contract (PPC), and feedback smart contract (FSC). Firstly, verification smart contract is used to verify whether the profile keywords are valid. Secondly, pricing and payment smart contract is used to set price based on the optimal pricing mechanism and transfer fee to corresponding user's account. Finally, feedback smart contract is used to reward data requester.

4.2. Workflow. Firstly, data requesters and data owners register at the AA for gaining attribute keys to join the system. Patients (data owners) construct a community to share their medical experiences and symptoms with others. They encrypt and upload their profile keywords. KP-ABE and verification smart contract verify whether the keywords belong to a keywords set. If the keywords are valid, they will be uploaded to blockchain for users to search. A data requester, who wants to seek similar patients without real identities, uses the master public key, keywords, and his/her private key to generate a token for searching. After that, if there are matching keywords, they will be sent to pricing and payment smart contract. The results of pricing will be sent to the DR. Furthermore, the DR is able to access DOs' data by paying fees to them. DRs send feedback information to

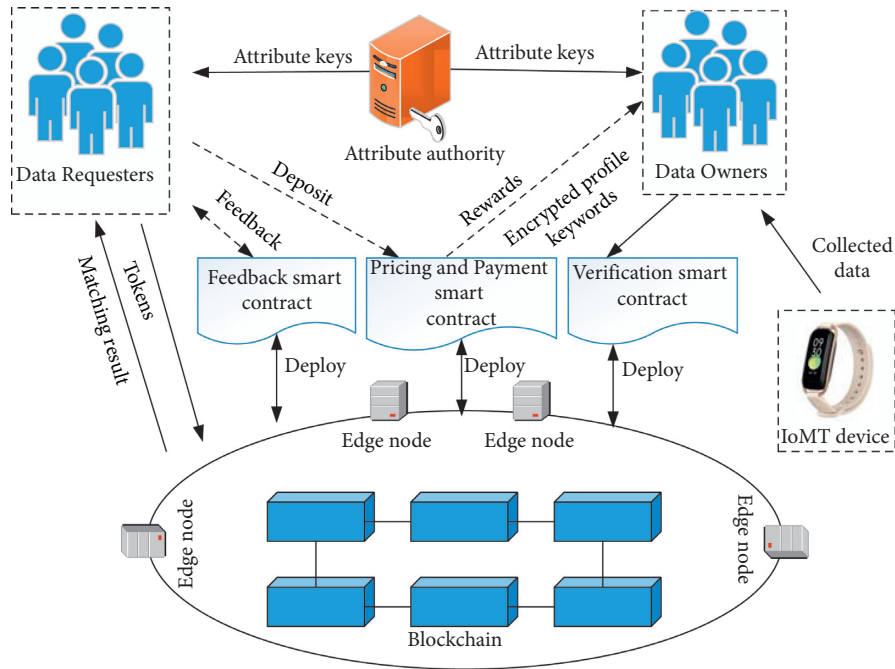


FIGURE 1: System architecture.

TABLE 1: Notation table of the architecture.

Notation	Description
KP-ABE	Key-policy attribute-based encryption
AA	Attribute authority
DOs	Data owners
DRs	Data requesters
VSC	Verification smart contract
PPC	Pricing and payment smart contract
FSC	Feedback smart contract

feedback smart contract. Moreover, a small fee is rewarded to the DR by feedback smart contract.

4.3. Security Threats and Design Goals. In our scheme, the AA is completely trusted for performing registration and generating attribute keys, but unauthorized entities may access the encrypted profile keywords and tokens to gain DOs’ private information such as identity and address during its transmission in the system. Furthermore, there may be some dishonest requesters who access DOs’ data without paying fees. Some DOs may provide false/fake information to requesters. Thus, we aim to achieve the following security goals.

4.3.1. Privacy Preservation. DOs’ identities contain some significant privacy information, which cannot be leaked or learned by unauthorized opponents. Besides, feedback information from requesters cannot reveal the DOs’ identity information.

4.3.2. Secure Match. In our scheme, the eavesdroppers attempt to guess attribute keys or search tokens for accessing

data. Therefore, it is of great significance to protect keys and tokens from revealing during matching process.

4.3.3. Fairness and Incentive. DRs must pay a reasonable fee to a DO for accessing the DOs’ data. On the other hand, DOs should provide true information for DRs. Smart contract could realize payment with fairness and verify the authenticity of the matched profile keywords. To incentivize more DRs and DOs to participate in sharing health data, we design an optimal pricing mechanism to maximize their profits. When DRs and DOs accomplish data sharing, they will obtain corresponding rewards.

4.3.4. Access Control. DOs have the ability to control data access and establish access policy. Access control ensures that requesters satisfying access policy can access the DOs’ information. Malicious attackers cannot eavesdrop or modify patients’ profile keywords, which are transmitted in the public channel of edge-based IoMT. The attribute-based encryption algorithm is adopted to achieve data confidentiality and integrity in this system.

5. The Proposed Protocol

In this section, we firstly describe the proposed protocol based on blockchain. Then, we demonstrate the optimal pricing mechanism.

5.1. Protocol Description. The proposed protocol contains three phases: system setup, index generation, and profile matching. The process of the proposed protocol is shown in Figure 2.

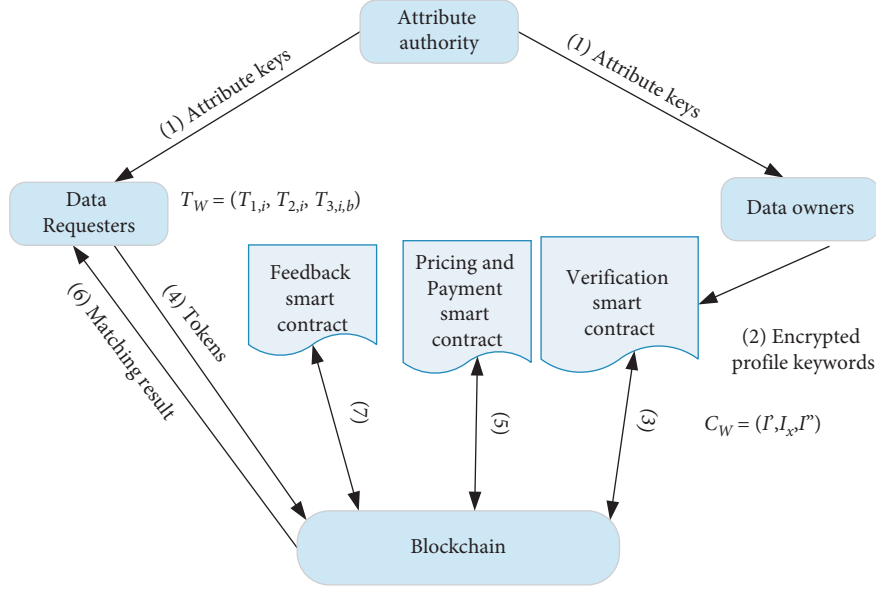


FIGURE 2: Proposed protocol.

5.1.1. Phase1: System Setup. Given a security parameter λ , the AA selects a bilinear map [40] $e: G_1 \times G_2 \rightarrow G_T$, where G_1 and G_2 are two additive cyclic groups of the same prime order q . G_T is a multiplicative cyclic group of prime order q . P_1 is the generator of G_1 , and P_2 is the generator of G_2 . AA chooses a hash function H_1 , where $H_1: \{0, 1\} \rightarrow Z_q^*$. In addition, the AA randomly chooses μ_1, μ_2 , and $\theta_i \in Z_q^*$, $g = \hat{e}(P_1, P_2)^{\mu_2}$. In the system, it defines the attribute space as U . Let m be the number of bits in a bloom filter (BF) and k be the amount of hash functions in a BF. The AA selects $U = |\hat{U}|$ random group elements $h_1, h_2, \dots, h_U \in G_1$. It generates k hash functions H'_1, H'_2, \dots, H'_k , which are used to add an element into corresponding positions in a BF. The master public key $MPK = (H_1, \hat{e}, g, P_1, P_2, \mu_1 P_1, \mu_2 P_2, \theta_i P_1, \theta_i P_2, h_1, h_2, \dots, h_U, H'_1, H'_2, \dots, H'_k)$, and the master secret key $MSK = (\mu_1, \mu_2, \theta_i)$.

The AA generates a secret key when a requester registers at the AA. There is an LSSS access structure (N, ρ) for the requester, where N is an $l \times n$ matrix and T_r is a set of diverse attributes in N . It means $T_r = \{b: \exists i \in [1, l], \rho(i) = b\}$, and $\forall b \in T_r, \rho(i)$. The function ρ makes a row of matrix N map to attributes. It chooses a random vector $(\vec{v}) \in Z_q^*$. This vector's values will be used to share μ_1 . For $1 \leq i \leq l$, it computes $\lambda_i = (\vec{v})N_i$, where N_i is i th row of N . Furthermore, it chooses $\sigma_1, \dots, \sigma_l \in Z_q^*$ and computes $A_i = \lambda P_2 + \theta_{\rho(i)} \sigma_i P_2$, $B_i = \sigma_i P_2$, $\forall b \in T_r, \rho(i)$, and $E_{i,b} = \theta_b \sigma_i P_2$. Let the private key be $ak = (A_i, B_i, E_{i,b})$.

5.1.2. Phase 2: Index Generation. Firstly, consensus users from edge nodes participating in profile matching are selected to participate in the network consensus, shown in Figure 3. Assume that the number of all consensus users is N_c in the blockchain network. The system generates a random number as the index of the consensus users to be the selected as the miner.

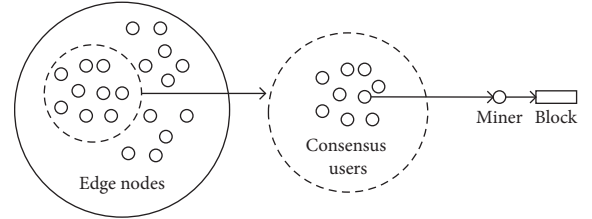


FIGURE 3: Consensus process.

Secondly, an edge node selects a random value $\chi \in Z_q^*$. He/she computes the keyword ciphertext $c_w = (I_1, I_z, I_2)$, where $I_1 = \chi P_1$, $I_z = \theta_z \chi P_1$, $z \in Att$, and $I_2 = g^{H_1(w)\chi}$. Then, the miner sends c_w to blockchain and generates transaction data stored in the transaction pool, packs them into a block, and sends the block to all the consensus users who can verify that the keyword of secure index in the new generating block is selected from W according to Algorithm 1. If $w \in W$, it returns 1, otherwise it returns 0. The bloom filter is used to verify the authenticity of keywords, shown in Algorithm 2.

5.1.3. Phase 3: Profile Matching. Keyword search. The DR searches desired data using the DR's private key ak to generate a keyword trapdoor T_w in blockchain. The trapdoor $T_w = (T_{1,i}, T_{2,i}, T_{3,i,b})$ is generated as follows.

Choose a random vector $(\vec{V}) = \{\mu, \gamma_1, \dots, \gamma_l\}$, where random number $\gamma_1, \dots, \gamma_l \in Z_q^*$. This vector's values will be used to share μ .

For $1 \leq i \leq l$, it computes $\eta_i = (\vec{V}) \cdot N_i$. It chooses $\varepsilon_1, \dots, \varepsilon_l \in Z_q^*$ and computes $T_{1,i} = \eta_i H_1(W) P_2 + \theta_i \sigma_i \varepsilon_i P_2$, $T_{2,i} = \varepsilon_i B_i$, $\forall b \in T_r, \rho(i)$, and $T_{3,i,b} = \varepsilon_i E_{i,b}$, where T is a set of distinct attributes in N for $1 \leq i \leq l$. Then, DR sends $T_w = (T_{1,i}, T_{2,i}, T_{3,i,b})$ to blockchain.

Input: A keywords set W , an array of length m , n keywords, k hash functions $H = \{H'_1, H'_2, \dots, H'_k\}$.
Output: Bloom Filter BF .
(1) BF = a new m - bits array of n elements
(2) **for** $i = 1$ to m do
(3) $BF[i] = \text{NULL}$
(4) **for** each element $w \in W$ do
(5) **for** $i = 1$ to k do
(6) $j = H'_i(w)$.
(7) **if** $BF[j] == \text{NULL}$
(8) $BF[j] = 1$.
(9) **return** BF .

ALGORITHM 1: BuildBF (W, m, n, k, h).

Input: A keyword w , symmetric key s_k , k hash functions, $Enc_{s_k}(H) = Enc_{s_k}(\{H'_1(w), H'_2(w), \dots, H'_k(w)\})$.
Output: **if** $w \in W$, return 1, else return 0
(1) **for** each element w .
(2) **for** $i = 1$ to k do
(3) $c_j = Enc_{s_k}(H'_j(w))$.
(4) $j = De_{s_k}(c_j)$.
(5) **if** each $BF[j] == 1$.
(6) **return** 1
(7) **else** return 0

ALGORITHM 2: VerifyBF (ADO, Hcw).

Test. Assume that attributes Att associated with c_w satisfy (N, ρ) . There exists $i \in I, \{\pi_i \in Z_q^*\}$ such that $\sum_{i \in I} \pi_i N_i = (1, 0, \dots, 0)$, where N_i is i th row of N . Given an original ciphertext c_w , a keyword w , and a search token T_w , the following equation is verified:

$$e\left(I_1, \sum_{i \in I} \pi_i \left(T_{1,i} + \sum_{z \in \Delta/\rho(i)} T_{3,i,z}\right)\right) \stackrel{?}{=} e\left(\sum_{z \in \Delta} I_z, \sum_{i \in I} \pi_i T_{2,i}\right) \cdot I_2. \quad (2)$$

If (2) holds, the matching keyword w' is sent to pricing and payment smart contract. The DO sets payment according to the optimal pricing mechanism. The incentive mechanism is expected to encourage more data owners and data requesters to participate in the process of profile matching. Otherwise, it aborts.

Correctness: when the encrypted keyword is the same as the keyword in the trapdoor, the correctness of the test algorithm is verified as follows:

$$\begin{aligned} & e\left(I_1, \sum_{i \in I} \left(T_{1,i} + \sum_{z \in \Delta/\rho(i)} T_{3,i,z}\right) \pi_i\right) = e\left(\chi P_1, \sum_{i \in I} \pi_i \left(\eta_i H_1(w) P_2 + \theta_i \sigma \varepsilon P_2 + \sum_{z \in \Delta/\rho(i)} \varepsilon_i E_{i,z}\right)\right) \\ & = e\left(\chi P_1, \sum_{i \in I} \pi_i \left((\vec{V}) \cdot M_i H_1(w) P_2\right)\right) e\left(\chi P_1, \sum_{i \in I} \pi_i \left(\theta_i \sigma_i \varepsilon_i P_2 + \sum_{z \in \Delta/\rho(i)} \varepsilon_i \theta_z \sigma_i P_2\right)\right) \\ & = e(\chi P_1, \mu_1 H_1(w) P_2) e\left(\chi P_1, \sum_{i \in I} \pi_i \left(\sum_{z \in \Delta} \varepsilon_i \theta_z \sigma_i P_2\right)\right) \\ & e\left(\sum_{z \in \Delta} I_z, \sum_{i \in I} \pi_i T_{2,i}\right) \cdot I_2 = e\left(\sum_{z \in \Delta} \theta_z \chi P_1, \sum_{i \in I} \pi_i \sigma_i \varepsilon_i P_2\right) \cdot e(P_1, P_2)^{\mu_2 \chi H_1(w)} \\ & = e\left(\sum_{z \in \Delta} I_z, \sum_{i \in I} \pi_i T_{2,i}\right) \cdot I_2. \end{aligned} \quad (3)$$

Based on PBFT, if not less than 2/3 consensus users verify the new block, following the basic setting of 2/3, etc. [41], the new block will be added into the blockchain as a new valid block. In this round, the generation of a valid block marks the completion of consensus. The miner will generate a random number $R \in [0, N_c - 1]$ to determine the next miner in the next round. Then, the DO participates in pricing and sets the fees. When the DR wants to access the desired data, he/she needs to pay the fee to the corresponding DO by PPC. The specific pricing process can be seen in the following optimal pricing mechanism.

5.2. Optimal Pricing Mechanism. When a data requester matches the profile keyword in blockchain, he/she accesses the corresponding data by paying fees to the data owner. In this section, we design an optimal pricing mechanism based on a Stackelberg game. Optimal price maximizes the profits of the data owners and the data requesters. The data owners determine their own prices of data based on profit functions as the leaders. The data requesters determine the access quantity of data as followers. The Stackelberg game is between data owners and data requesters, shown in Figure 4.

5.2.1. Stackelberg Game Problem Formulation. The number of data owners is M . A set of data owners can be expressed by $\tilde{M} = \{1, \dots, M\}$. Data owners provide the desired data for the data requester. In our optimal pricing mechanism, there is a DR determining access strategy, such as access amount of data. Meanwhile, each data owner $j \in M$ determines the optimal price for corresponding data. The access amount of the data requester from data owner j is x_j . The unit price of data owner j for the data requester is p_j . We denote the access amount set and the optimal access amount set of the data requester from different data owners as $X_j = \{x_1, \dots, x_M\}$ and $X_j^* = \{x_1^*, \dots, x_M^*\}$, respectively. The unit price set and the optimal price set of the data requester from different data owners are denoted as $P_j = \{p_1, \dots, p_M\}$ and $P_j^* = \{p_1^*, \dots, p_M^*\}$, respectively. The main symbols used in an optimal pricing mechanism are shown in Table 2.

To evaluate the access quality of data, an access quality factor is denoted by q . Data owner j provides data's quality, formulated as follows:

$$Q(x_j) = q \ln(1 + x_j). \quad (4)$$

The data requester's utility is expressed as follows:

$$\begin{aligned} U_r(x_j, p_j) &= f_j a_j Q(x_j) - x_j p_j \\ &= q f_j a_j \ln(1 + x_j) - x_j p_j. \end{aligned} \quad (5)$$

The data requester gives a feedback evaluation to the data of the data owner j and obtains small fees as rewards, denoted as f_j . a_j is the accessing willingness of the data requester. The data requester maximizes its utility based on the access quantity x_j , forming its subgame, which is given as follows.

Problem 1 (data requester's subgame for data owner j):

$$\begin{aligned} \max_{x_j} U_r(x_j, p_j), \\ \text{s.t. } x_j \in [x_{\min}, x_{\max}], \quad \forall j \in M. \end{aligned} \quad (6)$$

x_{\min} is the minimum amount of accessed data, and x_{\max} is the largest amount of accessed data.

Data owner's utility can be expressed as follows:

$$U_j(x_j, p_j) = s_j(x_j p_j - x_j c_j). \quad (7)$$

Here, c_j represents the unit cost set by data owner j . The data owner's reputation score is expressed as s_j , which changes dynamically according to the data requester feedback on the data owner's data and amount of files downloaded by the data requester. $s_j \in [s_{\min}, s_{\max}]$. The terms s_{\min} and s_{\max} are lower and upper bound reputation scores, respectively. $s_{\min} < s_{\text{bad}} < s_{\text{init}} < s_{\text{good}} < s_{\max}$. s_{bad} is the bad reputation score threshold. s_{init} is the initial reputation score threshold. s_{good} is the good reputation score threshold. The data owner j maximizes its utility based on the price p_j , forming its subgame, which is given as follows.

Problem 2 (data owner's subgame for the data requester):

$$\begin{aligned} \max_{p_j} U_j(x_j, p_j), \\ \text{s.t. } p_j \in [c_j, p_{\max}], \quad \forall j \in M. \end{aligned} \quad (8)$$

The data requester can accept the maximum price, denoted as p_{\max} . Problem 1 and problem 2 constitute the Stackelberg game, which aims for finding the equilibrium points. In other words, the profit of data owner comes up to maximization when the data requester obtains its largest profit.

Definition 2. The points (x_j^*, p_j^*) are an equilibrium if it satisfies both the following two conditions:

$$U_r(x_j^*, p_j^*) > U_r(x_j^*, p_j), \quad \forall j \in M, \quad (9)$$

$$U_j(x_j^*, p_j^*) > U_j(x_j, p_j^*), \quad \forall j \in M. \quad (10)$$

To analyze the Stackelberg game, we use backward induction [28].

5.2.2. Data Requesters' Accessing Strategy in Stage II. Data owner j provides data's unit price for the data requester (i.e., p_j , for all $j \in M$). The data requester from data owner decides its optimal access strategy (i.e., x_j , for all $j \in M$).

First, we take the derivative on the data requester's utility in (5) according to x_j , which is expressed as follows:

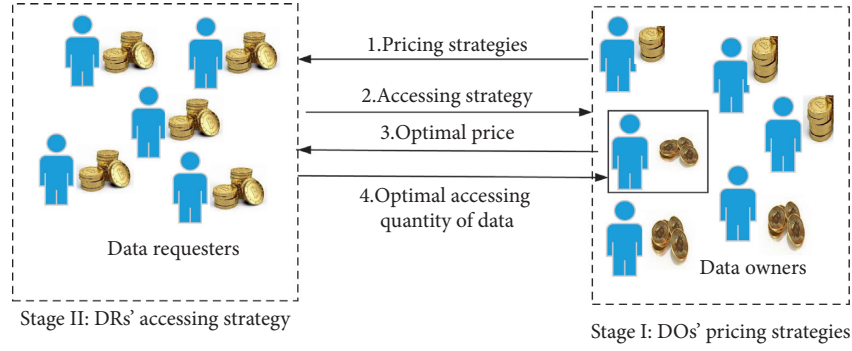


FIGURE 4: Stackelberg game-based optimal pricing mechanism.

TABLE 2: Notation table of the pricing mechanism.

Notation	Description
\tilde{M}	The set of data owners
M	The amount of data owners
x_j	The amount of accessed data
x_{\max}	The largest amount of accessed data
x_{\min}	The minimum amount of accessed data
x_j^*	The optimal amount of accessed data
X_j^*	The optimal amount set of accessed data
a_j	The accessing willingness of the data requester
s_j	The data owner's reputation score
f_j	The small fees as rewards
c_j	The cost of assessed data of data owner j .
p_j	The unit price set by data owner j .
p_{\max}	The maximum accepted price set by the data requester
p_j^*	The optimal price set by data owner j .
P_j^*	The set of optimal price

$$\frac{\partial U_r}{\partial x_j} = \frac{\partial [f_j a_j q \ln(1 + x_j) - x_j p_j]}{\partial x_j} = \frac{f_j a_j q}{1 + x_j} - p_j, \quad (11)$$

$$\frac{\partial^2 U_r}{\partial x_j^2} = \frac{\partial (f_j a_j q / (1 + x_j) - p_j)}{\partial x_j} = \frac{-f_j a_j q}{1 + x_j} < 0. \quad (12)$$

$U_r(x_j, p_j)$ is a strictly concave function shown by the above derivatives. To obtain the optimal response function, we set $[(\partial U_r)/(\partial x_j)] = 0$, which is given as follows:

$$\frac{\partial U_r}{\partial x_j} = \frac{f_j a_j q}{1 + x_j} - p_j = 0. \quad (13)$$

Then,

$$x_j^* = \frac{f_j a_j q}{p_j} - 1. \quad (14)$$

According to (14), x_j^* is data requester's optimal quantity of accessed data.

5.2.3. Data Owners' Pricing Strategies in Stage I. Data owner can obtain his/her largest profit based on the data requester's optimal access strategy. According to formulation (7)–(14), the optimal utility of data owner for the DR can be rewritten as follows:

$$\begin{aligned} U_j(x_j^*, p_j) &= s_j(x_j p_j - x_j c_j) \\ &= (s_j f_j a_j q + s_j c_j) - \left(s_j p_j + \frac{s_j f_j a_j q c_j}{p_j} \right). \end{aligned} \quad (15)$$

First, we take the derivative on data owner's utility in (15) according to p_j , which is expressed as follows:

$$\frac{\partial U_j}{\partial p_j} = \frac{\partial [(s_j f_j a_j q + s_j c_j) - (s_j p_j + s_j f_j a_j q c_j / p_j)]}{\partial p_j} = \frac{s_j f_j a_j q c_j}{p_j^2} - s_j, \quad (16)$$

$$\frac{\partial^2 U_j}{\partial p_j^2} = \frac{\partial (s_j f_j a_j q c_j / p_j^2 - s_j)}{\partial p_j} = \frac{-2s_j f_j a_j q c_j}{p_j^3} < 0. \quad (17)$$

$U_j(x_j^*, p_j)$ is a strictly concave function shown by the above derivatives. To obtain the optimal response function, we set $[(\partial U_j)/(\partial p_j)] = 0$, which is given as follows:

$$\frac{\partial U_j}{\partial p_j} = \frac{s_j f_j a_j q c_j}{p_j^2} - s_j = 0. \quad (18)$$

Then,

$$p_j^* = \sqrt{f_j a_j q c_j}. \quad (19)$$

According to (19), when f_j , a_j , q , and c_j are the active impact, it will get data owner's optimal price.

According to (7), we gain the data requester's optimal utility from data owner, given as follows:

$$\begin{aligned} U_j(x_j^*, p_j^*) &= s_j x_j^* p_j^* - s_j x_j^* c_j \\ &= \left(\sqrt{f_j a_j q} - \sqrt{c_j} \right)^2 s_j. \end{aligned} \quad (20)$$

According to (5)–(19), we are able to get the optimal utility of the data requester from data owner j , given as follows:

$$\begin{aligned} U_r(x_j^*, p_j^*) &= f_j q a_j \ln(1 + x_j^*) - x_j^* p_j^* \\ &= f_j q a_j \ln \sqrt{\frac{f_j a_j q}{c_j}} - f_j a_j q + \sqrt{f_j a_j q c_j}. \end{aligned} \quad (21)$$

By finding the equilibrium point of the game, both the data requester and the data owners can obtain their own optimal utility. A Nash equilibrium reaches between them. Meanwhile, the incentive mechanism promotes the data owners to take an active part in sharing their experiences. There will be some contradictions and conflicts of benefits if there is no balance between data requesters and data owners.

5.3. Smart Contract Design. To satisfy the system's requirements, we design smart contracts with various functions. The interactions among contracts (programmed in Solidity/footnote[https:// remix ethereum org/](https://remix.ethereum.org/)) for Ethereum include the following steps. The function is given in Algorithm 3.

KeywordCiphertext (CW): This function is called to store keyword ciphertext in blockchain. Before storing the data in blockchain, the DO sends the keyword with k hash functions to invoke VSC for verifying that the keyword is selected from a valid keyword set. If the keyword is correct, the keyword ciphertext will be packed in a block.

Pricing (Pricinglist, pricing): This function is used to formulate pricing list for blockchain. Before generating the pricing list based on an optimal pricing mechanism provided by the DO, PPC will invoke VSC for verifying the validity of the keyword. If it is valid, the DO will take part in pricing built on an optimal pricing mechanism. Then, the pricing list will be stored in blockchain.

setFee (Paymentstruct, payment): The function is called to set charges for desired data and feedback data. The fee is divided into two parts: the DO's data fee and the DR's reward fee.

Payment (account, fee): The DR calls this function to transfer the fee to the intended DO's account. Then, he/

she can access the desired data. If deposit of the DR's account is enough, it will be executed. Furthermore, PPC sends the payment result to FSC.

6. Security Proof

In this section, we analyze how the proposed scheme is able to effectively realize the goals defined in the section of system model.

6.1. Privacy Preservation. Users send data to blockchain through their blockchain account during data transmissions in edge-based IoMT. Due to the immunity characteristics of blockchain, data in the blockchain are tamper-proof. Therefore, users' sensitive information can be protected. Furthermore, the encrypted keywords are stored in blockchain. They will not divulge any information of users. Our scheme utilizes a bloom filter in smart contract to verify the validity of keywords, which reduces the unnecessary consumption. The optimal pricing mechanism is used to incentivize the DO and the DR to share their data. Malicious behaviors are prevented from getting illegal fees via the anonymous blockchain account.

6.2. Secure Match

Definition 3. Secure match. An adversary \mathcal{A} cannot distinguish the keyword from keyword ciphertext or search trapdoor.

Theorem 1. *The proposed protocol can achieve secure match in the random oracle model on the DBDH assumption.*

To avoid reinvent the wheel, we refer to [43] for the keyword secrecy game. Security proof is as follows.

Proof. The random oracles of algorithms *Private key*, *Trapdoor*, and *Test* are O_{ak} , $O_{trapdoor}$, and O_{test} , respectively. Assume that \mathcal{A} is an attacker who has advantage ε to attack the proposed chosen keyword attack game. We build a challenger \mathcal{C} . He/she plays game with \mathcal{A} to derive the solution to the DBDH problem as follows:

List ak^{list} : record $((N, \rho), ak_{(N, \rho)})$. □

6.2.1. Setup. Given a security parameter λ and an input tuple $(r_1 P_1, r_2 P_1, r_3 P_1)$, challenger \mathcal{C} generates the system master public key $MPK = (H_1, \hat{e}, g, P_1, P_2, \mu_1 P_1, \mu_1 P_2, \theta_1 P_1, \theta_1 P_2, H_1)$ and the master secret key $MSK = (\mu_1, \mu_2, \theta_1)$.

- (a) $H_1(w)$: if the query exists on H_1^{list1} in a tuple (w, κ_1) , κ_1 is returned; else, $\kappa_1 \in_R \mathbb{Z}_q^*$ is chosen, (w, κ_1) is added to H_1^{list1} , and $H_1(w) = \kappa_1$ is returned
- (b) $H_1(I_1, I_2, \{I_z\}_{z \in Att})$: if the query exists on H_1^{list2} in a tuple $(I_1, I_2, \{I_z\}_{z \in Att}, \kappa_2)$, κ_2 is returned; else,


```

Input: the keyword array Keyword, the pricing's array pricing
(1) function KEYWORDCIPHERTEXT (CW) public returns()
(2)   keyword.push(CW).
(3)   return result
(4) end function
(5) function PRICING(Pricinglist, pricing) public
(6)   query the corresponding  $c_w$ .
(7)   if the query is ok then
(8)     Pricinglist.push(price).
(9)   else
(10)    return false
(11)  end if
(12)  return result.
(13) end function
(14) function SETFEE(Paymentstruct, payemnt) only DO public
(15)  query the corresponding pricing result
(16)  if the query is ok then
(17)    paymentstruct.push(payment)
(18)  else
(19)    return false.
(20)  end if
(21)  return result.
(22) end function
(23) function PAYMENT (DR an dD RAccount, payment) returns()
(24)  compute the DO fee DO Fee and the DR fee DR Fee
(25)  query the balance of DR's account
(26)  if DR Balance > DO Fee + DR Fee then
(27)    transfer fee to DO's account and DR's account
(28)    return result.
(29)  else
(30)    return false.
(31)  end if
(32) end function

```

ALGORITHM 3: Smart Contract Algorithm.

$\kappa_2 \in_R Z_q^*$ is chosen, $(I_1, I_2, \{I_z\}_{z \in Att}, \kappa_2)$ is added to $H_1^{\text{list}2}$, and $H_1(I_1, I_2, \{I_z\}_{z \in Att}) = \kappa_2$ is returned

6.2.2. Phase 1.

- (a) $O_{ak}(N, \rho)$: Since challenger \mathcal{C} has knowledge of μ_1 and $\theta_{\rho(i)}$, it can construct private key corresponding to any (N, ρ) , and next, the tuple is added to ak^{list} . When (N, ρ) satisfies Att^* , \perp is output.
- (b) $O_{\text{trapdoor}}((N, \rho), w)$: If (N, ρ) matches Att^* , challenger \mathcal{C} outputs \perp . Else, \mathcal{C} can set $T_{2,i}, T_{3,i,b}$ as in O_{ak} . \mathcal{C} chooses a random vector (\vec{V}) with the first element μ and next sets $\eta_i = (\vec{V}) \cdot N_i$. It chooses $\varepsilon_1, \dots, \varepsilon_l \in Z_q^*$ and constructs the trapdoor T_w as in the real scheme.
- (c) $O_{\text{test}}((N, \rho), w)$: If the attribute set Att associated with c_w is Att^* , \mathcal{C} outputs \perp . Else, \mathcal{C} can always compute a trapdoor T_w as in O_{trapdoor} . Then, it proceeds to the test easily. If the test holds, \mathcal{C} outputs 1 and 0 otherwise.

6.2.3. *Challenge*. \mathcal{A} outputs w_1^* and w_0^* . Challenger \mathcal{C} sets the original challenge ciphertext as follows:

- (a) $I_1^* = r_1 P_1$, $I_z^* = r_1 r_2 P_1$, and $I_2^* = (g^{H_1(w)})^{r_3}$ are set.
- (b) A random coin $b \in (0, 1)$ is flipped, and an H_1 query on (w_b^*) is issued to achieve κ_1^* .
- (c) An H_1 query on $H_1(I_1^*, I_2^*, \{I_z^*\}_{z \in Att^*})$ is issued to achieve κ_2^* .
- (d) The challenge original ciphertext is output as $c_w^* = (Att^*, \{I_1^*, I_2^*, \{I_z^*\}_{z \in Att^*}\})$.
- (e) If $Q = r_1 r_2 r_3 P_1 \in G_1$, then c_w^* generated a valid ciphertext in which $r_1' = r_1$ and $r_2' = r_2 r_3$. After that, \mathcal{C} sends (c_w^*, T_w^*) to \mathcal{A} .

6.2.4. *Phase 2*. The phase is the same as the *Phase 1*.

6.2.5. *Guess*. \mathcal{A} outputs a guess bit $b!$, if $b = b!$, and \mathcal{C} outputs 1; else, it outputs 0.

In the guess phase, if $Q \neq r_1 r_2 r_3 P_1$, then we have the following:

$$\Pr[\mathcal{A}(P_1, r_1 P_1, r_2 P_1, r_3 P_1, Q) = 1] = \frac{1}{2}. \quad (22)$$

In addition,

$$\begin{aligned}
& \Pr[\mathcal{A}((P_1, r_1P_1, r_2P_1, r_3P_1, Q) = 1)] \\
&= \Pr[\mathcal{A}(P_1, r_1P_1, r_2P_1, r_3P_1, Q) = 1 | A_{wins}] \Pr[A_{wins}] \\
& \quad + \Pr[\mathcal{A}(P_1, r_1P_1, r_2P_1, r_3P_1, Q) = 1 | \overline{A_{wins}}] \Pr[\overline{A_{wins}}] \\
&= 1 \cdot \varepsilon''(\lambda) + \frac{1}{2}(1 - \varepsilon''(\lambda)) = \frac{\varepsilon''(\lambda)}{2} + \frac{1}{2}.
\end{aligned} \tag{23}$$

Consequently,

$$\text{Adv}_{\mathcal{A}}^{DBDH}(\lambda) = \frac{\varepsilon''(\lambda)}{2} = \varepsilon'. \tag{24}$$

Thus, the probability of \mathcal{A} winning the keyword secrecy game is ε' at least.

6.3. Fairness and Incentive. The DO's attribute key is used to encrypt the profile keywords. The bloom filter and smart contract can verify the authenticity of keywords. By this way, only valid keywords can be uploaded to blockchain. To search the desired keyword, DR must generate a search trapdoor according to his/her selected access structure. Thus, other entities cannot obtain any information about keywords and matching results during the process of keyword searching. Attackers cannot learn any information from encrypted keywords and trapdoors. Therefore, our scheme can achieve secure match.

6.4. Access Control. Our scheme utilizes key-policy attribute-based encryption (KP-ABE) in the proposed protocol. The keyword w is encrypted by the DO's attribute set Att . The DR uses his/her key ak to generate a search trapdoor T_w , which is related to the access structure. It is used for searching the matching keyword. Only when the attribute Att of keyword ciphertext c_w satisfies the access structure of search trapdoor T_w , the DR can access the desired data after calling pricing and payment smart contract (PPC) to transfer fees to a specific DO's account. If the DR fails to pay fees to the DO's account, he/she cannot query the intended data. Thus, DO is able to control the access of his/her data.

7. Implementation and Performance Evaluation

In this section, we implement the proposed algorithms in a simulated edge-based IoMT environment with Java programming and JPBC library. We deploy the designed smart contracts on Ethereum test platform. Firstly, we introduce the parameter settings. We compare the security properties of our solution with other relevant solutions. Then, the computational overhead and communication overhead are analyzed for the proposed protocol. We design the smart contracts and evaluate the performance of the designed smart contracts on Ethereum test platform. Finally, we evaluate the performance of the proposed optimal pricing mechanism.

7.1. Parameter Settings and Platform. The system security parameter $\lambda = 128$. For some prime $p = 3 \bmod 4$, we utilize type A pairing on the elliptic curve $y^2 = x^3 + x$ over the field F_p . The cryptographic primitives are implemented using JPBC library and Java on a laptop computer with Intel (R) Core (TM) i5-7400 CPU @3.00 GHz, 8 GB RAM, and Microsoft Windows 10 operating system.

In addition, we employ Ganache (client version) to build a local test chain on Linux system. We use solidity language to write data into smart contracts and then upload it to blockchain. Smart contract framework and solidity compiler are truffle @0.5.0 and solc @0.5.0, respectively. To gain time consumption of publishing smart contracts, we utilize Web3js library of Nodejs to interact with smart contracts on the blockchain and test the time cost of marking transactions. Due to the limited space, the specific deployment process is skipped.

7.2. Comparisons of Security Properties. We compare the security properties of our scheme with other matching schemes in Table 3. The comparison results indicate that the proposed scheme is capable of providing a promising solution to improve profile matching service in edge-based IoMT scenarios [25, 42, 43]. Their scheme does not achieve security properties of *blockchain-based* and *fairness and incentive*. However, our scheme can provide all of the security properties.

7.3. Communication Overhead and Computational Cost. In edge-based IoMT scenarios, the communication and computation resources are constrained. In this subsection, we show the improved performances of the proposed scheme. We denote $|G_1|$, $|G_2|$, $|G_T|$, and $|Q|$ as the size of elements in G_1 , G_2 , G_T , and Z_p . The communication overhead is caused by index generation phase and keyword search phase, shown in Table 4. In index generation phase, DO sends c_w to blockchain for searching data, and the total length is $(n+1)|G_1| + |G_T|$ bytes. The DR sends search trapdoor T_w using the secret key to blockchain for searching the desired data, and the total length is $3|G_2|$ bytes during the process of keyword search. We compare the communication overhead with [42, 43], shown in Table 4. As can be seen from Table 4, the index generation overhead of the proposed scheme is lower. In addition, the overhead of keyword search phase in [42, 43] is higher than our proposed scheme.

We compare the computational overhead in Table 5. The algorithm *SystemInit* simulates system setup phase. The generated keyword ciphertext is simulated by the algorithm *Encrypt*. Furthermore, the algorithm *Trapdoor* generates secret key and search trapdoor for the DR. The algorithm *Test* is used to test whether the keyword ciphertext and trapdoor match. From Table 5, we observe that our computational overhead is higher than that in [43] during the process of the algorithm *Test*. Nonetheless, our scheme's computational overhead is lower than other two schemes.

TABLE 3: Comparison of security properties.

Properties	[25]	[42]	[43]	The proposed
Blockchain integration	√	×	×	√
Privacy preservation	√	√	√	√
Secure match	√	×	×	√
Fairness and incentive	×	×	×	√
Access control	√	√	√	√

√: the scheme supports this property; ×: the scheme does not support this property.

TABLE 4: Communication overhead of proposed protocol.

Phases	The proposed	Cui et al. [42]	Miao et al. [43]
Index generation	$(n + 1) G_1 + G_T $	$ G_1 + (5n + 1) G $	$(2n + 1) G + G_T $
Keyword search	$3l G_2 $	$(6l + 2) G + M $	$(4n + 3) G + Q $

TABLE 5: Computational overhead of cryptographic algorithms (in ms).

	Algorithms	SystemInit	Encrypt	Trapdoor	Test
The proposed	Average time	152	547	523	368
	Max time	665	651	592	376
	Min time	23	527	482	361
Cui et al. [42]	Average time	232	550	696	431
	Max time	746	618	731	457
	Min time	97	513	672	420
Liu et al. [43]	Average time	369	610	626	321
	Max time	766	650	700	367
	Min time	263	582	581	290

TABLE 6: Time consumption of transactions.

Transactions	Cipheretext	Pricing	Feedback
Average time (ms)	98.8	354.84	79.49
Max time (ms)	255	469	255
Min time (ms)	82	251	46
Gas cost (ether)	0.014663	0.0354413	0.0103504

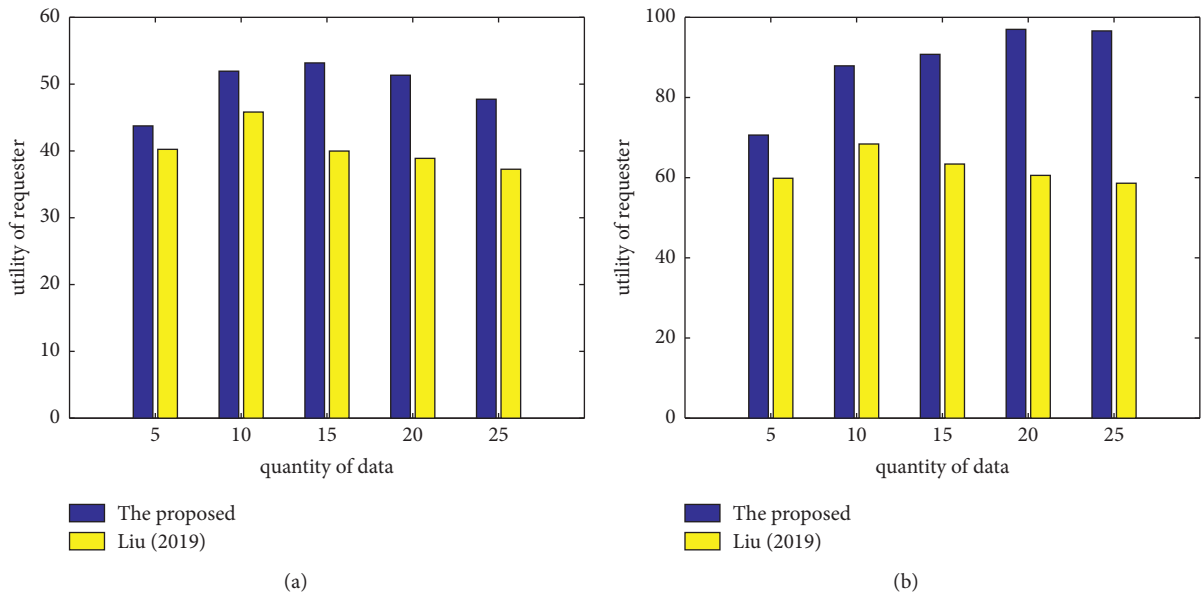


FIGURE 5: Continued.

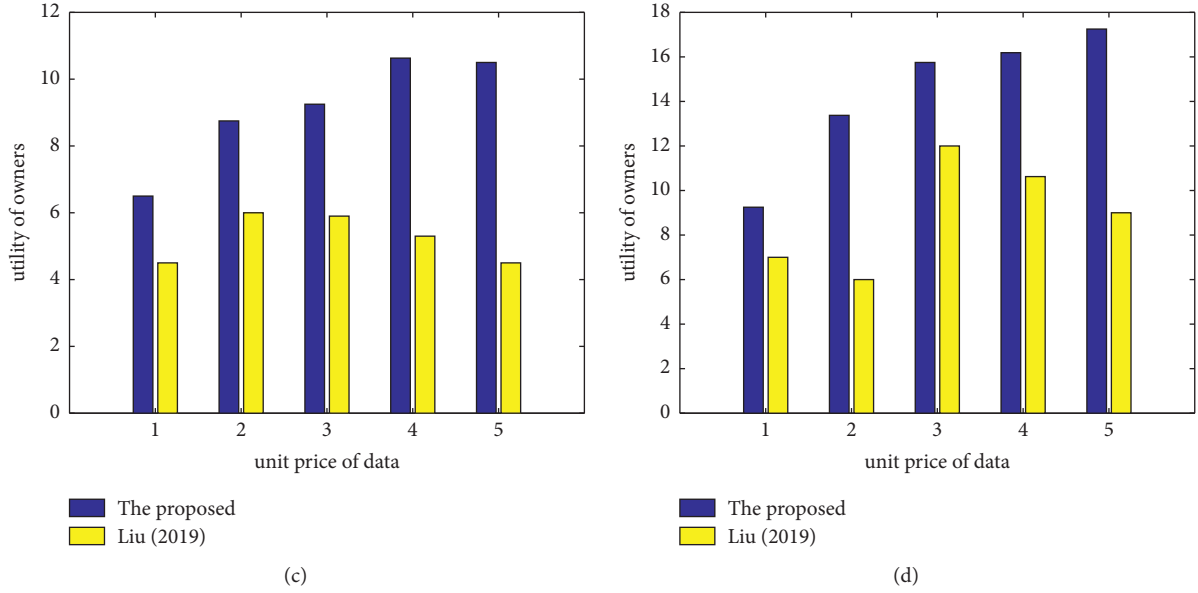


FIGURE 5: Computational cost taken by (a) $a_j = 20$, (b) $a_j = 30$, (c) $a_j = 20$, and (d) $a_j = 30$.

7.4. Time Consumption of Smart Contracts and Resource Consumption of Nodes. Because the length of data affects the time consumption of transmitting a transaction to blockchain, we firstly discuss its length. According to Section 7.3, the length of index generation and keyword search is $(n + 1)|G_1| + |G_T|$ and $3|G_2|$. As can be seen from Table 6, we see that the average time consumption of sending a transaction *KeywordCipherertext* to blockchain is 98.8 ms, a transaction *Pricing* is 354.84 ms, and a transaction *Feedback* is 79.49 ms. The gas consumption of transaction *KeywordCipherertext* is 0.0146637 ether, transaction *Pricing* is 0.03544138 ether, and transaction *Feedback* is 0.01035046 ether.

7.5. Performance Analysis of Pricing Mechanism. We evaluate the performance for data owners and data requesters under the proposed optimal pricing mechanism, as shown in Figure 5. We set some parameter values, denoted as follows: $p_{\max} = 5$, $x_{\min} = 10$, $x_{\max} = 100$, $s_j \in [0, 1]$, and $f_j \in [1, 5]$. The parameter a_j represents the data requester's willingness to access the data. When a_j values are different, we simulate the utility of data owners and data requesters, where a_j is equal to B in [44]. In Figures 5(a) and 5(b), we compare optimal pricing mechanism with independent pricing scheme in [44]. We find that the amount of accessed data increases and data requester's utility increases. It shows that in our scheme, the data requester can have more utility than Liu's scheme in [44]. Figure 5(c) and 5(d) shows that the data owners can get more utility with the increase in unit price. Moreover, the results show that the data owners get more profits in our scheme than Liu's scheme in [44] when the accessing willingness of requester increases.

8. Conclusion and Future Work

In this work, we introduce a new blockchain-based profile matching scheme by utilizing KP-ABE algorithm and bloom filter, which guarantees privacy preservation and security of health data in edge-based IoMT. Firstly, we present a system framework based on blockchain for profile matching among different users. Secondly, we design a consensus mechanism for proposed blockchain to achieve the consensus of the system. Thirdly, smart contract with an optimal pricing mechanism is designed to formulate pricing list and encourage more users to participate in the system. We evaluate the performance of communication overhead. We employ JPBC library to evaluate computational cost of the proposed protocol, compared with other schemes. Finally, we deploy the smart contracts on Ethereum platform and test the time consumption of smart contracts.

In our future work, we plan to deploy smart contracts on Hyperledger Fabric and store original data using encryption algorithm in IPFS for profile matching, which has a potential to improve the performances in edge-based IoMT scenarios.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62072005 and the Natural Science Foundation of Anhui Province under Grant 2108085Y22.

References

- [1] D. He, N. Kumar, H. Wang, L. Wang, K.-K. R. Choo, and A. Vinel, "A provably-secure cross-domain handshake scheme with symptoms-matching for mobile healthcare social network," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 633–645, 2018.
- [2] W. Tang, J. Ren, and Y. Zhang, "Enabling trusted and privacy-preserving healthcare services in social media health networks," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 579–590, 2019.
- [3] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: multiaccess edge computing for 5G and Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6722–6747, 2020.
- [4] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor Authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, p. 1, 2020.
- [5] A. Asheralieva and D. Niyato, "Distributed dynamic resource management and pricing in the IoT systems with blockchain-as-a-service and UAV-enabled mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1974–1993, 2020.
- [6] K. Liu, W. Chen, Z. Zheng, Z. Li, and W. Liang, "A novel debt-credit mechanism for blockchain-based data-trading in Internet of vehicles," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9098–9111, 2019.
- [7] Y. Qu, L. Gao, T. H. Luan et al., "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5171–5183, 2020.
- [8] L. Cui, Y. Qu, G. Xie et al., "Security and privacy-enhanced federated learning for anomaly detection in IoT infrastructures," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3492–3500, 2022.
- [9] Q. Wang, D. Wang, C. Cheng, and D. He, "Quantum2FA: efficient quantum-resistant two-factor Authentication scheme for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2021.
- [10] S. U. Amin and M. S. Hossain, "Edge intelligence and Internet of Things in healthcare: a survey," *IEEE Access*, vol. 9, pp. 45–59, 2021.
- [11] I. V. Pustokhina, D. A. Pustokhin, D. Gupta, A. Khanna, K. Shankar, and G. N. Nguyen, "An effective training scheme for deep neural network in edge computing enabled Internet of medical Things (IoMT) systems," *IEEE Access*, vol. 8, Article ID 107112, 2020.
- [12] H. Zhong, Y. Zhou, Q. Zhang, Y. Xu, and J. Cui, "An efficient and outsourcing-supported attribute-based access control scheme for edge-enabled smart healthcare," *Future Generation Computer Systems*, vol. 115, pp. 486–496, 2021.
- [13] R. Akkaoui, X. Hei, and W. Cheng, "EdgeMediChain: a hybrid edge blockchain-based framework for health data exchange," *IEEE Access*, vol. 8, Article ID 113467, 2020.
- [14] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "BEdgeHealth: a decentralized architecture for edge-based IoMT networks using blockchain," *IEEE Internet of Things Journal*, vol. 8, no. 14, Article ID 11743, 2021.
- [15] T. Muhammed, R. Mehmood, A. Albeshri, and I. Katib, "UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities," *IEEE Access*, vol. 6, Article ID 32258, 2018.
- [16] R. Guo, G. Yang, H. Shi, Y. Zhang, and D. Zheng, "O3-R-CP-ABE: an efficient and revocable attribute-based encryption scheme in the cloud-assisted IoMT system," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8949–8963, 2021.
- [17] M. Li, N. Ruan, Q. Qian, H. Zhu, X. Liang, and L. Yu, "SPFM: scalable and privacy-preserving friend matching in mobile cloud," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 583–591, 2017.
- [18] D. He, N. Kumar, H. Wang, L. Wang, K.-K. R. Choo, and A. Vinel, "A provably-secure cross-domain handshake scheme with symptoms-matching for mobile healthcare social network," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 633–645, 2018.
- [19] V. Malamas, P. Kotzanikolaou, T. K. Dasaklis, and M. Burmester, "A hierarchical multi blockchain for fine grained access to medical data," *IEEE Access*, vol. 8, Article ID 134393, 2020.
- [20] M. Wang, Y. Guo, C. Zhang, C. Wang, H. Huang, and X. Jia, "MedShare: a privacy-preserving medical data sharing system by using blockchain," *IEEE Transactions on Services Computing*, 2021.
- [21] P. Zhang and M. Zhou, "Security and trust in blockchains: architecture, key technologies, and open Issues," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 3, pp. 790–801, 2020.
- [22] K. Gai, J. Guo, L. Zhu, and S. Yu, "Blockchain meets cloud computing: a survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2009–2030, 2020.
- [23] L. Yang, M. Li, H. Zhang, H. Ji, M. Xiao, and X. Li, "Distributed resource management for blockchain in fog-enabled IoT networks," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2330–2341, 2021.
- [24] W. Meng, W. Li, S. Tug, and J. Tan, "Towards blockchain-enabled single character frequency-based exclusive signature matching in IoT-assisted smart cities," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 268–277, 2020.
- [25] F. Yang, Y. Wang, C. Fu, C. Hu, and A. Alrawais, "An efficient blockchain-based bidirectional friends matching scheme in social networks," *IEEE Access*, vol. 8, Article ID 150902, 2020.
- [26] Y. Jiao, P. Wang, S. Feng, and D. Niyato, "Profit maximization mechanism and data management for data analytics services," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2001–2014, 2018.
- [27] G. El Rahi, S. R. Etesami, W. Saad, N. B. Mandayam, and H. V. Poor, "Managing price uncertainty in prosumer-centric energy trading: a prospect-theoretic stackelberg game approach," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 702–713, 2019.
- [28] X. Chen, C. Tang, Z. Li, L. Qi, Y. Chen, and S. Chen, "A pricing approach toward incentive mechanisms for participant mobile crowdsensing in edge computing," *Mobile Networks and Applications*, vol. 25, no. 4, pp. 1220–1232, 2020.
- [29] K. Liu, X. Qiu, W. Chen, X. Chen, and Z. Zheng, "Optimal pricing mechanism for data market in blockchain-enhanced Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9748–9761, 2019.
- [30] Y. Li, J. Wan, R. Chen et al., "Top-k vehicle matching in social ridesharing: a price-aware approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 3, pp. 1251–1263, 2021.
- [31] Z. Xiong, J. Kang, D. Niyato, P. Wang, and H. Poor, "Cloud/edge computing service management in blockchain networks: multi-leader multi-follower game-based ADMM for pricing,"

- IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 356–367, 2020.
- [32] W. Zhang, Z. Hong, and W. Chen, “Hierarchical pricing mechanism with financial stability for decentralized crowd-sourcing: a smart contract approach,” *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 750–765, 2021.
- [33] J. Nie, J. Luo, Z. Xiong, D. Niyato, and P. Wang, “A stackelberg game approach toward socially-aware incentive mechanisms for mobile crowdsensing,” *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 724–738, 2019.
- [34] B. H. Bloom, “Space/time trade-offs in hash coding with allowable errors,” *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.
- [35] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar, “A survey on privacy protection in blockchain system,” *Journal of Network and Computer Applications*, vol. 126, no. 1, pp. 45–58, 2019.
- [36] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, “A survey of distributed consensus protocols for blockchain networks,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.
- [37] W. Li, M. Cao, Y. Wang, C. Tang, and F. Lin, “Mining pool game model and nash equilibrium analysis for PoW-based blockchain networks,” *IEEE Access*, vol. 8, Article ID 101049, 2020.
- [38] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, “Proof-of-Stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities,” *IEEE Access*, vol. 7, Article ID 85727, 2019.
- [39] F. Yang, W. Zhou, Q. Wu, R. Long, N. N. Xiong, and M. Zhou, “Delegated proof of Stake with downgrade: a secure and efficient blockchain consensus algorithm with downgrade mechanism,” *IEEE Access*, vol. 7, Article ID 118541, 2019.
- [40] L. Xu, W. Li, F. Zhang, R. Cheng, and S. Tang, “Authorized keyword searches on public key encrypted data with time controlled keyword privacy,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2096–2109, 2020.
- [41] A. Zhang and X. Lin, “Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain,” *Journal of Medical Systems*, vol. 42, no. 147, pp. 140–214, 2018.
- [42] H. Cui, Z. Wan, R. H. Deng, G. Wang, and Y. Li, “Efficient and expressive keyword search over encrypted data in cloud,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 409–422, 2018.
- [43] Y. Miao, R. Deng, X. Liu, K. Choo, H. Wu, and H. Li, “Multi-authority attribute-based keyword search over encrypted cloud data,” *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 4, pp. 1667–1680, 2021.
- [44] K. Liu, X. Qiu, W. Chen, X. Chen, and Z. Zheng, “Optimal pricing mechanism for data market in blockchain-enhanced Internet of Things,” *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9748–9761, 2019.

Research Article

A Novel Trusted Software Base for Commercial Android Devices Using Secure TF Card

Yuting Zhou ¹, Bo Zhao ¹, and Yang An ²

¹School of Cyber Science and Engineering, Wuhan University, Wuhan, China

²School of Computer Science, Wuhan University, Wuhan, China

Correspondence should be addressed to Bo Zhao; zhaobo409@126.com

Received 15 September 2021; Revised 17 December 2021; Accepted 26 January 2022; Published 16 February 2022

Academic Editor: Ding Wang

Copyright © 2022 Yuting Zhou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the help of edge computing, the Internet of Things (IoT) provides users with efficient data transmission and processing capabilities. As a main control device of the IoT and the communication portal of edge computing, user terminals represented by Android devices have potential security risks in IoT. Trusted computing is a universal method to construct trusted environment for computing platforms. However, due to the strict space, cost, and power limitations, commercial Android devices would not be applicable to implement a dedicated onboard chip or the software Trusted Platform Module (TPM) by modifying its firmware. We have designed a practical Trusted Software Base (TSB) for mobile devices to enhance their security. By using the secure TF card as the hardware to provide secure storage and cryptographic capabilities, we implement the trusted boot and trust extension for applications on a commercial device to verify the feasibility of the TSB to ensure a trusted environment for users. Our implementation does not require any modification to the firmware or any additional hardware other than the secure TF card. Experimental evaluation shows that our method can provide trusted computing capability for commercial Android devices with low performance overheads.

1. Introduction

The Internet of Things (IoT) uses edge computing close to the geographic location of users or data sources to provide more reliable and fast services. Mobile devices serve as the main IoT control center and the communication portal for mobile edge computing. Especially, existing works reveal that Android devices face many practical risks [1–3]. For example, in the scene of the Internet of Vehicles, if the vehicle control application of the Android device is tampered with, the vehicle will become unavailable and the privacy information may also be leaked.

Android's dominance in the global smartphone market has resulted in a significant increase in the number of malware, which pose a huge threat to the Android devices' security. Researches mainly focused on the detection and analysis of malicious applications [4–7]. These approaches contribute to the protection of applications and data on the Android platform, but the security during trusted boot is not guaranteed. Due to the lack of trusted hardware, the security

software on the device may also be maliciously tampered with.

Trusted computing technology establishes a specific integrity measurement mechanism in computing system [8] and builds a chain of trust from root of trust to the whole software and hardware [9]. It enables the computing platform to have the ability to distinguish whether the program code is trusted when it is running, thereby establishing effective prevention methods and measures for untrusted program codes [10]. The most extensively implemented form of trusted computing on end-consumer devices is the Trusted Platform Module (TPM), a dedicated hardware chip that offers facilities for cryptographic coprocessing, protected credentials, secure storage, or even the attestation of its host platform's state. The special benefits provided by TPM have always relied on the implementation of TPM as a "local trusted third party" on the device. The use of TPM to implement the PC trusted computing platform has been very mature, and in the latest Windows 11, Microsoft requires a TPM module in the device [11].

However, limited by space, cost, and power consumption, the implementation of TPM on mobile devices is to build a hardware-based trusted execution environment (TEE) [12–14], like ARM TrustZone where the TPM is implemented as protected software application inside the TEE. But there are still some attacks against trusted software in TEE [15–19].

On commercial mobile devices, the firmware is customized by the vendor and hard to be modified. The solutions mentioned above have only been verified in the development board or simulation environment. Some need to add onboard hardware or modify the firmware. Therefore, for users who cannot trust both the vendor and applications in the market, a security protection solution that can be conveniently used on commercial Android devices is critical. So, commercial device users need a convenient security solution without changing the structure of the device.

In order to solve these problems from a new perspective, in this paper, we offer an alternate implementation of the TPM based on the secure TF card combined with a well-designed Trusted Software Base (TSB) [20] to achieve the same capabilities as TPM without any changes to the hardware structure and firmware of commercial Android devices. A secure TF card as secure hardware should have encryption and secure storage capabilities. Although the SoC in the device already has encryption instructions, the encryption engine and secure storage space of the dedicated hardware can ensure the security of the calculation process and the keys. We implement our TSB on Xiaomi Mi Pad 4 and use the JW1292T8I-SDK TF card produced by Chengdu JAVEE Microelectronics. Applying a star style trust measurement model, we extend the trust from the TF card to the device when booting and ensure the trust of the application when launching or installing. To summarize, our main contributions are as follows:

- (i) Under the condition that changes can hardly be made to the firmware and hardware of commercial Android devices, we use secure TF card to achieve a trusted boot. Using a secure TF card as physically separate hardware to perform cryptographic calculations and securely store critical information will be more secure than the implementations without dedicated hardware.
- (ii) Besides boot process, our TSB guarantees the safety of applications for users. Since Android devices do not reboot frequently and are mainly threatened by malicious software, we design TSB so that it can monitor and control the launch and installation of applications in real time to prevent malware.
- (iii) By using the modified star style trust measurement model, TSB will take over the computing task after the Primary Trust Base confirms that it is trusted. This reduces the burden of the TF card and provides an accelerated measurement speed.
- (iv) We conducted experiments using Android devices that are available on the market rather than a reference board. The evaluation shows that the device can complete a trusted boot in about two minutes.

Also, the TSB has caused a minimum delay of less than 0.3 s for the application launch or installation and has little drag on the performance.

The rest of this article is organized as follows. Section 2 shows the background and related works. Section 3 presents an overview of the TSB design. Section 4 introduces our Android TSB implementation based on a secure TF card. Section 5 presents the experimental evaluation. Section 6 concludes this paper.

2. Background and Related Works

2.1. Background. Trusted computing technology guarantees the security of information systems by building a chain of trust. The way to obtaining trust is to examine two interacting entities. When there is no direct interaction between two entities, it is necessary to build a chain of trust through a third entity to extend trust. TCG had defined three roots of trust: root of trust for measurement (RTM), root of trust for storage (RTS), and root of trust for report (RTR) [21]. The RTM is responsible for performing reliable integrity measurements, the RTS maintains integrity digests values, and the RTR can reliably report information held by the RTS.

As seen in Figure 1, a serial chain connects the BIOS Boot Block, BIOS, OS Loader, and OS. BIOS Boot Block is the RTM. The hash value is generated from the concatenation of the current hash value and a new value. Then, the hash value is saved in the platform configuration register (PCR) as the new integrity measurement value. Equation (1) describes this process, where the symbol \parallel stands for concatenation and the $NewValue$ stands for the content of entity i .

$$NewPCR_i = HASH(OldPCR_i \parallel NewValue_i). \quad (1)$$

However, there will be losses in the process of trust transmission [22]. The longer the path of trust transmission is, the greater the loss may be. In addition, the traditional chain of trust measurement model is inflexible to add and delete components. Furthermore, embedded systems have strict requirements on functions, reliability, cost, size, and power consumption. Therefore, as shown in Figure 2, the star style trust measurement model is proposed [23], which adopts a star style chain of trust structure on embedded devices.

According to the star trust structure theory, the trust extension path of the star style chain of trust is shorter and more robust compared to the TCG chain of trust, which makes it easier to add or delete the component software version upgrades in the star style chain. However, the disadvantage is that the root node needs to measure the integrity of each node in the boot chain and make a judgment, so the computing pressure on the root node is heavy. Considering that hardware and firmware of commercial devices can hardly be changed, we modified the star trust structure to better adapt to TF cards and commercial Android devices. In our design, the TSB is the central node of this structure, and the details are shown in the next section.

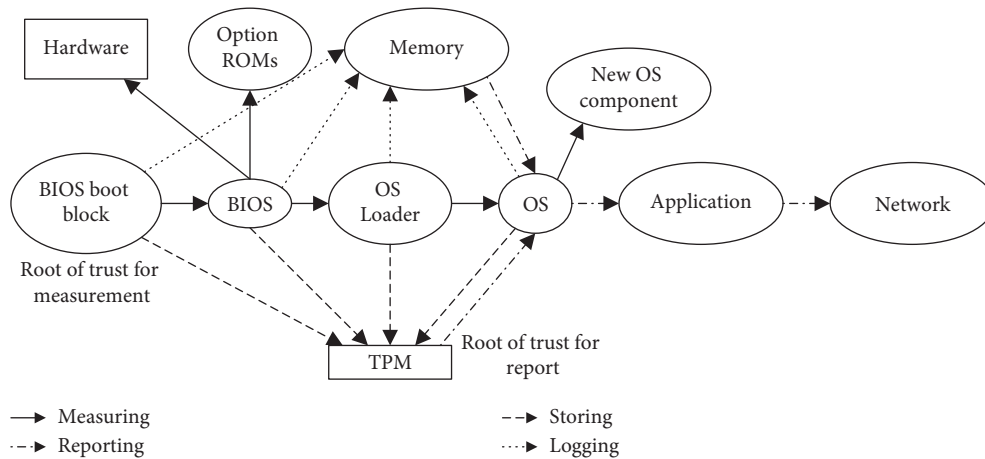


FIGURE 1: TCG chain of trust structure.

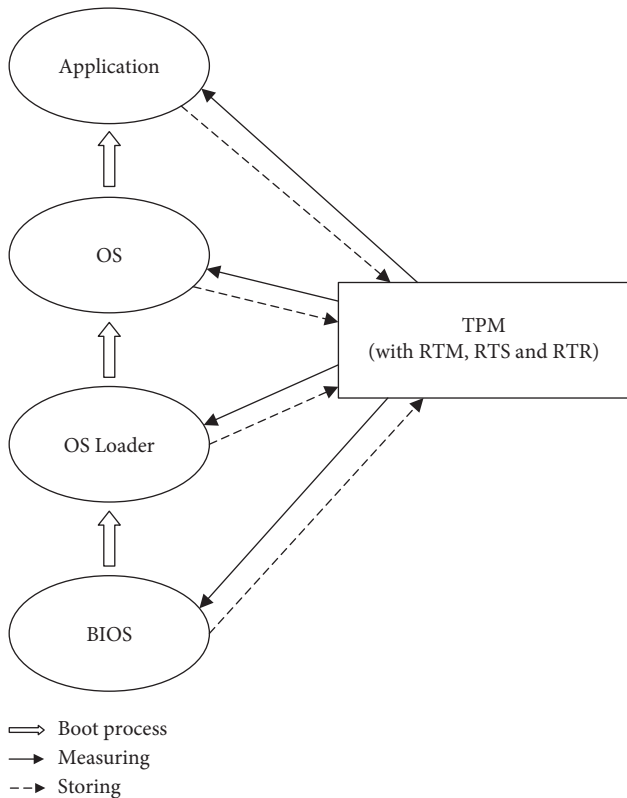


FIGURE 2: Star style chain of trust structure.

2.2. Related Works. Researchers have actively investigated how to bring trusted computing to mobile platforms. As mentioned earlier, Zhao et al. proposed an embedded system-TPM and a star style trust measurement model [23] to address the specific problems of the mobile domain. They mentioned that this model is better suited to the working environment of embedded devices and applied bus arbitration technique to manage the TPM and the embedded CPU to improve the security and efficiency. This work inspired our designs on mobile devices with limited resources, which draws on star style trust measurement model. In order to apply trusted computing technology in the mobile field, the Trusted

Computing Group (TCG) developed the Mobile Trusted Module (MTM) specifications [24], which has an impact on the formulation of the TPM2.0 specification [25]. The TPM 2.0 mobile reference architecture [26] proposes solutions such as virtualization and hardware-based isolation to implement TPM on mobile devices. Ekberg et al. presented onboard credentials [12] to safely open access to TEE functionality on mobile devices. McGillion et al. described Open-TEE [13], a virtual, hardware-independent TEE implemented in software on mobile devices. Raj et al. presented fTPM [14], an end-to-end implementation of a software-only TPM using ARM TrustZone, which provides security guarantees similar, although not identical, to a discrete TPM chip.

However, due to the lack of dedicated hardware, attacks against trusted software in TEE can be implemented. Lipp et al. demonstrated cross-core caching attacks on non-rooted devices to steal secrets in TrustZone [15]. Machiry et al. discovered the boomerang [16] vulnerability in the TEE in the mobile platform, which allows untrusted applications to leverage the TEE’s privileged position to perform the operations on its behalf. Ning and Zhang [17] discovered a security flaw in the arm debugging architecture, which can lead to arbitrary access to TrustZone.

Chakraborty et al. implemented a TPM called simTPM [27] without additional hardware by implementing TPM2.0 functionality on a subscriber identity module (SIM) card. Their work made it possible to use dedicated hardware on mobile devices to implement TPM-related functions.

The TSS specification [28] defines the way software at different layers uses TPM functionality, but does not establish a unified software architecture to provide trust support for devices.

Shi defined the overall trust support for system software by proposing the idea of Trusted Software Base (TSB) [20], which inspired the work of this paper to some extent.

3. Overview

3.1. Threat Model. In our threat model, the attacker needs to control the installation of malware on the victim’s device in order to be able to carry out the attack. This is a very

common threat model in the field of mobile security, as in previous work [29–31]. On Android devices, it is easy for attackers to entice victims to download malicious applications that they have prepared in advance.

As the control and proxy device of the IoT devices, mobile phones will cause the IoT devices to deviate from the preplanned work plan once they are controlled or unusable. Attackers entice victims to install and use malicious software in order to damage the Android device itself and cause information leakage of the device or IoT device. Malware can also modify the IoT management application itself or its configuration on the Android device to attack the IoT system. Attackers can also destroy the availability of IoT devices through mobile phones infected with malware, such as denial of service attacks, energy exhaustion attacks, and even directly shut down the IoT system.

3.2. Trusted Software Base and TF Card. TSB is the fundamental link that connects trusted hardware to user space applications [20]. TSB protects user space applications and manages TPM and undertakes the extension of the TPM trust chain. With the support of trusted hardware, TSB implements security capabilities in the host system through measurement and interception.

In Figure 3, TSB is composed of a defense mechanism and an underpinning mechanism. The defense mechanism includes Measurement Mechanism, Control Mechanism, Decision Mechanism, Baseline Repository, and Primary Trust Base. The TF card provides the foundation for effective TSB operation by providing a tamper-resistant hardware root of trust. Our TSB provides the following two services to ensure the trusted environment of the device. First, TSB measures the boot chain of the system during the boot process to ensure a trusted environment after booting. Second, after the device boots up, TSB measures the launch and installation of applications in the device in real time to prevent untrusted applications from running. We will not leave any confidential information in the storage. But we do not protect the key content in the memory in this process.

The main goal of the TSB is to examine whether the host system is trusted. TSB itself must be trusted and tamper-resistant. Software measurement alone is not enough to meet the requirements of tamper-resistant [32]. We avoid using the method of changing original hardware on commercial devices. Since most Android phones that support dual sim cards are compatible with TF cards, the secure TF card is our final choice.

Our purpose of using a secure TF card is to use dedicated hardware to compute and store in the case of unchangeable hardware and firmware of commercial devices. Like TCG’s TPM specification, the secure TF card should provide secure storage, public key and symmetric key encryption engine, hash engine, and a random number generator, as shown in Figure 3. These cryptographic engines and the random number generator exist in the main control chip of the TF card in the form of hardware and provide external calling

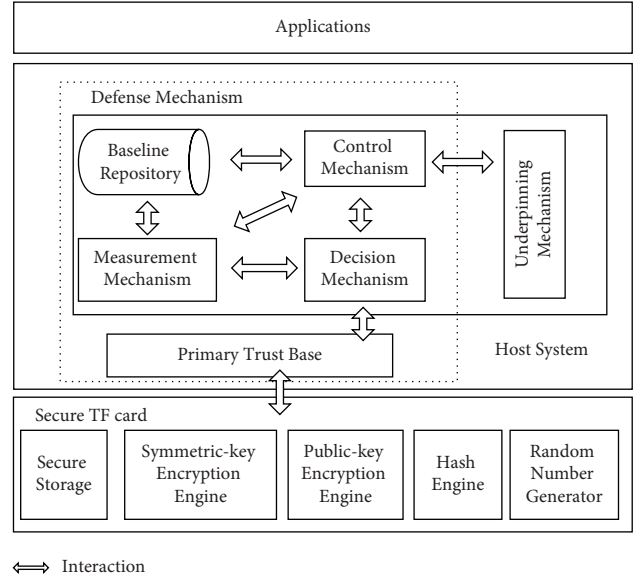


FIGURE 3: TSB structure.

interfaces. The information of the TF card used in our experiment will be described in more detail in Section 5.1.

The key used for data encryption is generated and managed by the TF card. Secure storage provides space for the password to the Baseline Repository and the whitelists formed by correctly measured values of all items to be measured and the PCR value of the boot chain. The content that needs to be safely stored in the TF card is encrypted by a symmetric key. These symmetric keys are generated and protected by the TF card. The TF card uses its built-in public key to encrypt and save these symmetric keys. Since the TF card is not mounted normally, ordinary users cannot operate the TF card without the system permissions that TSB has. In addition, the interface of the TF card is not exposed, and the authentication method is secretly shared by the TF card and the TSB. Therefore, the TSB and the TF card are strongly bound, and the outside world cannot access the TF card through this unique method.

Due to the limitations of commercial Android devices, our implementation of TSB is different from traditional methods. Nevertheless, it still contains the essential mechanisms that TSB should have. We will explain in detail the function of each part of our design and the specific process of trust extension in the next section.

4. Methodology

4.1. Trusted Software Base Structure. We will describe each part of TSB and how they work together.

4.1.1. Primary Trust Base. Primary Trust Base is the essential component in TSB and the most miniature set of scalable software with basic measurement capabilities. With the support of the underlying hardware platform (the secure TF card in this paper), it extends trust to other parts of TSB. It

completes the most fundamental measurement work when other mechanisms in TSB have not been confirmed as trusted yet. It implements the measurement operation to TSB by calling the interface of the TF card. This process passes trust to all remaining parts of TSB. In order to reduce the calculation pressure of the TF card and increase the calculation speed, we regard TSB instead of the TF card as the central node of the star style chain of trust structure. Then, TSB will be responsible for measuring all other entities in the star style chain.

4.1.2. Control Mechanism. Then, the Control Mechanism decides whether to allow, block, and audit the behavior of the measured object in accordance with the control policies and the result of the Decision Mechanism.

The control policies include (1) blocking the installation of apps that are not in the whitelist; (2) preventing the tampered application from running; and (3) prohibiting booting of devices with modified critical files.

4.1.3. Measurement Mechanism. Because of requirements for power consumption and real-time performance in Android platform, a lightweight Measurement Mechanism is required. The Measurement Mechanism is based on the integrity measurement model proposed by TCG. The measurement process refers to calculating the hash value of the object through the measurement algorithm and then comparing it with the expected hash value in the TF card. If the two values are the same, we can say that the measured object is trusted. The measurement result will be returned to the Decision Mechanism for the next decision. When initializing those hash values, Measurement Mechanism needs to compute the measured values of all objects that need to be measured and store these values in the TF card through Primary Trust Base.

4.1.4. Decision Mechanism. The main task of the Decision Mechanism is to support the composite measurement which needs to consider multiple measurement results. In contrast, judging based only on the hash value of the program is the basic measurement because the decision strategy is simple. We only want to verify the feasibility of TSB based on a TF card on commercial Android devices, so we adopted a simple decision strategy. The Decision Mechanism obtain decision rules from the Baseline Repository and determines the behavior of Control Mechanism by comparing the result from Measurement Mechanism with the related measured value in the TF card.

4.1.5. Baseline Repository. Baseline Repository provides TSB baseline information, which can be in the form of configuration information required by the TSB, including measurement plans, control policies, and decision criteria, or in the form of results output by TSB, such as measured values. In our design, the correct measured values are safely stored in the TF card and the current ones are stored in the Baseline Repository.

4.1.6. Underpinning Mechanism. Underpinning mechanism connects TSB with the outside world and helps them interact with each other. The device obtains the trust support of TSB through this mechanism. This mechanism is also responsible for the management and configuration of the trust strategy. However, the management of the trust strategy is not the focus of this paper, so there is no related display in Figure 3.

4.2. Trusted Boot. The trusted boot is an essential capability of the TSB we designed. In order to cooperate with the TSB as an application, we implement the TSB to the trusted boot as a retrospective measurement, as shown in Figure 4. We have modified the star style trust measurement model. After TSB is confirmed to be trusted, it becomes the central node of star style trust measurement structure. The measurements of any trusted boot cannot be valid unless the TF card is binding to a trusted, local RTM because an adversary may simply insert the TF card into an untrusted device and replay any wanted measurement sequence, that is, create arbitrary PCR values similar to a TPM reset attack [33, 34]. Since the TF card is not physically bound with the local RTM, we need the TF card to authenticate the RTM to ensure that it is a trusted code. In addition, we make the TSB turn off the device when the TF card is detached in case of device running without TF card.

Our TSB is designed in the form of an Android application to avoid modified firmware. So, it is necessary to wait for the device to finish booting before the TSB launches and performs retrospective measurement. In order to ensure the timeliness and validity of TSB measurement, there are two points to be ensured: (1) TSB starts immediately after the device boot and (2) TSB measurement operation is not allowed to be interrupted by other programs.

In order to make TSB run immediately after the device boot, the TSB registers a broadcast receiver to receive Android's boot broadcast. After receiving that broadcast, TSB's measurement activity starts immediately. In addition, the priority of the broadcast receiver is set to maximum value 1000 to ensure TSB to be the first to start.

The measurement operation must ensure effectiveness and atomicity so as not to be disturbed by user behavior or other processes. We set the TSB as the Android Device Administration application to implement the kiosk mode which is the lock task mode that allows Android devices to run tasks in an immersive, kiosk manner. When the device runs in lock task mode, users without permission cannot receive notifications, access unauthorized applications, or return to the home screen. We lock the screen in the measured activity, so the user cannot exit or perform any operations until the measure boot process ends.

In Figure 4, the trusted boot process of an Android device with our TSB is divided into the following parts:

- (1) After the power button is pressed, the bootloader will be loaded into the internal RAM (Random Access Memory). Then, the bootloader will initialize the memory and load the kernel. After the kernel starts, the kernel sets up the interrupt controller, memory protection, cache, and process scheduling and then

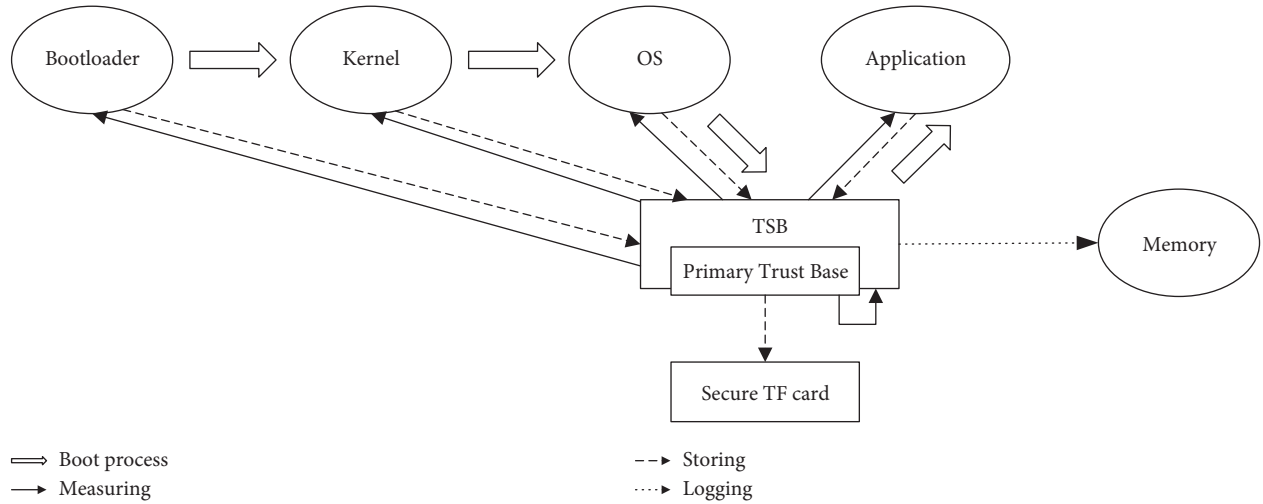


FIGURE 4: Trusted boot.

starts the user space process. After that, the init process starts, including parsing the `init.rc` script, loading the file system, starting zygote, and starting the system process. Zygote sets the Java runtime and inits memory for Android objects. Finally, the system service starts.

- (2) After the system services start, the Primary Trust Base receives the system boot broadcast, skips the system interface launcher, and directly starts the measurement of TSB by using the secure TF card. Before that, the TF card will confirm the identity of the Primary Trust Base by verifying the signature. The Primary Trust Base gets the measurement result from the TF card to decide to start TSB or shut down immediately.
- (3) After the entire TSB becomes trusted and has the password to access the Baseline Repository, it performs retrospective measurement of the boot chain, including the bootloader, system kernel, and key files in the Android system. Considering the limited computing power of the TF card and the time it takes in the measurement, all the compute processes are completed by the TSB after the Primary Trust Base has confirmed that the TSB is trusted. The hash result still needs to be compared with the measured value in the TF card.
- (4) After TSB has measured the boot chain, it exits the locked task mode and hands the device control to the user. The TSB starts a background process to monitor the launch and installation of applications in the system. TSB records each measurement result in the logs in the device storage represented by memory in Figure 4.

We marked the startup process in Figure 4 to ensure its clarity. It is worth noting that we only verified the feasibility of TSB, so there is no reporting operation in Figure 4.

Trusted boot can be formulated as the follows where i represents each entity on the boot chain. To perform

measurement on TSB, the TF card performs the hash computation of it. After that, the TSB calculates the hash value of the concatenation of its hash value and each entity i . For the convenience of understanding, although there is no specific hardware form of PCR to store the calculated hash value in TF card, we still use it to mark the measured value in the equation.

$$\begin{aligned} \text{PCR}_i &= \text{HASH}(\text{Value}_{\text{TSB}}), \\ \text{NewPCR}_i &= \text{HASH}(\text{PCR}_i \parallel \text{NewValue}_i). \end{aligned} \quad (2)$$

After the trusted boot, TSB establishes a star style chain of trust, which covers the boot chain of Android devices. If any measurement fails to pass the Decision Mechanism, the Control Mechanism shuts down the device to achieve the effect of suspending the boot process. This provides a safe environment after the device boots.

4.3. Trusted Applications. Since the Android devices do not reboot frequently and the security threats to Android devices mainly come from malicious applications, we add a permanent process to measure user applications. A flexible time segment instead of fixed period is adopted for application measurement considering the user's sensitivity to timeliness and power consumption. That means TSB measures every time an application is opened or installed, not at a specified moment. As shown in Figure 4, in order to ensure that the TSB is trusted every time the application is measured, the TSB in advance needs to pass the trust verification by the Primary Trust Base and TF card. The trust strategy is simple because we only verify the feasibility of TSB. Only when the application is in the whitelist and has the same measured value as in the TF card can it be installed and launched.

TSB uses Android's accessibility service to monitor user clicks and obtain UI information to get the name of the application launched. Then, it measures the corresponding application file and compares it with the measured value in the TF card. The application launches normally if the

measurement result passes the Decision Mechanism, which is unaware to the users. If the integrity check fails, TSB will occupy the screen to warn the user and the launch will be discarded. As for the application installation, a service TSB finds the application name and file through the broadcast of the application installation obtained by a receiver. The Control Mechanism controls whether the application will be installed according to the measurement result.

Since the measurement is not a one-time, but a long-term process, two measures need to be taken in practice. (1) To prevent the service from being recycled, we use the Android foreground service to keep monitoring the application. The foreground service is hard to be killed by the system when the system memory is insufficient. (2) To deal with the service of being killed in some extreme cases, we set up a system-level alarm clock that broadcasts once a minute. We register a broadcast receiver for the alarm to check whether the foreground service is still running. If the TSB finds no such service in the system, the TSB turns it on. These two measures ensure that the measurement of user applications will always exist after the device is booted.

4.4. Security Analysis. As mentioned in Section 1, one of the guarantees of TPM's security capabilities is that it is used as a dedicated hardware for computing. On commercial devices with limited resources and cost, we achieve similar capabilities of TPM with TSB and a secure TF card, which is physically isolated from Android malware. In the face of potential attackers who control Android malware, our solution can protect users from adversary attacks from two aspects. The trusted boot process ensures the trust of the user's device use environment, so that the device is in a safe state every time it boots. Whether the application is installed or launches, it will be measured and verified by TSB, which also ensures that users will not mistakenly use malware carefully set by the attacker.

A typical trusted boot checks the integrity of the component before it runs. It can only guarantee the integrity of these components and the safety of the boot process. Our TSB also checks the integrity of all components in the boot chain of commercial Android devices without changing its hardware and firmware. Defense against attacks before measurement, such as attacks on the kernel, relies on some mechanisms of the operating system itself, such as enabling mandatory access control SELinux [35] and kernel control flow integrity (CFI).

In fact, the implementation cannot fully comply with the TCG specification. In an implementation of the trusted PC [36], the BIOS will start running after the system is powered on and then load the PMBR (Pre-MBR) in the TPM into the main memory of the platform. The security of the BIOS is also guaranteed by its own access control. We use dedicated hardware to ensure the safety of boot process to a certain extent but cannot fully realize the ability of trusted computing. As mentioned in the threat model, attacks on Android devices come from malware. The device is in a trusted state when TSB and secure TF card are installed for the first time. Then, TSB begins to continuously measure the application of the device to prevent malicious software attacks.

However, we cannot completely cover potential TF card replacement or software attacks against TF cards. For the former, it depends on whether the secure TF card manufacturer will give the TF card a unique identity, using a technology such as PUF (Physical Unclonable Function) [37]. For the latter, the TF card is connected via a bus, which leads to possible bus attacks [38]. However, for daily users, these situations are outside our attack model.

5. Experimental Evaluation

We conducted experiments on commercial android devices and TF cards. Unfortunately, since our goal is to solve the problem of trust extension for commercial Android devices, we are not able to compare with other solutions such as fTPM under the same hardware conditions.

We tested the feasibility of our design as well as the performance overhead. In fact, after verifying the feasibility, the time cost of the solution is an aspect that needs to be compared. The time overhead for details such as key generation depends on the speed of the products produced by the TF card manufacturer. In addition, in our solution, the time-consuming computing work is handed over to the TSB process for execution after it is measured. That is, the device hardware of the TSB used, such as the SoC of the Android device, is generally faster than discrete TPM in theory.

5.1. Implementation Details. We choose Xiaomi Mi Pad 4 LTE and the JW1292T8I-SDK TF card produced by Chengdu JAVEE Microelectronics as the experimental hardware. Mi Pad 4 has Qualcomm Snapdragon 660 AIE SoC and 4 GB of RAM, and its system version is Android 9.0.

The TF card contains the public key encryption engine, hash engine, and random number generator corresponding to TCG's TPM specification. It also has the ability of symmetric encryption, which was added in the TPM2.0 specification. Different from the TCG specification, the cryptographic algorithms used in the TF card belong to the Chinese commercial cryptographic algorithms. The security TF card used in our experiment contains an SoC built with a 32-bit RISC processor. This SoC has hardware implementation of SM2 [39], SM4 [40], and random number generator (RNG). The SM3 [41] hash algorithm is completed by the RISC processor. These functions have dedicated calling interfaces for our TSB.

Experiments have proved that TSB can complete the trust extension during the boot process and the measurement of application launch and installation while putting a small burden on performance.

5.2. Feasibility. We verify the feasibility of trust extension to the boot chain and application.

5.2.1. Boot with TSB and TF Card. During boot process of the device, the integrity of the critical files from the boot-loader and kernel to the system and TSB itself must be

ensured. If the measurement result is inconsistent with the measured value in the TF card, the boot process will be terminated.

We replaced the kernel with a modified one and tested whether the device can boot. In addition, we tested the modification of key system files such as `init.rc` in the experiment. Most importantly, we also tested whether the device can boot normally when the TSB code is modified. As for the bootloader, we found that once it is modified, the device cannot boot or recover, so we mark the relevant column in boldface in Table 1. Table 1 shows the results, which verifies the effectiveness and feasibility of our TSB design to extend the trust in commercial device booting. Each case in the table is completely the same except for the modified part represented by ✓. The unmodified components are marked with •. The modified part should not meet the integrity measurement and cause the device to fail to boot up. It can be seen that any modification of any part of the boot chain can cause boot failure. The TSB can find the integrity damage in the boot chain and shut down the device.

5.2.2. Application Trust Measurement

(1) *Application Installation.* Malicious applications are the main form of attack against commercial Android devices, so trust in applications is vital to users. We developed two applications called applications 1 and 2 and only added the first one to the Baseline Repository. We compiled and installed the code of the first app again after changing it and found that the installation failed.

As shown in Table 2, experiments prove that applications that are not in the whitelist or that do not match the measured value in it cannot be installed, even if these applications are not malicious, and their names and icons are the same.

Because the malware is not in the whitelist, it cannot be installed either. However, we still tested a few malware, called Spitmo, Tapsnake, and DroidDream, which are all Trojan. The experimental results in Table 3 show that none of them can be installed and therefore no attack can be carried out.

(2) *Application Launch.* In fact, according to previous experimental results, it is not feasible to install malware on a device. In order to test applications that are not in the whitelist, we temporarily closed the Measurement Mechanism and Control Mechanism for application installation. We tested two applications, contacts and retromusic. The contacts app is in the whitelist and the retromusic is not.

Experimental results in Table 4 show that TSB can prevent applications that are not in the whitelist from being launched. We did not test the modified applications in the whitelist because the previous experimental results have shown that such applications cannot be installed, let alone launched.

5.3. *Performance Overhead.* We also consider whether the impact of TSB on the performance and energy consumption

TABLE 1: Cases of different modifications and their boot status.

	Bootloader	Kernel	System	TSB	Boot status
Case 1	X	•	•	•	Boot up
Case 2	X	✓	•	•	Shutdown
Case 3	X	•	✓	•	Shutdown
Case 4	X	•	•	✓	Shutdown

✓ stands for modified content. • stands for unchanged content. Since the bootloader of commercial device cannot be modified, it is marked with X. It can be seen that any cases that modify any content on the boot chain cannot be successfully started.

of the device is within the range that users can tolerate. When designing the TSB, we adopted some imperceptible measures to users. When the application is installed and opened, the user will not get any feedback if the measurement passes, thereby improving the user experience and device response. In addition, after the Primary Trust Base measured the TSB, computing tasks such as encryption are no longer run by the TF card but are handed over to the device to increase the running speed.

We conducted tests on device booting, application launch, installation time, power consumption, and performance and counted the impact of TSB on the above aspects.

5.3.1. *Time.* We measure the impact of TSB on time from three aspects: device boot, application launch, and installation.

(1) *Device Boot Time.* Since many users do not use TF cards, we measured the average boot time of the device in the three cases of no TF card or TSB, only a secure TF card, and both TF card and TSB. The boot time interval is defined as the device screen lighting up until the user can operate it. Therefore, we obtain the time interval from when the device is started to when TSB hands over the control of the device to the user after the measurement.

Table 5 shows that the secure TF card has considerable influence on the boot time. It takes about extra 60 seconds for the system to boot with TF card only. In the experiment, we used a customized secure TF card but not a universal one. The cryptographic capability in the TF card needs to be initialized when the device boots up. The extra time comes from the initialization of the TF card and the system trying to mount it. However, the real time spent in measurement is within 20 seconds. Considering that ordinary users do not reboot their devices frequently in daily life, the increase in boot time has little impact on the daily use of users.

(2) *Application Launch Time.* Application launch is the most frequently used operation on Android devices. In order to test the impact of TSB on the launch time of the application, we selected 16 applications for experimentation, A1 to A16 in Figure 5. The 16 applications are Via browser, Facebook lite, Instagram lite, Spotify lite, Firefox lite, Google Contacts, Google News, Shazam, Speedtest, WhatsApp, Instagram, Google Maps, QQ Music, TikTok, YouTube, and Photoshop Express in descending order of size ranging from 1 MB to 140 MB. Their sizes are marked as orange dots in Figures 5

TABLE 2: Installation status of the application in the whitelist before and after modification and the application not in the whitelist.

Application	Measured value	In the whitelist	Installation status
Application 1	eba4e2dbe4c75bd1b592c0625e09d550727fb5a1edeacd771e5351316c016229	Yes	Installed
Modified app 1	a82d263869037a9e80e359bbe3e2b2c3130c562e32ca9165a2adaf8fc96df60e	Yes	Not installed
Application 2	ae9123de2fc403666c9f48a6546fdcf5257ff69536f7c4d8a8d5d327d6a4e061	No	Not installed

TABLE 3: Installation status of malware.

Malware	Measured value	In the whitelist	Installation status
Spitmo	c6fa34e54c9b42ac33e39d59d0f063cf6e91dfd812ec6f56fc11c09a3f741a0	No	Not installed
Tapsnake	9df604b05082df6e1a891533f1f3cb32627730cf702be06ad5e3e5ed5ed3ea1c	No	Not installed
DroidDream	ec80b2d0256b0ac75a48a3698e0814a71b3cf5fe8c86d06776fe8f61b9937cdf	No	Not installed

TABLE 4: Launch status of applications in the whitelist and not in the whitelist.

Application	Measured value	In the whitelist	Launch status
Retromusic	aee539f03ca21386c3394c98b18a5eb322486662b9529b91422c5a70b9b57a64	Yes	Launched
Contacts	ad38108cd8c9526966762c4e2d7704b97c7a5f343a2eba2ac827647e028e3219	No	Not launched

TABLE 5: Average boot time in different cases.

Case	Time (s)
W/o TF card or TSB	42.37
W/TF card only	105.98
W/TF card & TSB	122.67

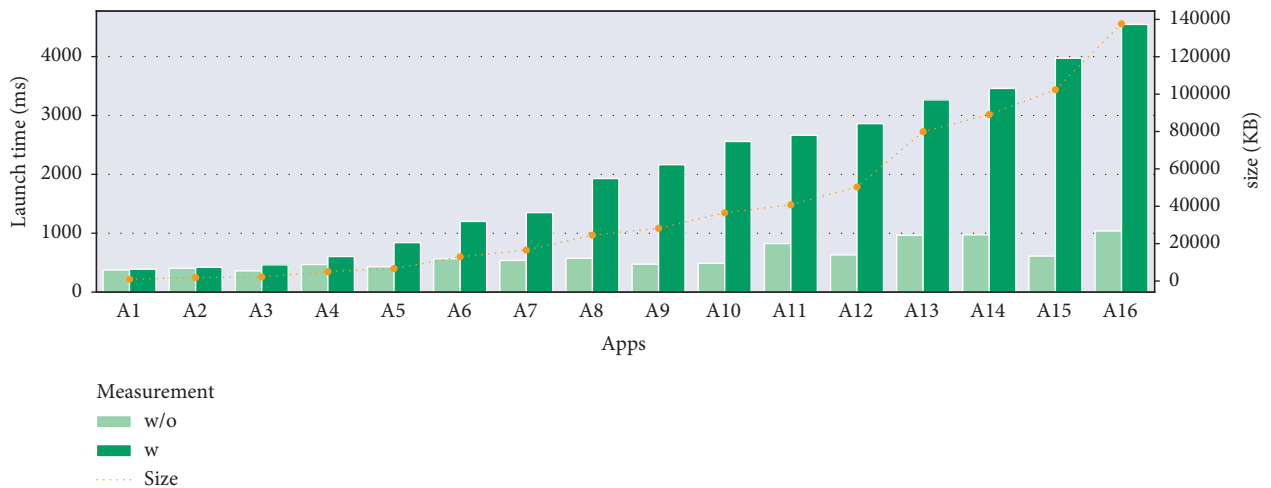


FIGURE 5: Application launch time.

and 6. We opened ten times and calculated the average time for each application with and without TSB and measured the time. TSB checks the integrity of the application while it is preparing to launch. Then, the application is ready to start until the check is complete. If the check fails, this launch will be discarded.

It is worth noting that to avoid interference from other factors as much as possible, we took some measures to ensure the consistency of the test environment, such as cleaning up the background before launching application and keeping the room temperature constant. In Figure 5, it can be seen that the application launch time is affected

in the presence of TSB. The average delays shown are between 0.3 seconds and about 4 seconds. Since the main time consumption comes from the hash operation, the time required for measurement also increases as the application package size increases.

(3) *Application Installation Time.* We tested the installation time of those 16 applications and added them to the whitelist in advance to ensure that the applications can be installed. When TSB detects that the application is installed, it will first jump to its own activity and wait for the measurement to be completed before TSB jumps back to the installer.

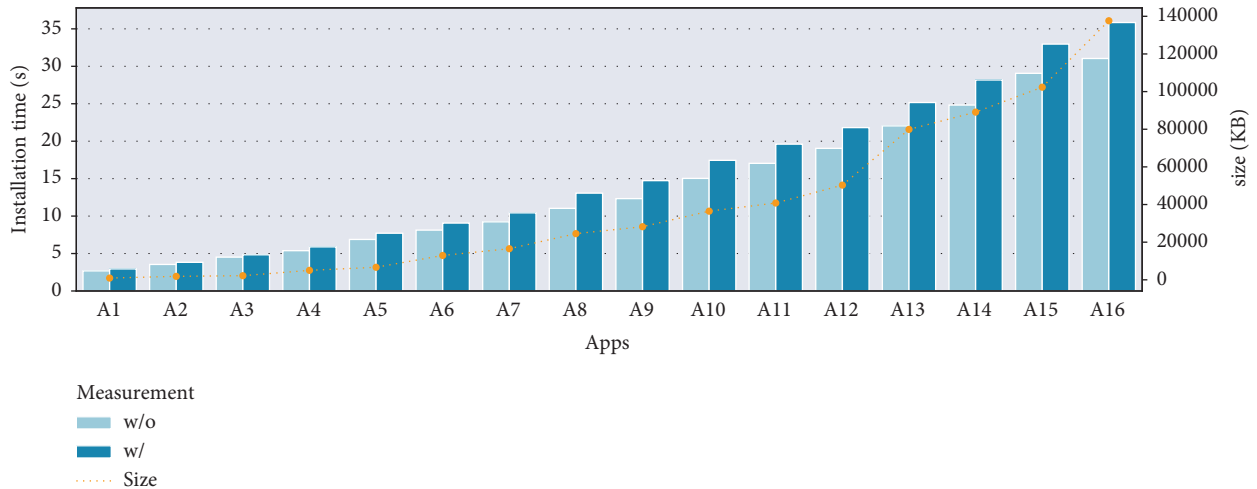


FIGURE 6: Application installation time.

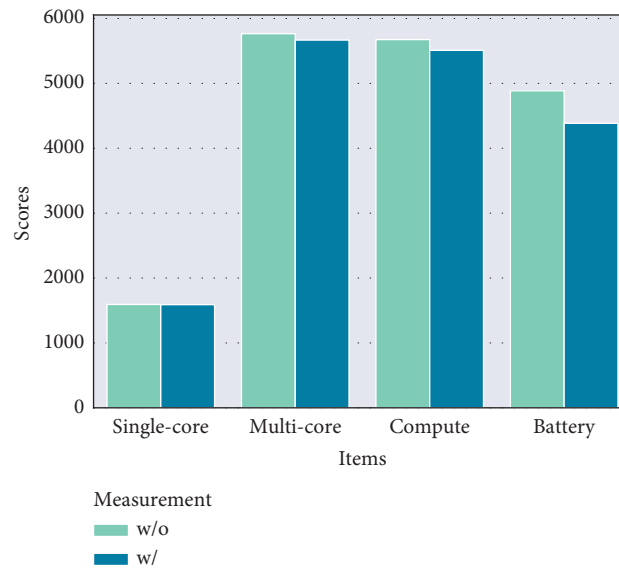


FIGURE 7: Performance and battery life.

Similarly, the increase in installation time is also caused by calculating the hash value of the applications. Figure 6 shows that the delay of application installation is at least less than 1 second and at most about 5 seconds. Moreover, for users, the frequency of application installation is low. From the experimental results, it can be considered that the measured installation has little impact on the user experience.

5.3.2. Performance and Battery Life. Since TSB spends most of its time on the measuring the application launch and installation, the trusted boot takes a small proportion of the time in the daily use of the device. Therefore, the impact of TSB on performance and energy consumption mainly comes from the always running foreground service. We use the mainstream benchmark application GeekBench 4.4.0 for testing. Figure 7 shows the results of

the experiment. Single-core scores and multicore scores reflect the processor capabilities. The compute score here represents the score of GeekBench's compute test item in Figure 7. The battery scores represent the endurance of the device in high resource requirement situation.

It can be seen from the results that after using the secure TF card and TSB, the processor and compute scores drop very little. However, the battery life score dropped by about 10%, indicating that the foreground service still has some impact on the device's battery life.

6. Conclusion

Ensuring the trust of Android devices can reduce attacks on IoT and edge facilities from the mobile domain. This allows security research to focus more on IoT and edge devices themselves. Existing methods for measuring the trust of Android devices all need to modify the underlying firmware

or hardware of the device, or both. But it is unrealistic for most Android devices on the market, especially smartphones, to be promoted by device vendors. In view of this situation, we propose a TSB design for commercial Android devices. The device does not need to modify any hardware and firmware but only add a specific TF card. Based on the star style trust measurement model, we designed TSB to ensure the trusted environment for device boot and application running. While our method ensures the safety of the device, the performance and energy consumption are not significantly affected. Considering that our TSB is still a prototype system, the performance overhead and delay caused by it will become lower with the software optimization.

In future work, improvements and in-depth research can be carried out in the following aspects:

- (1) The trust strategy in this paper is simple and cannot be updated. Adding a trust strategy management center without affecting system performance and managing trust strategy is worthy of study.
- (2) Trusted Network Connection (TNC) [42] is needed if TSB connects to the Internet to update the Baseline Repository or report the measurement result.
- (3) We have not considered the situation of multiple devices and TF cards. The access control of devices based on TF cards is worth being studied.
- (4) TSB needs to be designed with better robustness and compatibility. For application measurement, TSB needs to test whether it will not crash on the device with mass applications. In addition, TSB needs to be compatible with a variety of Android devices and Android OS versions from 5.0 to 11.0, accounting for 94.1% of the current Android version distribution. As an application, TSB requires code protection, such as obfuscation techniques [43].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (no. U1936122) and Primary Research & Development Plan of Hubei Province (nos. 2020BAB101 and 2020BAA003).

References

- [1] Y. Lee, S. Woo, J. Lee, Y. Song, H. Moon, and D. H. Lee, "Enhanced android app-repackaging attack on in-vehicle network," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 5650245, 13 pages, 2019.
- [2] M. Park, J. Han, H. Oh, and K. Lee, "Threat assessment for android environment with connectivity to iot devices from the perspective of situational awareness," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 5121054, 14 pages, 2019.
- [3] D. Wang, J. Ming, T. Chen, X. Zhang, and C. Wang, "Cracking iot device user account via brute-force attack to sms authentication code," *Proceedings of the First Workshop on Radical and Experiential Security*, pp. 57–60, 2018.
- [4] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DI-droid: deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, Article ID 101663, 2020.
- [5] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," *Security and Communication Networks*, vol. 2020, Article ID 8863617, 11 pages, 2020.
- [6] Y. Pan, X. Ge, C. Fang, and Y. Fan, "A systematic literature review of android malware detection using static analysis," *IEEE Access*, vol. 8, pp. 116363–116379, 2020.
- [7] S. Arshad, M. A. Shah, A. Khan, and M. Ahmed, "Android malware detection & protection: a survey," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, pp. 463–475, 2016.
- [8] D. Fu and Y. Liu, "Remote attestation on behavioral traces for crowd quality control based on trusted platform module," *Security and Communication Networks*, vol. 2021, Article ID 8859618, 12 pages, 2021.
- [9] C. Shen, H. Zhang, H. Wang et al., "Research on trusted computing and its development," *Science China Information Sciences*, vol. 53, no. 3, pp. 405–433, 2010.
- [10] F. Dengguo, Q. Yu, W. Dan, and C. Xiaobo, "Research on trusted computing technology," *Journal of Computer Research and Development*, vol. 48, no. 8, p. 1332, 2011.
- [11] Microsoft The Windows Team, "Update on windows 11 minimum system requirements," 2021, [https://blogs.windows.com/windows-insider/2021/06/28/update-on-windows-11-minimum-system-requirements/June 2021](https://blogs.windows.com/windows-insider/2021/06/28/update-on-windows-11-minimum-system-requirements/June%2021).
- [12] J.-E. Ekberg, K. Kostiainen, and N. Asokan, "The untapped potential of trusted execution environments on mobile devices," *IEEE Security & Privacy*, vol. 12, no. 4, pp. 29–37, 2014.
- [13] B. McGillion, T. Dettenborn, T. Nyman, and N. Asokan, "Open-TEE -- an open virtual trusted execution environment," *2015 IEEE Trustcom/BigDataSE/ISPA*, IEEE, vol. 1, pp. 400–407, 2015.
- [14] H. Raj, S. Saroiu, A. Wolman et al., "fTPM: a software-only implementation of a TPM chip," in *Proceedings of the 25th USENIX Security Symposium*, pp. 841–856, Austin, TX, USA, August 2016.
- [15] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard, "Armageddon: cache attacks on mobile devices," in *Proceedings of the 25th USENIX Security Symposium*, pp. 549–564, Austin, TX, USA, August 2016.
- [16] A. Machiry, E. Gustafson, C. Spensky et al., "Boomerang: exploiting the semantic gap in trusted execution environments," in *Proceedings of the 24th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2017.
- [17] Z. Ning and F. Zhang, "Understanding the security of arm debugging features," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy*, pp. 602–619, San Francisco, CA, USA, May 2019.
- [18] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: exposing the perils of security-oblivious energy management," in *Proceedings of the 26th USENIX Security*

- Symposium*, pp. 1057–1074, Vancouver, BC, Canada, August 2017.
- [19] C. Cimpanu, “Trust issues: exploiting TrustZone TEEs,” 2021, <https://www.bleepingcomputer.com/news/security/trustzone-downgrade-attack-opens-android-devices-to-old-vulnerabilities/>.
 - [20] W. Shi, “On design of a trusted software base with support of tpcm,” in *Proceedings of the International conference on trusted systems*, pp. 1–15, Beijing, China, December 2009.
 - [21] Trusted Computing Group, “TCG specification architecture overview,” 2007, <https://trustedcomputinggroup.org/resource/tcg-architecture-overview-version-1-4/2007>.
 - [22] C. Shen, H. Zhang, D. Feng, Z. Cao, and J. Huang, “Survey of information security,” *Science in China - Series F: Information Sciences*, vol. 50, no. 3, pp. 273–298, 2007.
 - [23] B. Zhao, H.-G. Zhang, J. Li, L. Chen, and S. Wen, “The system architecture and security structure of trusted PDA,” *Chinese Journal of Computers*, vol. 33, no. 1, pp. 82–92, 2010.
 - [24] Trusted Computing Group, “TCG mobile trusted module specification,” 2021, <https://trustedcomputinggroup.org/resource/mobile-phone-work-group-mobile-trusted-module-specification/>.
 - [25] Trusted Computing Group, *Tpm 2.0 Library Specification*, <https://trustedcomputinggroup.org/resource/tpm-library-specification/>, 2021.
 - [26] Trusted Computing Group, “Tpm 2.0 mobile reference architecture specification,” 2021, <https://trustedcomputinggroup.org/resource/tpm-2-0-mobile-reference-architecture-specification/>.
 - [27] D. Chakraborty, L. Hanzlik, and S. Bugiel, “simTPM: user-centric TPM for mobile devices,” in *Proceedings of the 28th USENIX Security Symposium*, pp. 533–550, Austin, TX, USA, August 2019.
 - [28] Trusted Computing Group, “TCG TSS 2.0 overview and common structures specification,” 2021, <https://trustedcomputinggroup.org/resource/tss-overview-common-structures-specification/>.
 - [29] A. Bianchi, J. Corbetta, L. Invernizzi, Y. Fratantonio, C. Kruegel, and G. Vigna, “What the app is that? deception and countermeasures in the android user interface,” in *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, pp. 931–948, IEEE, San Jose, CA, USA, May 2015.
 - [30] J. Reardon, Á. Feal, P. Wijesekera, A. E. B. On, N. Vallina-Rodriguez, and S. Egelman, “50 ways to leak your data: an exploration of apps’ circumvention of the android permissions system,” in *Proceedings of the 28th USENIX Security Symposium*, pp. 603–620, Austin, TX, USA, August 2019.
 - [31] C. Ren, Y. Zhang, H. Xue, T. Wei, and P. Liu, “Towards discovering and understanding task hijacking in android,” in *Proceedings of the 24th USENIX Security Symposium*, pp. 945–959, San Diego, CA, USA, February 2015.
 - [32] D. Lie, C. A. Thekkath, and M. Horowitz, “Implementing an untrusted operating system on trusted hardware,” *ACM SIGOPS - Operating Systems Review*, vol. 37, no. 5, pp. 178–192, 2003.
 - [33] S. Han, W. Shin, J. H. Park, and H. Kim, “A bad dream: subverting trusted platform module while you are sleeping,” in *Proceedings of the 27th USENIX Security Symposium*, pp. 1229–1246, Baltimore, MD, USA, August 2018.
 - [34] N. Lawson, “Tpm hardware attacks,” 2021, <https://rdist.root.org/2007/07/16/tpm-hardware-attacks/>.
 - [35] B. Radhika, N. N. Kumar, R. Shyamasundar, and P. Vyas, “Consistency analysis and flow secure enforcement of selinux policies,” *Computers & Security*, vol. 94, Article ID 101816, 2020.
 - [36] Y. Fajiang and Z. Huanguo, “Design and implementation of a bootstrap trust chain,” *Wuhan University Journal of Natural Sciences*, vol. 11, no. 6, pp. 1449–1452, 2006.
 - [37] A. Cui, C. H. Chang, W. Zhou, and Y. Zheng, “A new puf based lock and key solution for secure in-field testing of cryptographic chips,” *IEEE Transactions on emerging topics in computing*, vol. 9, no. 2, pp. 1095–1105, 2019.
 - [38] J. Boone, “TPM Genie,” 2021, <https://www.nccgroup.com/ae/our-research/tpm-genie/>.
 - [39] International Organization for Standardization, *IT Security Techniques — Digital Signatures with Appendix — Part 3: Discrete Logarithm Based Mechanisms*, International Organization for Standardization, Geneva, Switzerland, Standard ISO/IEC 14888-3:2018, 2018.
 - [40] International Organization for Standardization, *Information Technology — Security Techniques — Encryption Algorithms — Part 3: Block Ciphers*, International Organization for Standardization, Geneva, Switzerland, Standard ISO/IEC 18033-3:2010, 2010.
 - [41] International Organization for Standardization, *IT Security Techniques — Hash-Functions — Part 3: Dedicated Hash-Functions*, International Organization for Standardization, Geneva, Switzerland, Standard ISO/IEC 10118-3:2018, 2018.
 - [42] H.-G. Zhang, L. Chen, and L.-Q. Zhang, “Research on trusted network connection,” *Chinese Journal of Computers*, vol. 33, no. 4, pp. 706–717, 2010.
 - [43] S. Dong, M. Li, W. Diao et al., “Understanding android obfuscation techniques: a large-scale investigation in the wild,” in *Proceedings of the International Conference on Security and Privacy in Communication Systems*, pp. 172–192, Springer, Singapore, August 2018.

Research Article

A Hybrid Opportunistic IoT Secure Routing Strategy Based on Node Intimacy and Trust Value

Lin Yu ^{1,2}, Gang Xu ^{1,2}, ZhiFei Wang ^{1,2}, Na Zhang ^{1,2} and FengQi Wei ^{1,2}

¹College of Computer Science, Inner Mongolia University, Hohhot 010021, China

²Inner Mongolia A.R. Key Laboratory of Data Mining and Knowledge Engineering, Inner Mongolia University, Hohhot 010021, China

Correspondence should be addressed to Gang Xu; csxugang@imu.edu.cn

Received 11 July 2021; Revised 4 December 2021; Accepted 13 January 2022; Published 9 February 2022

Academic Editor: Qi Jiang

Copyright © 2022 Lin Yu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Routing strategy is one of the most important researches in Opportunistic Internet of Things (IoT), and it highly influences the efficiency of data transmission. In this paper, a hybrid Opportunistic IoT secure routing strategy based on node intimacy and trust value (HIRouter) is proposed to resolve the problem of unbalanced transmission efficiency and security in the message delivery process. According to the records of node encounter and message forwarding, the strategy proposed in this paper can calculate nodes' intimacy and trust value. The messages are then forwarded based on the intimacy and trust value between nodes. Experimental results verify that HIRouter algorithm we proposed can improve the message delivery rates and reduce the overhead rate in the Opportunistic IoT with dense nodes and frequent interactions between nodes.

1. Introduction

IoT is a network that uses sensing devices to connect objects and networks and enable objects to interact and exchange information, then achieve intelligent identification and management. It is mainly used in fields such as intelligent transportation, intelligent logistics, and smart cities. Opportunistic IoT is a kind of information interactive IoT characterized by temporary, opportunistic, and mobile self-organization. Opportunistic IoT enables intelligent objects to interact with each other by contact opportunity during the node moving [1–3]. Figure 1 shows an application scenario of the opportunistic IoT. An opportunistic network is derived from the delay-tolerant network [4–7]. There are no specific links between nodes in opportunistic networks, and messages are propagated in a “store-carry-forward” manner [8–10]. Research hotspots related to opportunistic networks include routing strategies, congestion control [11–14], energy consumption [15–18], and incentive mechanisms [19–23]. Among these, routing strategies are the key points of opportunistic network research. Currently, several routing algorithms have been proposed. The most

prominent routing algorithm is the epidemic algorithm. In the algorithm, nodes can forward their information to every node they encounter in the network before reaching the destination node [24]. The node of the spray and wait (S&W) algorithm produces L message copies for L neighboring nodes; then the L nodes that receive the copies seek the destination node [25–28]. The PROPHET algorithm forwards messages based on the historical information of node encounters [29].

Messages are forwarded in a “storage-carry-forward” manner in mobile Opportunistic IoT; in other words, messages are placed in intermediate node buffers [30]. Data transmission and distribution are completed through the encounter and contact opportunities created by node movements, and data eventually reach the destination node. The performance related to forwarding basis, such as message delivery success rate, transmission latency, and network resource consumption, determine the efficiency of message transmission in the data transmission process. Using the historical encounter information between nodes, the PROPHET routing algorithm formulates message-forwarding strategies; however, updating the message-

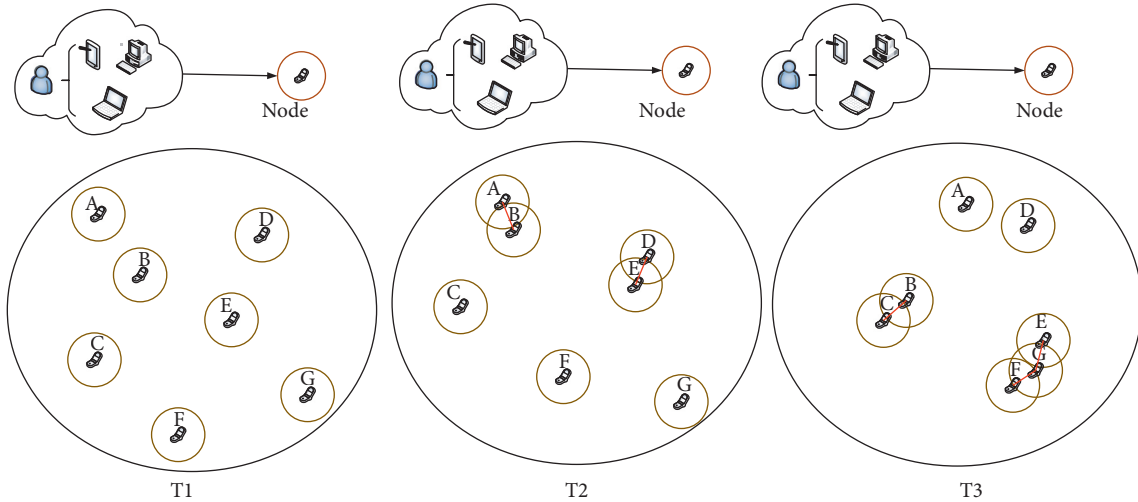


FIGURE 1: IoT application scenario.

forwarding efficiency in real-time with changes in the operating environment is difficult. A trusted routing scheme based on social similarity (TRSS) incorporates social trust into the routing decision process [31]. In the TRSS scheme, nodes move and associate with each other based on their common interests or social similarities and the next-hop forwarding node is assumed to be the node exhibiting common social characteristics with the destination node. This scheme mainly targets untrustworthy nodes, i.e., greedy or malicious nodes. The schema is not universal. In this paper, a hybrid Opportunistic IoT secure routing strategy based on node intimacy and trust value is proposed. The mixed utility value of node intimacy and trust value is determined by the historical data information of nodes and successful message-forwarding records. Moreover, messages are forwarded from nodes with low mixed utility values to nodes with high mixed utility values until the destination node is reached. The probability of messages being forwarded to selfish and malicious nodes is reduced, which laterally improves the security of messages in the network.

2. Related Works

Ma reviewed the research progress on mobile opportunistic network routing and described the main research content of mobile opportunistic networks in detail [9]. Li proposed a trust-based opportunistic routing (TOR) method by exploiting the trust mechanism, which improves the forwarding efficiency and information security of nodes in the opportunistic network [32]. The TOR method is suitable for networks with more malicious and greedy nodes. Although the bundling and carrying mechanism of the method improves security, it also increases the information storage consumption under this mechanism.

In order to solve the problem that the selfishness of the node due to the energy consumption of the message transmission leads to the serious degradation of the message transmission process and the increase of the network delay, N. Gupta proposed an incentive design mechanism based on

contract theory to reward intermediate nodes appropriately to forward the messages [30]. This mechanism appropriately encourages intermediate nodes to forward messages. According to their ability to forward the message, the mechanism divides nodes into a finite number of types and model the service transaction between the forwarding node and the forwarding node. The necessary and sufficient conditions are further derived to provide incentives for nodes participating in message forwarding so that the nodes are more active in the process of message transmission.

Based on the temporal and spatial constraints of node movement in a social-based opportunistic network, Yao proposed an energy-efficient message-forwarding algorithm in community-based opportunistic networks [33]. The algorithm divides the nodes in the network into communities and uses different transmission strategies for intra-community and intercommunity. The algorithm balances the efficiency of data forwarding on the opportunistic network with the energy consumption of the network. However, high message-forwarding efficiency cannot be achieved using this algorithm.

Based on the connection time, Duan proposed a probabilistic routing algorithm named PROPHET-CT, which resolved the problem of heavy network loads when the PROPHET routing algorithm calculated the node encounter probability [34]. The algorithm achieves a high message delivery success rate in the node-intensive scenario, but the delivery success rate is lower in a network with sparse nodes.

A trust-based data forwarding algorithm was proposed by Yuan [35]. The node trust value is determined by the three social attributes, and messages are forwarded in the network based on the trust value. However, this algorithm does not consider the influence of the message-forwarding path on the node trust value and the subsequent effect on the forwarding efficiency.

Since nodes in opportunistic networks have a large autonomy, there are many nodes that exhibit selfishness or even malicious behavior. For these nodes, Liu proposed an

alert system for bins based on opportunistic networks [36]. The system primarily evaluates whether the nodes are trustworthy for information transmission and forwards the message to the trusted nodes. This system is only applicable to networks in which the nodes have greater node autonomy and are not universal.

For the problem of inefficient message transmission due to unreliable link quality between nodes in the network, Yang proposed an opportunistic routing algorithm based on node connectivity [37]. The algorithm defines a set of forwarding candidates and the priority of each candidate node ensures efficient data transmission and reduces energy consumption during data transmission. The algorithm determines the reliability between nodes only based on node connectivity, which is always slightly incomplete, and the relationship between nodes should be fully considered.

Since the independence maintained by nodes in the process of motion will have a certain impact on data transmission, Zhang proposed an opportunistic network routing algorithm based on node motion for the motion characteristics of nodes [38]. The algorithm's data forwarding priority evaluation model and differential copy transfer policy guarantee message group delivery and lower delivery latency while limiting system overhead. The algorithm does not achieve a high delivery rate in networks with sparse nodes, so the algorithm is only suitable for networks with a certain node density.

A routing strategy for ad hoc networks based on node degree estimation and static game forwarding strategy is proposed for the broadcast storm problem caused by broadcast route grouping in the route discovery process of ad hoc networks [39]. The node degree estimation method and static game forwarding strategy of the strategy greatly reduce the network consumption caused by broadcasting Hello messages to obtain node degree information and also reduce a large amount of redundancy, competition, and conflicts generated during the route request packet broadcasting process, and improve the efficiency of route request packet broadcasting in the route discovery process. The strategy is currently only applicable to scenarios where the number of network nodes is large, and the nodes are evenly distributed in the network.

To address unreliable transmission in the network, Prashant Kumar proposed a reliability strategy [40]. In this strategy, the source node can recognize the message status. If an error in message transmission occurs, the source node can respond appropriately to resend the message. The strategy is instrumental because even if a message is lost, the message-forwarding success rate will not decrease.

T. Spyropoulos proposed an efficient routing scheme for intermittently connected mobile networks (S&W) [28]. S&W routing is divided into two phases, the node distributes the message to their neighbor nodes during the spraying phase, and the waiting phase carries the message nodes to the destination node. This scheme controls the number of copies of the message to be delivered in the network and solves the disadvantages of contagion routing. In the scheme, the control of the number of copies of messages is a very important issue.

In order to solve the problem of how to transmit messages when there is no linking path between the source node and the destination node, A Vahdat proposed epidemic routing for partially connected ad hoc networks (epidemic) [41]. The nodes in this route deliver the messages to all the encounter nodes, so ideally the delivery rate of this algorithm is the highest as long as the cache of the node is large enough. This is an extreme situation. In fact, the size of the node buffer is not enough to receive all the messages from other nodes, so the delivery rate of this route is not high due to the cache problem.

Lindgren proposed probabilistic routing in intermittently connected networks (Prophet) [42]. Prophet routing uses the historical encounter information between nodes to dynamically update the predicted value of the node's delivery. The message is delivered according to the predicted delivery value. If the predicted delivery value of the encountering node is higher than the node currently carrying the message, the message is copied and forwarded. The route is more suitable in areas with high node density and frequent encounters and is not suitable for remote areas where nodes are sparse.

To solve the problem of uneven energy consumption of nodes due to excessive calculation of key nodes and the loss of important messages due to limited remaining cache of nodes, Chen proposed an energy balance and cache optimization routing algorithm based on communication willingness (EC_CW) [43]. Based on the PROPHET algorithm, this algorithm forwards messages according to the multicopy mechanism and the willingness to communicate between nodes. In a sparse node network environment, it reduces the average delay and overhead rate, and in a node dense network environment, it improves the delivery rate. Although the delivery rate of the algorithm has been improved, its delivery rate is still at a relatively low level.

In summary, the existing opportunistic network routing algorithms do not consider the forwarding efficiency when data security is guaranteed. To this end, a hybrid opportunistic IoT secure routing strategy based on node intimacy and the trust value is proposed. The routing strategy mainly calculates the mixed utility value between nodes based on the encounters between the nodes and the message-forwarding path. The mixed utility value is used as the basis for message forwarding. This strategy considers the encounter situations between the nodes and successful message-forwarding records to ensure that messages are forwarded to the nodes with high intimacy and trust value thereby the strategy balances the efficiency and security of data forwarding.

3. Intimacy

In this work, a double hash table is used to simulate the relevant information between nodes in a matrix, which is stored under each node. Then, the nodes update the hash table during moving. Thus, by storing information about all nodes under every single node, the concept of decentralization can be implemented. The contents stored in the double hash table are shown in Figure 2.

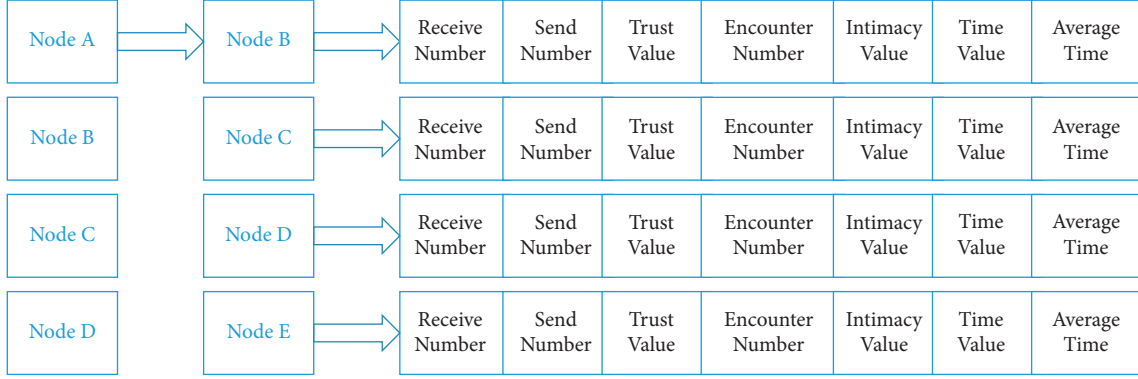


FIGURE 2: Information storage structure.

3.1. Encounter Information

Definition 1. EncounterNumber (EN) represents the number of connections between two nodes [44], i.e., the number of times two nodes have reached each other's communication range.

When two nodes are connected, each node updates its EncounterNumber information in the hash tables.

Definition 2. EncounterDuration (ED) represents the duration of node encounter [44], i.e., the length of time from when the communication link between two nodes is connected to when the communication link is disconnected.

Time Value indicates the connection time. The calculation of node encounter duration is shown as follows:

$$\text{EncounterDuration} = \text{CurrentTime} - \text{TimeValue}. \quad (1)$$

Definition 3. AverageEncounterDuration (TI) denotes the average encounter duration of each encounter between nodes [44], i.e., the ratio of the total encounter duration to the number of encounters between two nodes during network operation.

Generally, if the node encounter duration is short, network bandwidth is small, or transmission data are large, then complete data delivery may not be guaranteed. Therefore, forwarding messages to nodes with a longer average encounter duration can more effectively improve the message-forwarding success rate. To avoid forwarding messages to nodes with a short meeting duration, we need to calculate the average encounter duration $TI_{(avg,i,j)}$ of the node. When two nodes are disconnected, the encounter time and average encounter duration between the two nodes are calculated and stored. The average encounter duration is calculated as shown below:

$$TI_{(avg,i,j)} = \frac{\sum_{k=1}^n TI_k}{n}, k \leq n. \quad (2)$$

TI_k is the duration of the k th encounter between node i and j , and $\sum_{k=1}^n TI_k$ represents the sum of n encounter time.

3.2. Intimacy Information

Definition 4. Intimacy Value (I) signifies the frequency of contact between nodes during network operation.

More closely related nodes will meet frequently, while less closely related nodes will meet with a low probability. The node intimacy can be calculated using their meeting information, as follows:

$$I_{(i,j)} = EN_{(i,j)} * w + TI_{(avg,i,j)} * (1 - w). \quad (3)$$

Let $I_{(i,j)}$ denote the intimacy between nodes i and j , let $EN_{(i,j)}$ denote the number of encountering nodes, and let $TI_{(avg,i,j)}$ denote the average encounter time between nodes i and j . The variable w represents a weight value, $0 < w < 1$; the w is determined to be 0.57 based on the ratio of the encounter duration and the number of encounters in the experiment.

4. Trust Degree

Definition 5. Trust degree (T) between the destination and current nodes is defined as the ability of the current node to forward messages to the destination node. Based on successful message-forwarding records, the trust reward of the node forwarding data to the destination node is calculated.

Based on the forwarding path of the node successfully forwarding the message to the destination node, the trust degree of the destination node to other nodes on the successful path is determined.

4.1. Forwarding Path. In the simulation experiment, the messages are tracked, and the message-forwarding process is recorded. When the message successfully reaches the destination node, the successful message-forwarding path of this message is obtained.

Obtain the message-forwarding path:

$$\text{Path: } N_0 \longrightarrow N_1 \longrightarrow N_2 \longrightarrow N_3 \longrightarrow N_4 \longrightarrow \dots \longrightarrow N_j. \quad (4)$$

By obtaining the message-forwarding record, the nodes on the path on which the message is successfully forwarded are obtained, and these nodes are rewarded with trust.

To reflect the fairness and incentive of rewards for each node, a trust reward mechanism is established:

- (1) The position of the forwarding node in $\text{path}(N_i)$ determines the trust reward value of the forwarding node. If the forwarding node is closer to the destination node, the trust reward value of the forwarding node is higher, and vice versa. The trust reward value represents the trust degree between the destination and forwarding nodes and can be understood as the probability of the node forwarding a message to the destination node again in the future message-forwarding process.
- (2) For messages with the same communication link length, a shorter message delay time Δt (reception time - creation time) signifies a greater trust reward value of the forwarding node. The smaller the Δt , the less time it will require for the forwarding node to transmit the message to the destination node, i.e., smaller delay.

4.2. Trust Reward. The currently rewarded node is provided with a trust value for the first time, as follows [32]:

$$T_{ij}^{(m)} = \frac{1}{2} \left(e^{-\lambda \times (\Delta t / TTL)} + \varphi + (1 - \varphi) \times \left(\frac{\text{path}(N_i)}{|\text{path}(m)|} \right)^2 \right). \quad (5)$$

$T_{ij}^{(m)}$ represents the trust reward value of node j to node i , and $e^{-\lambda \times (\Delta t / TTL)}$ represents the time reward function. The smaller Δt is (i.e., the less time it takes for the message to be received from the creation to the destination node), the greater the reward obtained. The value of $(\Delta t / TTL)$ ranges from 0 to 1, and λ is the message delay reward adjustment factor. $\varphi + (1 - \varphi) \times (\text{path}(N_i) / |\text{path}(m)|)^2$ is the trust reward value function, which is determined according to the position of the node in the path, φ is the adjustment factor of trust reward value, $|\text{path}(m)|$ represents the length of the entire forwarding path, $\text{path}(N_i)$ represents the position of node N_i in the forwarding path, and the value of $\text{path}(N_i)$ is between 1 and $|\text{path}(m)|$, i.e., $(\text{path}(N_i) / |\text{path}(m)|)$ is between 0 and 1. In the trust mechanism, the closer the node position is to the destination node, the larger the $\text{path}(N_i)$ is and the greater the trust reward value is obtained. The message is forwarded from a low-trust-value node to a high-trust-value node.

The old trust value of the currently rewarded node is shown in the following:

$$T_{ij}^{(m)} = \text{old Trust Value} + (1 - \text{old Trust Value}) \times T_{ij}^{(m)}. \quad (6)$$

The trust value is updated based on the old trust value and the new reward trust value.

The node repeatedly forwards the message to the destination node and gradually improves the trust value reward; in this way, an oriented trust is established between the nodes in the network. Therefore, there is a trust-based

forwarding mechanism in the network. The mechanism can be expressed as a two-dimensional array. The index of the array is the node address value, and the content of the array is the size of the trust value.

When the message reaches the destination node, the Δt of the message is calculated by obtaining message reception time and creation time. Then, the time reward function is calculated using Δt and time-to-live (TTL) of the message. The message transmission path and the trust table of the destination node are obtained, and the path is traversed. If a node does not have a trust record in the trust table of the destination node, the trust calculation using the formula (5) is performed.

5. A Hybrid Opportunistic IoT Routing Algorithm Based on Node Intimacy and Trust Value

In the message-forwarding routing process, the node considers the encounter information with other nodes and the trust value of the two encountering nodes under the destination node. When determining whether the message should be forwarded to the meeting node, the routing mechanism will assign weights to the two forwarding bases to calculate the final forwarding probability.

Because the storage structure uses a hash table, the node address value corresponds to the key value of the hash table, and the relevant information about the node corresponds to the value of the hash table. The algorithm directly obtains the value of the hash table based on the key value of the hash table; hence, the time complexity of the algorithm is $O(1)$.

Forwarding probability is the mixed utility value.

Definition 6. The mixed utility value (denoted by FB) is obtained by the weighted summation of the intimacy and trust value of the nodes. It is used to describe the reliability relationship between nodes. This value is calculated using formula equation (6):

$$FB = T_{il}^{(m)} \times \alpha + \frac{EN_{(i,l)}}{EN_{(i,l)} + EN_{(j,l)}} \times \beta + \frac{TI_{(avg,i,l)}}{TI_{(avg,i,l)} + TI_{(avg,j,l)}} \times \gamma. \quad (7)$$

$EN_{(i,l)}$ is the number of encounters between the meeting node i and destination node l , $EN_{(j,l)}$ is the number of encounters between the current node j and destination node l , $TI_{(avg,i,l)}$ is the average encounter duration between the meeting node i and destination node l , and $TI_{(avg,j,l)}$ is the average encounter duration of the current node j and destination node l .

The forwarding probability calculation of the encounter node phase forwarding the message to the destination node consists of three parts: the trust degree of this encounter node under the destination node; the ratio of the number of encounters between the encountered node and the destination node and the number of encounters between the two nodes and the destination node respectively; the ratio of the encounter duration between the encountered node and the destination node and the sum of the encounter duration

```

(i) Input: Path, Trust Set (Nodedestination)
(ii) Output:  $T_{ij}^{(m)}$ 
(1) While message  $\rightarrow$  Nodedestination do
(2) Get Emulator Current Time
(3) Get Message Create Time
(4)  $\Delta t$  = Emulator Current Time – Message Create Time
(5) Get  $\Delta t$ ,  $TTL$ 
(6)  $CalH = e^{-\lambda \times (\Delta t / TTL)}$ 
(7) Get Path, Trust Set (Nodedestination)
(8) For  $i \in$  Path do
(9) If Node( $i$ )! = NULL then
(10)  $T_{ij}^{(m)} = \text{old Trust Value} + (1 - \text{old Trust Value}) \times T_{ij}^{(m)}$ 
(11) Else
(12)  $T_{ij}^{(m)} = 1/2(H + \varphi + (1 - \varphi) \times (\text{path}(N_i)/|\text{path}(m)|)^2)$ 
(13) End If
(14) End For
(15) End While

```

ALGORITHM 1: Trust reward algorithm.

```

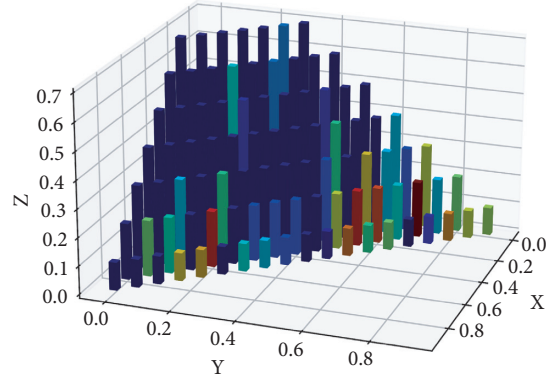
(i) Input: Trust Set (Nodedestination)
(ii) Output: message forwarding
(1) While (connection NodeA, NodeB)
(2) If NodeB == Nodedestination
(3) message  $\rightarrow$  NodeB
(4) Else
(5) Get Trust Set (Nodedestination)
(6) If Key(NodeA)  $\neq$  NULL then
(7) Get Value (NodeA)
(8) End If
(9) If Key(NodeB)  $\neq$  NULL then
(10) Get Value (NodeB)
(11) End If
(12)  $Cal\ FB_{A, destination}, FB_{B, destination}$ 
(13) If ( $FB_{B, destination} > FB_{A, destination}$ ) then
(14) message  $\rightarrow$  NodeB
(15) Else
(16) message  $\rightarrow$  NodeA
(17) End If
(18) End If
(19) End While

```

ALGORITHM 2: Message-forwarding algorithm.

between the two nodes respectively and the destination node. These three components are weighted and summed separately to obtain the forwarding probability of the encountered node. Similarly, the forwarding probability of the node currently carrying the message is also calculated in this way. α , β , and γ are the weights of the trust, the number of encounters, and the average duration of the encounters, respectively. The values of α , β , and γ are experimentally determined to be 0.1, 0.7, and 0.2, respectively. As shown in Figure 3, x , y , and z represent α , β , and γ , respectively. A more intense red color signifies a higher delivery success rate.

Step 1. indicates that the message-forwarding node meets the other nodes and establishes a connection. Steps 2–17 signify whether the currently established node performs message transmission. Steps 2–3 mean that if the message-forwarding node meets the destination node, the message is directly forwarded to the destination node. Steps 4–12 reflect that if the message-forwarding node does not meet the destination node, the node information is first obtained, and then the mixed utility value is calculated. Steps 13–16 indicate that if the mixed utility value of the message-forwarding node is less than that of the meeting node, the

FIGURE 3: Success rates under different α , β , and γ .

message is forwarded to the meeting node; otherwise, it is not forwarded.

6. Experimental Environment and Configuration

The algorithm we proposed was experimentally verified, and performance analysis is conducted on the simulation tool ONE [45]. The simulation tool version 1.4.1 is used in the work. Table 1 shows the design of simulation experiment parameters.

6.1. Performance. Several indicators are often used to determine the performance of the network.

Herein, the five indicators used for network performance evaluation and the importance of the five indicators are listed below.

The delivery rate is undoubtedly the most important in Opportunistic IoT. The higher the delivery rate, the more messages successfully forwarded to the destination node. The second is the average delay and overhead rate. The less time the message is delivered in the network and the smaller the cache overhead of the nodes in the network, the better the performance of the network. Finally, the number of relay nodes and the average number of relay nodes. The smaller the number of relay nodes means that the fewer times the message is forwarded by the node in the network, the less the total energy consumed by the node to forward the message, and the more the network performance can be improved.

- (1) Success rate is the ratio of the number of messages successfully forwarded to the destination node to the total number of messages. The indicator is denoted as equation (7).

$$S_{\text{success_rate}} = \frac{N_{\text{delivered}}}{N_{\text{created}}} \quad (8)$$

- (2) Average hops are the average number of hops required for a message to be successfully forwarded to the destination node. The indicator is denoted as equation (8).

$$N_{\text{average_hops}} = \frac{N_{\text{total_hops}}}{N_{\text{delivered}}} \quad (9)$$

- (3) Number of relay nodes is the number of intermediate nodes required for the successful forwarding of a message to the destination node. The indicator is denoted as equation (9).

$$N_{\text{relay_nodes}} = N_{\text{intermediate_delivery_nodes}} \quad (10)$$

- (4) *Average delay* is the average time required for a message to be forwarded from the forwarding node to the destination node. The indicator is denoted as equation (10).

$$N_{\text{average_delay}} = \frac{N_{\text{total_intermediate_delivery_nodes}}}{N_{\text{delivered}}} \quad (11)$$

- (5) Routing overhead is the ratio of the difference between the number of delayed messages and the number of successfully forwarded messages to the number of successfully forwarded messages. The indicator is denoted as equation (11).

$$R_{\text{overhead_rate}} = \frac{N_{\text{relayed}} - N_{\text{delivered}}}{N_{\text{delivered}}} \quad (12)$$

6.2. Experiment Analysis. In the experiment, by increasing the number of nodes in the network, the performance of HIRouter is compared with the performance of the epidemic [41], PROPHET [42], EC_CW [43], and S&W algorithms [28].

6.2.1. Success Rate. In Figure 4, when the number of nodes is small, the success rates of the four algorithms do not differ considerably. As the number of nodes increases, the success rate of HIRouter significantly increases. Moreover, as the number of nodes increases, the transmission success rate of the epidemic algorithm increases, and the prediction accuracy of the propagation probability of the PROPHET and EC_CW algorithms also improve. Therefore, the success rates of the epidemic and PROPHET algorithms are slightly

TABLE 1: The settings of simulation experiment parameters.

Parameter	Value
Experimental map	Helsinki map
Experimental area size	4.5 × 3.4 km (width × height)
Intra-area node movement model	Map-based shortest path movement model
Experimental simulation time	12 h (43200 s)
Number of nodes	36/66/96/126/156
Communication radius	10 m
Transfer speed	250 KB/s
Node moving speed	The first group: 0.5 ~ 1.5 m/s the second group: 2.7 ~ 13.9 m/s the third group: 0.5 ~ 1.5 m/s the fourth group: 7 ~ 10 m/s the fifth group: 7 ~ 10 m/s the sixth group: 7 ~ 10 m/s
Node cache size	Person: 5 M tram: 50 M
Transfer method	Person: Bluetooth interface tram: Bluetooth/high-speed interface
Message generation interval	25 ~ 35 s
Message size	500 kb-1 MB
Message life cycle	5 h

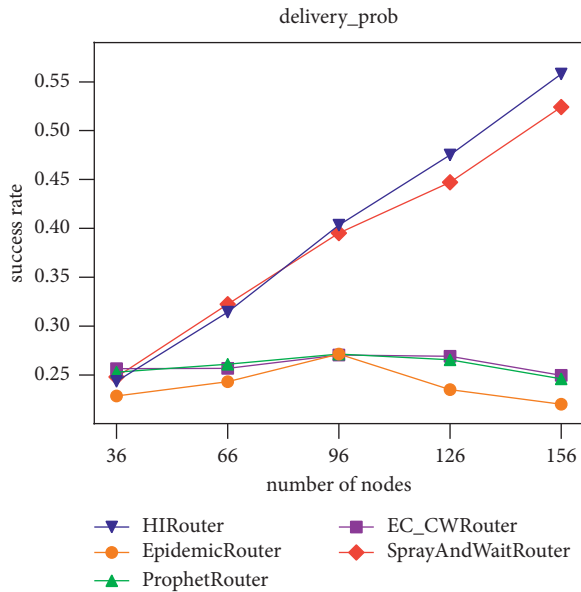


FIGURE 4: Success rate under the different number of nodes.

improved. The success rates of S&W and HIRouter algorithms are higher than those of the other two algorithms. Owing to an increase in the number of nodes in the S&W algorithm, the copy of spray's neighbor nodes significantly increases and the success rate also considerably increases. Compared with other routing algorithms, the HIRouter algorithm achieves the best success rate. As the number of mobile nodes increases, more accurate node relationships are estimated to provide a better forwarding strategy for opportunistic networks.

6.2.2. Average Hops. In Figure 5, as the number of nodes increases, the number of average hops of the epidemic algorithm increases, owing to the infection mechanism of the algorithm. Nodes will copy their message to forward it to the encountered nodes. The number of average hops of the PROPHET and EC_CW algorithms increases with the number of nodes; however, the increase is lower than that in

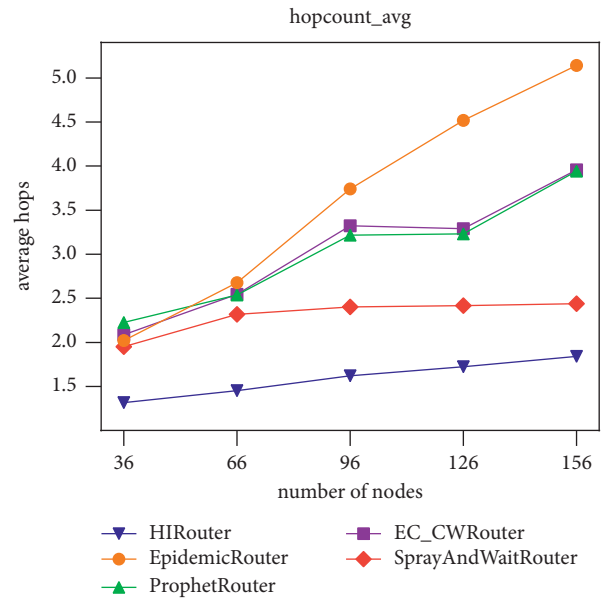


FIGURE 5: Average hops under the different number of nodes.

the epidemic algorithm. The number of average hops of S&W and HIRouter algorithms slightly increases when the nodes are increased; however, the number of average hops of the HIRouter algorithm is relatively few. This finding demonstrates that the HIRouter algorithm selects the optimal relay nodes during message transmission, reduces the number of relay nodes, and ensures that the message reaches the destination node in as few hops as possible.

6.2.3. Number of Relay Nodes. In Figure 6, due to the contagion mechanism of the epidemic algorithm, the number of relay nodes increases with the number of nodes when the epidemic algorithm is used to transmit information. The PROPHET and EC_CW algorithms show an increased number of relay nodes when the number of nodes increases; however, the number of relay nodes is less than that in the epidemic algorithm. When the number of nodes

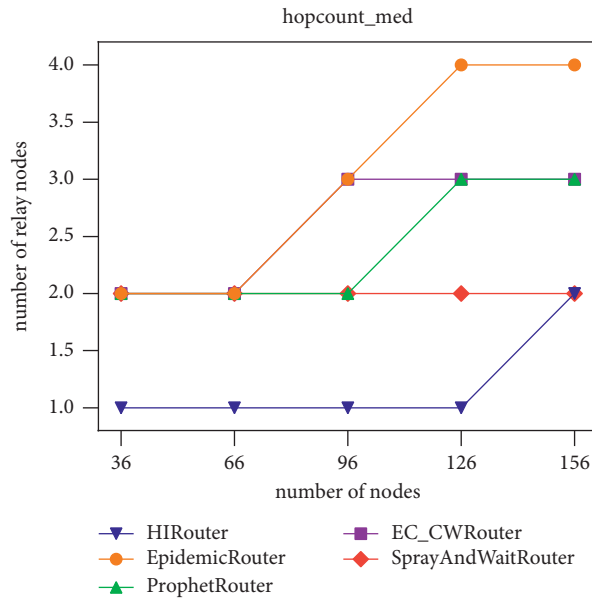


FIGURE 6: Number of relay nodes under different numbers of nodes.

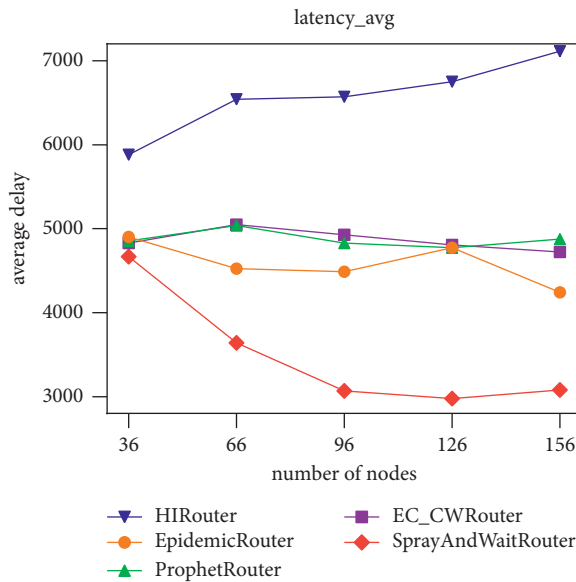


FIGURE 7: Average delay under the different number of nodes.

is small, the number of relay nodes of S&W and HIRouter algorithms is one. However, with an increasing number of nodes, the number of relay nodes of the HIRouter algorithm is minimal. This finding indicates that the HIRouter algorithm selects the optimal relay node during message transmission, thereby reducing the number of relay nodes; i.e., the number of message transmissions is reduced and the message loss rate is reduced.

6.2.4. Average Delay. In Figure 7, the average delay of the HIRouter algorithm is high. During message transmission, the nodes only forward messages to the nodes with a higher

probability of transmission. In the procedure, the nodes may encounter many nodes that are not qualified for forwarding and wait for the relay nodes with a higher possibility of transmitting the message to the destination node or the destination node itself. Therefore, the average delay of the HIRouter algorithm is high.

6.2.5. Overhead Rate. In Figure 8, as the number of nodes increases, the overhead rate of the epidemic and PROPHET algorithm significantly increases. This is attributed to the infection mechanism of the epidemic algorithm. Messages are copied between the nodes that meet, and the memory

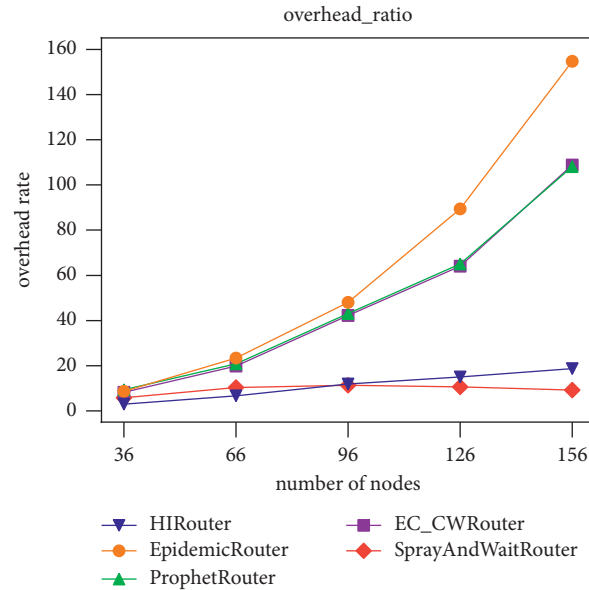


FIGURE 8: Overhead rate under the different number of nodes.

overhead is large. For the PROPHET and EC_CW algorithms, the recording of the forwarding probability and the calculation of the forwarding probability increase with an increasing number of nodes. Thus, the memory overhead increase. When the S&W algorithm is in the waiting phase, messages are forwarded only when nodes encounter the destination node, and there is no need for excess information recording and calculation. Therefore, as the number of nodes increases, the spray efficiency of the S&W algorithm shows no significant changes. As the number of nodes increases, although the overhead rate also increases, the HIRouter algorithm still maintains a low overhead. This is because the HIRouter algorithm selects better relay nodes when the message is forwarded, optimizing the delivery process and reducing overhead.

7. Conclusion

In this work, a hybrid Opportunistic IoT Secure routing strategy based on node intimacy and trust value is proposed. This strategy fully utilized the movement encounter information and successful message-forwarding information of the nodes in the network to determine the mixed utility value of nodes and identify the optimal relay nodes for message transmission.

From the experimental results, the HIRouter algorithm we proposed achieved a better success rate, a number of relay nodes, the average number of hops, and an overhead rate than the epidemic, PROPHET, EC_CW, and S&W algorithms. The routing algorithm improved the message delivery success rate and decreased the number of relay nodes and the average number of hops required for message transmission in the Opportunistic IoT with dense nodes and frequent interactions between nodes. At the same time, the routing algorithm reduces the information loss rate and the probability of malicious node damage from the side and

improves the security of the network. Moreover, the routing algorithm decreases network congestion and resource consumption.

In summary, the HIRouter algorithm we proposed can effectively accomplish data transmission in Opportunistic IoT with high efficiency and quality. Although the HIRouter proposed in this paper has achieved certain results in terms of success rate and overhead rate, the routing algorithm has a high average delay, which has an impact on the network performance of opportunistic IoT. The algorithm determines the forwarding utility value through the encounter between nodes and the successful forwarding record of the message. Therefore, the algorithm is suitable for the Opportunistic IoT with dense nodes and frequent interactions between nodes. For the Opportunistic IoT with sparse nodes and less interaction between nodes, this algorithm does not have a better delivery efficiency. In future research, certain replica strategies and incentive mechanisms will be used to promote message forwarding to reduce the average delay and improve network performance and propose an algorithm for the Opportunistic IoT with sparse nodes and less interaction between nodes.

Data Availability

Data are available upon request.

Conflicts of Interest

The authors declare no conflicts of interest related to this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants 62061036, 61841109, and 62077032; Natural Science Foundation of Inner Mongolia

under Grant 2019MS06031 and in part by the Self-Open Project of Engineering Research Center of Ecological Big Data, Ministry of Education.

References

- [1] A. Lohachab and A. Jangra, "Opportunistic Internet of Things (IoT): Demystifying the Effective Possibilities of Opportunistic Networks towards IoT," in *Proceedings of the 2019 6th International Conference On Signal Processing And Integrated Networks (SPIN)*, pp. 1100–1105, Noida, India, March 2019.
- [2] V. Petrov, A. Samuylov, V. Begishev et al., "Vehicle-based relay assistance for opportunistic crowdsensing over narrowband IoT (NB-IoT)," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3710–3723, 2018.
- [3] M. Gharbieh, H. ElSawy, M. Emara, H.-C. Yang, and M.-S. Alouini, "Grant-free opportunistic uplink transmission in wireless-powered IoT: a spatio-temporal model," *IEEE Transactions on Communications*, vol. 69, no. 2, pp. 991–1006, 2021.
- [4] Y.-P. Xiong, L.-M. Sun, J.-W. Niu, and Y. Liu, "Opportunistic networks," *Journal of Software*, vol. 20, no. 1, pp. 124–137, 2009.
- [5] J. Sweta and C. Meenu, "Survey of buffer management policies for delay tolerant networks," *Journal of Engineering*, vol. 7, no. 3, pp. 117–123, 2014.
- [6] C. Wang, D. Wang, G. Xu, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy for industry 4.0," *Science China Information Sciences*, vol. 65, no. 1, 2022.
- [7] P. Asuquo, H. Cruickshank, Z. Sun, and G. Chandrasekaran, "Analysis of dos attacks in delay tolerant networks for emergency evacuation," in *Proceedings of the 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, pp. 228–233, Cambridge, UK, September 2015.
- [8] H. C. Gao, X. J. Chen, D. Xu, Y. Peng, Z. Y. Tang, and D. Y. Fang, "Balance of energy and delay opportunistic routing protocol for passive sensing network," *Journal of Software*, vol. 30, no. 8, pp. 2528–2544, 2019.
- [9] H. D. Ma, P. Y. Yuan, and D. Zhao, "Research progress on routing problem in mobile opportunistic networks," *Journal of Software*, vol. 26, no. 3, pp. 600–616, 2015.
- [10] Y. Lu, W. Wang, L. Chen, Z. Zhang, and A. Huang, "Opportunistic forwarding in energy harvesting mobile delay tolerant networks," in *Proceedings of the 2014 IEEE International Conference on Communications (ICC)*, pp. 526–531, Sydney, NSW, Australia, June 2014.
- [11] J. Wu and Z. Chen, "Sensor communication area and node extend routing algorithm in opportunistic networks," *Peer-to-Peer Networking and Applications*, vol. 11, no. 1, pp. 90–100, 2018.
- [12] M. Misumi and N. Kamiyama, "Evacuation-route recommendation using DTN with evacuee attributes in disasters," in *Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7, Nanjing, China, March 2021.
- [13] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor Authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, 2020.
- [14] A. Marandi, M. Faghih Imani, and K. Salamatian, "Practical bloom filter based epidemic forwarding and congestion control in dtns: a comparative analysis," *Computer Communications*, vol. 48, pp. 98–110, 2014.
- [15] F. De Rango and S. Amelio, "Geographic and energy aware epidemic strategy for mobile opportunistic DTN," in *Proceedings of the 2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–8, Honolulu, HI, USA, August 2020.
- [16] A. Khalil, N. Mbarek, and O. Togni, "Fuzzy Logic based model for self-optimizing energy consumption in IoT environment," in *Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7, Nanjing, China, March 2021.
- [17] H.-T. Ye, X. Kang, J. Joung, and Y.-C. Liang, "Optimization for wireless-powered IoT networks enabled by an energy-limited UAV under practical energy consumption model," *IEEE Wireless Communications Letters*, vol. 10, no. 3, pp. 567–571, 2021.
- [18] X. Zhou, X. Yang, J. Ma, and K. I.-K. Wang, "Energy efficient smart routing based on link correlation mining for wireless edge computing in IoT," *IEEE Internet of Things Journal*, vol. 1, 2021.
- [19] G. Goudar and S. Batabyal, "Optimizing bulk transfer size and scheduling for efficient buffer management in mobile opportunistic networks," *IEEE Transactions on Mobile Computing*, vol. 1, 2021.
- [20] J. Xu, J. Xiang, and D. Yang, "Incentive mechanisms for time window dependent tasks in mobile crowdsensing," *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 6353–6364, 2015.
- [21] S. K. Dhurandher, J. Singh, M. S. Obaidat, I. Woungang, S. Srivastava, and J. J. P. C. Rodrigues, "Reinforcement learning-based routing protocol for opportunistic networks," in *Proceedings of the ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, Dublin, Ireland, June 2020.
- [22] D. Wang, X. Zhang, Z. Zhang, and P. Wang, "Understanding security failures of multi-factor Authentication schemes for multi-server environments," *Computers & Security*, vol. 88, no. 2020, pp. 1–13, 2020.
- [23] Q. Xu, Z. Su, and S. Guo, "A game theoretical incentive scheme for relay selection services in mobile social networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6692–6702, 2015.
- [24] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan, "Prioritized epidemic routing for opportunistic networks," in *Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking*, pp. 62–66, New York, NY, USA, June 2007.
- [25] A. Al-Hinai, H. Zhang, Y. Chen, and Y. Li, "Tb-snw: trust-based spray-and-wait routing for delay-tolerant networks," *The Journal of Supercomputing*, vol. 69, no. 2, pp. 593–609, 2014.
- [26] M. Y. Mir and C.-L. Hu, "Exploiting mobile contact patterns for message forwarding in mobile opportunistic networks," in *Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7, Seoul, Korea (South), May 2020.
- [27] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang, "Understanding node capture attacks in user authentication schemes for wireless sensor networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, p. 1, 2020.
- [28] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM*

- SIGCOMM Workshop on Delay-Tolerant Networking*, pp. 252–259, New York, NY, USA, August 2005.
- [29] S. Shabalala, Z. Shibeshi, and K. Khalid, “Design of energy-aware prophet and spray-and-wait routing protocols for opportunistic networks,” in *Proceedings of the International Conference on Wireless Intelligent and Distributed Environment for Communication*, pp. 35–46, Manhattan, NY, USA, 2018.
- [30] N. Gupta, J. Singh, S. K. Dhurandher, and Z. Han, “Contract theory based incentive design mechanism for opportunistic IoT networks,” *IEEE Internet of Things Journal*, p. 1, 2021.
- [31] L. Yao, Y. Man, Z. Huang, J. Deng, and X. Wang, “Secure routing based on social similarity in opportunistic networks,” *IEEE Transactions on Wireless Communications*, vol. 15, no. 1, pp. 594–605, 2015.
- [32] F. Li and Y. Si, “Trust-based security routing decision method for opportunistic networks,” *Journal of Software*, vol. 29, no. 9, pp. 2829–2843, 2018.
- [33] M. Yao and S. Zhang, “Energy balanced routing algorithm based on community in opportunistic networks,” *Journal of Chinese Computer Systems*, vol. 39, no. 9, p. 5, 2018.
- [34] Z. Duan and Y. Yang, “Opportunistic forwarding algorithm based on connection time in probabilistic routing,” *Microelectronics & Computer*, vol. 35, no. 12, pp. 50–54, 2018.
- [35] J. Yuan, Z. Zhang, and W. Yang, “Data forwarding scheme based on social trust in opportunistic networks,” *Computer Engineering and Design*, vol. 36, no. 8, pp. 2011–2015, 2015.
- [36] M. Liu, Z. Chen, and J. Wu, “Design and implementation of routing security in mobile opportunistic networks,” *Electronic Technology & Software Engineering*, vol. 134, no. 12, Article ID 244, 2018.
- [37] X. Yang, T. Liang, and X. He, “Opportunistic routing based on node connectivity for wireless sensor networks,” *Journal of Chinese Computer Systems*, vol. 40, no. 11, p. 24, 2019.
- [38] Q. F. Zhang, C. Gui, Y. Song, B. L. Sun, and Z. F. Dai, “Routing algorithm in opportunistic networks based on node mobility,” *Journal of Software*, vol. 32, no. 8, pp. 2597–2612, 2021, in Chinese.
- [39] Q. W. Wang, Q. Qi, W. Cheng, and D. Li, “Node degree estimation and static game forwarding strategy based routing protocol for ad hoc networks,” *Journal of Software*, vol. 31, no. 6, pp. 1802–1816, 2020, in Chinese.
- [40] P. Kumar, N. Chauhan, and N. Chand, “Node activity based routing in opportunistic networks,” in *Proceedings of the International Conference on Futuristic Trends in Network and Communication Technologies*, pp. 265–277, Manhattan, NY, USA, 2018.
- [41] A. Vahdat and D. Becker, “Epidemic routing for partially-connected ad hoc networks,” *Handbook of Systemic Auto-immune Diseases*, 2000.
- [42] A. Lindgren, A. Doria, and O. Schelén, “Probabilistic routing in intermittently connected networks,” in *Proceedings of the International Workshop on Service Assurance with Partial and Intermittent Resources*, Berlin, Heidelberg, 2004.
- [43] J. Chen, G. Xu, X. Wu, F. Wei, and L. He, “Energy balance and cache optimization routing algorithm based on communication willingness,” in *Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, Nanjing, China, March 2021.
- [44] Y. Yun-hui, X. M. Wang, L.-C. Zhang, S. Liu, and Y.-G. Lin, “An encounter-based routing algorithm for social opportunistic networks,” *Computer Technology and Development*, vol. 028, no. 002, pp. 64–68, 2018.
- [45] Z. Wang, X. Wang, and J. Sui, “Extending research for one simulator of opportunistic network,” *Application Research of Computers*, vol. 29, no. 1, 2012.

Research Article

Lattice-Based Self-Enhancement Authorized Accessible Privacy Authentication for Cyber-Physical Systems

Jinhui Liu ^{1,2}, Yong Yu ³, Houzhen Wang,⁴ and Huanguo Zhang⁴

¹School of Cyber Security, Northwestern Polytechnical University, Xi'an 710072, China

²Research & Development Institute of Northwestern Polytechnical University, Shenzhen 518057, China

³School of Cyber Security, Xi'an University of Posts and Telecommunications, Xi'an 710072, China

⁴School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

Correspondence should be addressed to Yong Yu; yuyongxy@163.com

Received 24 September 2021; Accepted 14 December 2021; Published 9 February 2022

Academic Editor: Ding Wang

Copyright © 2022 Jinhui Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Healthcare cyber-physical system significantly facilitates healthcare services and patient treatment effectiveness by analyzing patients' health information data conveniently. Nevertheless, it also develops the threats to the confidentiality of health information, patients' privacy, and decidability of medical disputes. And, with the advances of quantum computing technology, most existing anonymous authentication schemes are becoming a growing threat to traditional cryptosystems. To address these problems, for healthcare cyber-physical systems, we propose a new lattice-based self-enhancement authorized accessible privacy authentication scheme by using a strong designated verifier double-authentication-preventing signature technique, called SEAPA. The SEAPA achieves three security and privacy requirements including unforgeability, anonymity for patients' information, and self-enhancement for patients themselves. A detailed security proof shows our proposal achieves those required security goals. Finally, our construction is demonstrated by parameter analysis and performance evaluation to have reasonable efficiency.

1. Introduction

Cyber-physical system is an integration of computational resources, physical processes, and communication capabilities, which is a multidimensional complex system combined by sensors, embedded devices, and wireless links. The advances in medical sensors, cloud computing, Internet of things, and wireless sensor networks (WSN) have witnessed CPS a powerful candidate for healthcare applications [1–6]. For example, COVID-19 has been causing incalculable damage to human health, economy, and life, and it is worth noting that wireless medical sensor network plays an important role in China's activity in resisting COVID-19. To provide a more convenient service and healthcare environment, a healthcare cyber-physical system (HCPS) is proposed [1]. Taking a patient at anywhere as an example, his privacy information can be collected by various sensors and then it can be sent to a

third party cloud server. At the same time, doctors in a hospital can monitor the patient's physical condition and give some prescriptions or suggestions. Although each aspect of HCPS has made a great progress, security and privacy of patients' personal health information in HCPS have always been in the spotlight.

In a general HCPS system model, there occur components, including the medical sensor node of the patient, data sink which can collect patient's privacy information, and healthcare centers which have different hospitals, databases, and doctors. The system model of the healthcare cyber-physical system is shown in Figure 1. It has advantages of remote consultation system and mobile sensor system. On the one hand, the privacy information of patients can be collected in data sink and uploaded to database in corresponding hospitals. On the other hand, multiple remote doctors can provide some timely and accurate medical services by analyzing patients' physiological data.

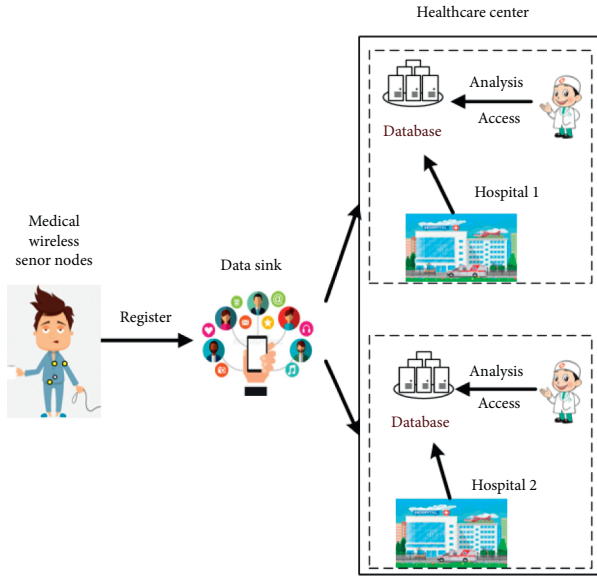


FIGURE 1: A system model of HCPS.

Although the HCPS system model provides convenience for the patients' treatment, it still faces some defects and potential treatments. For instance, in our daily life, there exist some misbehavior patients who want to see a doctor and do not want doctors to know her (or his) disease characteristics completely. The patient tells some ambiguous assertion to two kinds of authorized physicians, so she has to sign a pair of colliding messages with the same personal information and some different assertions. If there exist some medical disputes between them, the patient's misbehavior can be found. At present, all related digital signatures cannot provide a method to solve this problem and thus, we have to use another technique called double-authentication-preventing signature.

In addition, "Sycamore" quantum computer from the United States and the "Nine Chapter" quantum computer designed by China have been an important milestone. Since classical cryptographic protocols in HCPS will be broken by malicious adversary using quantum computers, private information of patients used to diagnose disease will be let out, which will result in loss of property and life. Lattice-based signatures have two kinds of advantages. One is that the hardness of some average-case lattice hard problems is equivalent to that of NP hard problem; the other is that lattice-based signatures have high efficiency, because it is based on operations between matrix additions and multiplications.

To realize these issues above, a novel privacy-preserving model for HCPS is established to allow patients to authorize privileges to different kinds of physicians located in the healthcare centers. Based on the model above, a self-enhancement postquantum secure privacy-preserving authentication scheme (SEAPA for short) in HCPS is proposed and it satisfies security requirements for patients. If the patient misbehaves, he (or she) will be punished by extracting his (or her) private keys. A rigorous security proof is shown that our proposed scheme is secure under the

assumption of computational shortest integer solution problem in the random oracle model. Security proof and performance evaluation show that our scheme has reasonable efficiency for real applications.

2. Related Work

Digital signatures that provide message integrity, message authenticity, and nonrepudiation are publicly verifiable. However, in the HCPS model, the signed messages may infer signer's health information which reflects emotions and life of patients. In our daily life, if these signatures of messages are publicly verified, it will reveal the patients' personal health information and make some troubles for the patient. To solve these issues, Jakobsson et al. proposed an idea of designated verifier signature (DVS) that it convinces one, and only one (the designated verifier), to prove the validity of a signature [7]. Since the proposed DVS cannot resist an adversary to get the signature before it is obtained by the designated verifier, Jakobsson et al. also proposed a strong designated verifier signature (SDVS) [7]. As building blocks, DVSs are widely used in privacy-preserving security protocols such as cloud computing [8, 9], big data [10], Internet of things [11, 12], electronic voting systems [13, 14], and healthcare information systems [15, 16].

A series of designated verifier signatures with particular functions were proposed [17–25]. For example, to solve the key management problem, Huang et al. proposed an identity-based SDVS [26]. To solve the key-escrow problem of identity-based SDVS, Chen et al. proposed a certificateless SDVS with nondelegatability [22]. In recent years, He et al. proposed a certificateless designated verifier proxy signature scheme for unmanned aerial vehicle networks [27]. Zheng et al. presented a practical quantum designated verifier signature scheme for E-voting applications [28]. However, those properties may not be desirable. Consider such a scenario that a patient wants to see a doctor, while in our daily life, there exist many misbehaved patients who do not want others to know her (or his) disease characteristics completely. The patient tells some ambiguous assertion to two kinds of authorized physicians, so he/she has to sign a pair of colliding messages with the same personal information and some different assertions. If there exist some medical disputes between them, the patient's misbehavior can be found.

There are three categories in our proposed HCPS including directly authorized physicians, indirectly authorized physicians, and unauthorized physicians. By a new designated verifier signature called a designated verifier double-authentication-preventing signature (DVDAPS), which is derived from a double-authentication-preventing signature (DAPS) and a designated verifier signature (DVS), it realizes three different privacy-preserving requirements. DVDAPS can be deterrable (or punishable) by extracting the patients' secret keys of a signature on colliding messages if there exists a dispute. If patients' secret keys are extracted, their personal health information will be revealed to anyone. The DVDAPS can be considered as an attack algorithm or as a self-enhancement digital signature scheme.

2.1. Motivations. In practice, a signer may maliciously sign the messages twice to spread inappropriate contents or even sell patents more than once to gain illegal profits. Such actions must be punished as the actions impact the security, reputation, and robustness of the entire system. However, to the best of our knowledge, current digital signatures cannot provide the property of punishability. To address this issue, a double-authentication-preventing signature (DAPS for short) can be used to realize the requirement with deterrability. However, one cannot employ the existing general DAPS to solve the confidentiality and identity privacy of patients' personal health information in HCPS.

2.2. Contributions. In this paper, we give an affirmative answer to the above problem by introducing the first formal treatment for deterrability. We present a new deterrable digital signature, which is proven secure under a standard assumption on lattice, and then based on it, we realize a practical construction of DVDAPS in SEAPA. The major contributions of the paper are three-fold. Firstly, we give a formal definition of DVDAPS and propose the notions of unforgeability, anonymity, and self-enhancement in presence of attacks. Secondly, DVDAPS serves as the fundamental building blocks to offer the properties of privacy and deterrability simultaneously. To instantiate an efficient construction, we propose a secure construction under the assumption of lattice hard problems. Finally, we provide a concrete construction of SEAPA with performance evaluation.

2.3. Organization. In this paper, we propose a lattice-based self-enhancement authorized accessible privacy authentication scheme for HCPS. First, we introduce some notions, cryptographic primitives, and a security model of HCPS used in this paper. Second, we establish an authorized accessible privacy model for HCPS. Third, we present a concrete lattice-based privacy-preserving authentication scheme with properties of completeness, unforgeability, nontransferability, and extractability. Finally, we analyze our proposal from aspects of security and parameter settings.

3. Preliminaries

3.1. Notations. For a set \mathcal{S} , $a \leftarrow \mathcal{S}$ indicates that a is selected randomly from the set \mathcal{S} .

Ring is $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$.

Column vector $\mathbf{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_k \end{pmatrix}$ could be represented as $[a_1; \dots; a_k]$.

$\|\mathbf{a}\|_1 = \sum_{i=0}^{n-1} |a_i|$, $\|\mathbf{a}\| = \sqrt{\sum_{i=0}^{n-1} |a_i|^2}$, and $\|\mathbf{a}\|_\infty = \max_i |a_i|$.

For a full-rank integer lattice Λ , the discrete distribution is

$$D_{\Lambda, \mathbf{c}, \sigma} = \frac{e^{-\|\mathbf{v} - \mathbf{c}\|^2 / 2\sigma^2}}{\sum_{\mathbf{w} \in \Lambda} e^{-\|\mathbf{w} - \mathbf{c}\|^2 / 2\sigma^2}}. \quad (1)$$

Definition 1 (R-SIS $_{q, \beta, m}$). Given m uniform elements $a_i \in R_q$ at random and let $\mathbf{a} = (a_1, \dots, a_m) \in R_q^m$, find out a nonzero vector $\mathbf{z} \in R^m$ with norm $\|\mathbf{z}\| \leq \beta$ such that

$$\mathbf{a} \cdot \mathbf{z} = 0 \in R_q. \quad (2)$$

Note that in Ring-SIS, each $a_i \in R_q$ corresponds to n -related vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$ in SIS, where n is the degree of R over \mathbb{Z} . Each $z_i \in R$ of a Ring-SIS solution corresponds to a block of n integers. That is to say, $\mathbf{a} \in \mathbb{Z}_q^{m \times m}$ and $\mathbf{z} \in \mathbb{Z}^{m \times m}$.

3.2. Ring-SIS Signature Scheme. Lyubashevsky's signature scheme is given as follows [29].

Secret key: $\mathbf{S} \leftarrow \{-d, \dots, d\}^{m \times m}$

Public key: $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times m}$, $\mathbf{T} \leftarrow \mathbf{A}\mathbf{S}$ and $H_1: \{0, 1\}^* \rightarrow \{\mathbf{v}: \mathbf{v} \in \{-1, 0, 1\}^m, \|\mathbf{v}\|_1 \leq \kappa\}$

Sign: given a message μ , compute the following:

(i) $\mathbf{y} \leftarrow D_\sigma^m$

(ii) $\mathbf{c} \leftarrow H_1(\mathbf{A}\mathbf{y}, \mu)$

(iii) $\mathbf{z} \leftarrow \mathbf{S}\mathbf{c} + \mathbf{y}$

(iv) Output (\mathbf{z}, \mathbf{c}) with probability $\min(D_\sigma^m(\mathbf{z}) / \text{MD}_{\mathbf{S}\mathbf{c}, \sigma}^m(\mathbf{z}), 1)$, where $M = O(1)$

Verify

(i) Accept iff $\|\mathbf{z}\| \leq 2\sigma\sqrt{m}$ and $\mathbf{c} = H_1(\mathbf{A}\mathbf{z} - \mathbf{T}\mathbf{c}, \mu)$

We transform the signature scheme above to Ring-SIS signature scheme as follows:

Secret key: $\mathbf{s} \leftarrow R_{-d, d}^m$, where let $R_{-d, d}^m$ be $\{-d, \dots, 0, \dots, d\}^{m \times m}$

Public key: $\mathbf{a} \leftarrow R_q^m$, $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s}$ and $H: \{0, 1\}^* \rightarrow \{\mathbf{v}: \mathbf{v} \in R_{-1, 1}, \|\mathbf{v}\|_1 \leq \kappa\}$

Sign: given a message μ , compute the following

(i) $\mathbf{y} \leftarrow D_\sigma^m$

(ii) $\mathbf{c} \leftarrow H(\mathbf{a}\mathbf{y}, \mu)$

(iii) $\mathbf{z} \leftarrow \mathbf{s}\mathbf{c} + \mathbf{y}$

(iv) Output (\mathbf{z}, \mathbf{c}) with probability $\min(D_\sigma^m(\mathbf{z}) / \text{MD}_{\mathbf{s}\mathbf{c}, \sigma}^m(\mathbf{z}), 1)$, where $M = O(1)$

Verify

(i) Accept iff $\|\mathbf{z}\| \leq 2\sigma\sqrt{m}$ and $\mathbf{c} = H(\mathbf{a}\mathbf{z} - \mathbf{t}\mathbf{c}, \mu)$

3.3. System Model Description. Our system model is illustrated in Figure 2, which mainly includes three parts as follows. Healthcare providers are equipped with cloud servers, wireless transmission networks, and body area networks. The health information of a patient is transmitted to two different healthcare providers to different kinds of authorized physicians for accessing and making some medical treatments, respectively. There are two healthcare centers with healthcare providers A and B and the medical research institutions C, where Dr. Alice, Dr. Bob, and Dr. Eve are working in Hospital 1. Each of them have their cloud server. If a patient registers at Hospital 1, his (or her) health information will stored in the cloud server of the Hospital 1, while his health information will

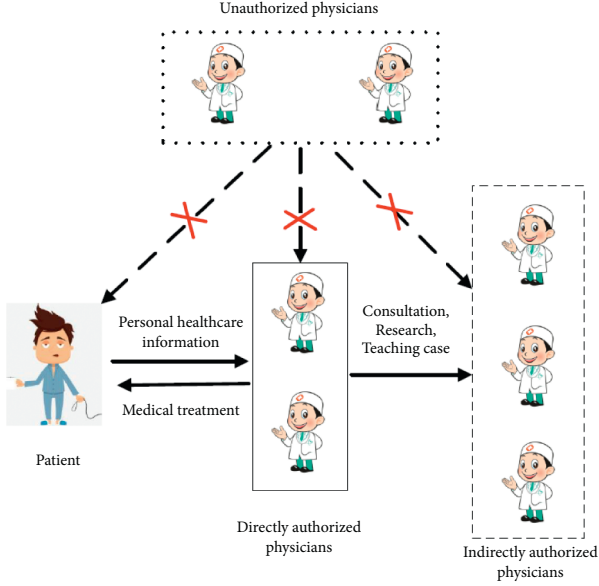


FIGURE 2: Our constructed HCPS system model.

not be seen in Hospital 2 and Dr. Alice is one of his authorized physicians. Besides, for other purposes (e.g., research and medical consultation) in cooperation with Hospital 2 and research institutions C, Dr. Alice needs to generate two indistinguishable transcript simulations to Hospital 2 and research institutions C. In some cases, the patient may register at other hospitals with ambiguous assertion about his or her healthcare information; if there exists a medical dispute, the patient can be traced.

Remark 1. If the patient does not register twice, our system model is correct directly and the purpose of the twice registration is to solve some medical disputes.

4. Authorized Accessible Privacy Model

In this section, we introduce an authorized accessible privacy model for HCPS which includes a strong designated verifier double-authentication-preventing signature (SDVDAPS for short) and the corresponding security models.

4.1. Strong Designated Verifier Double-Authentication-Preventing Signatures. We provide a self-enhancement privacy-preserving authentication scheme based on a SDVDAPS to satisfy three security and privacy requirements in healthcare cyber-physical systems. Our SEAPA algorithm is described as follows:

KeyGen: on inputting public parameters Param and security parameter κ , the algorithm outputs public-private keys (PK_A, SK_A) of a patient (Alice, for example) and public-private keys (PK_B, SK_B) and (PK_C, SK_C) of two designated physicians (Bob and Eve, for example) in two different hospitals.

DVDAPSig: on inputting Alice's colliding personal healthcare information (m_0, p_1) and (m_0, p_2) , Alice's

secret key SK_A , Bob's public key SK_B , and Eve's public key PK_C , the algorithm generates a signature σ_1 on (m_0, p_1) by using SK_A and PK_B and a signature σ_2 on (m_0, p_2) by using SK_A and PK_C .

DVDAPVer: on inputting Alice's colliding personal healthcare information (m_0, p_1) and (m_0, p_2) , (σ_1, σ_2) , and PK_A, SK_B , and SK_C , the algorithm outputs 0 which means reject or outputs 1 which means accept.

Sim: the algorithm generates a signature on (m_0, p_1) by using (PK_A, SK_B, PK_B) of an authorized physician Bob in Hospital 1 and a signature on (m_0, p_2) by using (PK_A, SK_C, PK_C) of an authorized physician Bob in other hospitals which are indistinguishable from those produced by $\text{DVDAPSig}((m_0, p_1), SK_A, PK_B)$ and $\text{DVDAPSig}((m_0, p_2), SK_A, PK_C)$, respectively.

Extract: on inputting Alice's colliding personal healthcare information (m_0, p_1) and (m_0, p_2) and a valid signature pair (σ_1, σ_2) , the algorithm could extract Alice's secret key SK_A .

Remark 2. If the patient does not register twice, the SDVDAPS will degenerate into a general designated verifier signature (DVS for short). Hence, we can get that our construction is correct directly:

Correctness. We need the SEAPA to be correct which means that any honestly computed signature can be verified by directly authorized physicians. That is to say, for any $\kappa > 0$, $(PK_i, SK_i) \leftarrow \text{KeyGen}(1^\kappa)$ for $i = A, B, C$ and $(m_0, p_i) \in \mathcal{M}$ for $i = 1, 2$, for $\delta_1 \leftarrow \text{DVDAPSig}((m_0, p_1), SK_A, PK_B)$ and $\delta_2 \leftarrow \text{DVDAPSig}((m_0, p_2), SK_A, PK_C)$, it holds that

$$\begin{aligned} \Pr[\text{DVDAPVer}((m_0, p_1), \sigma_1, PK_A, SK_B) = 1] &= 1, \\ \Pr[\text{DVDAPVer}((m_0, p_2), \sigma_2, PK_A, SK_C) = 1] &= 1. \end{aligned} \quad (3)$$

4.2. Security Models

4.2.1. Unforgeability. In the DVDAPS, unforgeability under chosen message attack is a basic security property, which means it is infeasible to produce a valid signature for any adversary who does not know secret keys of the signer. Then, we provide a formal description of existential unforgeability of the SEAPA.

(1) **Definition 1 (Unforgeability).** Our construction SEAPA shows unforgeability under chosen message attack if any adversary \mathcal{A} could not win the following game.

\mathcal{C} constructs public/private key pairs $(PK_i, SK_i) \leftarrow \text{KeyGen}(1^\kappa)$ for $i = A, B, C$, where κ is a security parameter and sends (PK_A, SK_B) and (PK_A, SK_C) to \mathcal{A} , where A is the patient and B and C are corresponding authorized physicians in different hospitals, respectively.

\mathcal{A} queries the signing oracle q_{s_1} times for the message (m_0, p_{i1}) and q_{s_2} times for the message (m_0, p_{i2}) , respectively.

\mathcal{C} answers \mathcal{A} 's queries by

$$\sigma_{i1} = \text{DVDAPSign}(\text{PK}_A, \text{SK}_A, \text{PK}_B, (m_0, p_{i1})), \quad (4)$$

and

$$\sigma_{i2} = \text{DVDAPSign}(\text{PK}_A, \text{SK}_A, \text{PK}_C, (m_0, p_{i2})), \quad (5)$$

respectively.

Finally, \mathcal{A} is successful if he outputs two new signatures σ_1^* and σ_2^* for message (m_0, μ^*) .

For running the above games in time t , the SEAPA shows (t, ε) unforgeability (EUF-CMA secure) if there exists a negligible function $\varepsilon(\kappa) > 0$ such that the following equation holds:

$$\Pr[\text{DVDAPVer}_{\text{SEAPA}, \mathcal{A}}^{\text{EUF-CMA}}((m_0, \mu_1^*), (m_0, \mu_2^*), \sigma_1^*, \sigma_2^*) = 1] \leq \varepsilon(\kappa). \quad (6)$$

4.2.2. Anonymity for the Patient. Only the authorized physicians could generate an indistinguishable signature from the one that could be produced by the signer.

(2). *Definition 2* (Anonymity for the Patient). The SEAPA shows anonymity for the patient if the game is successful between a PPT adversary \mathcal{A} and a distinguisher \mathcal{D} as follows.

\mathcal{C} constructs public/private key pairs $(\text{PK}_i, \text{SK}_i) \leftarrow \text{Key Gen}(1^\kappa)$ for $i = A, B, C$, where κ is a security parameter and sends public key pairs $(\text{PK}_A, \text{PK}_B)$ and $(\text{PK}_A, \text{PK}_C)$ to \mathcal{D} , where A is the patient and B and C are corresponding physicians in different hospitals, respectively.

\mathcal{D} queries the signing oracle q_{s_1} times for the message (m_0, p_{i1}) and q_{s_2} times for the message (m_0, p_{i2}) , respectively, where $q_s = \max\{q_{s_1}, q_{s_2}\}$.

\mathcal{C} answers \mathcal{A} 's query by

$$\sigma_{i1} = \text{DVDAPSign}(\text{PK}_A, \text{SK}_A, \text{PK}_B, (m_0, p_{i1})), \quad (7)$$

$$\sigma_{i2} = \text{DVDAPSign}(\text{PK}_A, \text{SK}_A, \text{PK}_C, (m_0, p_{i2})), \quad (8)$$

respectively.

\mathcal{D} makes queries on new messages (m_0, p_1^*) and (m_0, p_2^*) to obtain corresponding challenging signatures σ_1^* and σ_2^* .

\mathcal{C} tosses a coin $b \in \{0, 1\}$. If $b = 0$, \mathcal{C} runs DVDAPSign algorithm and returns corresponding signatures:

$$\sigma_1^* = \text{DVDAPSign}(\text{PK}_A, \text{SK}_A, \text{PK}_B, (m_0, p_1^*)), \quad (9)$$

$$\sigma_2^* = \text{DVDAPSign}(\text{PK}_A, \text{SK}_A, \text{PK}_C, (m_0, p_2^*)), \quad (10)$$

Else, \mathcal{C} runs Sim algorithm and returns

$$\sigma_1^* = \text{Sim}(\text{PK}_A, \text{PK}_B, \text{SK}_A, (m_0, p_1^*)), \quad (11)$$

$$\sigma_2^* = \text{Sim}(\text{PK}_A, \text{PK}_C, \text{SK}_A, (m_0, p_2^*)), \quad (12)$$

respectively.

\mathcal{D} is able to query other new messages except for (m_0, p_1^*) and (m_0, p_2^*) after receiving challenging signatures σ_1^* and σ_2^* .

Finally, \mathcal{D} is successful if he outputs $b' = b$.

For running the above games in time t , the construction SEAPA shows (t, q_s) anonymity for the patient against a chosen message distinguisher if there exists a negligible function $\varepsilon(\kappa) > 0$ such that the following equation holds:

$$\text{Adv}_{\text{DVDAPS}, \mathcal{D}}^{\text{NT-CMA}} = \left| \Pr[b' = b] - \frac{1}{2} \right| \leq \varepsilon(\kappa). \quad (13)$$

The security model of anonymity for the patient means that the probability of the signature produced by the DVDAPSign algorithm and the Sim algorithm is the same.

Extractability (or punishability) could be interpreted as Alice's secret keys can be extracted if there exists a medical dispute between Bob and Eve. To some extent, extractability can be considered as a self-enhancement mechanism for the patient.

(3). *Definition 3* (Self-Enhancement for the Patient). A SEAPA provides self-enhancement for the patient, if for any PPT adversary \mathcal{A} , there exists a negligible function $\varepsilon(\kappa)$ such that the following probability is negligible:

$$\Pr \left[\begin{array}{l} (\text{SK}_i, \text{PK}_i) \leftarrow \text{Key Gen}(1^\kappa) \\ \text{for } i = A, B, C \\ ((m_0, p_1), \sigma_1) \leftarrow \mathcal{A}(\text{PK}_A, \text{SK}_A, \text{PK}_B), \\ ((m_0, p_2), \sigma_2) \leftarrow \mathcal{A}(\text{PK}_A, \text{SK}_A, \text{PK}_C) \\ \text{SK}' \neq \text{SK} \end{array} \right] \quad (14)$$

$$\left. \begin{array}{l} p_1 \neq p_2 \wedge p_1 \neq 0, p_2 \neq 0 \wedge \\ \text{SK}' \leftarrow \text{Extract}(m_0, p_1, p_2, \sigma_1, \sigma_2) \\ \wedge \text{DVDAPVer}(m_0, p_1, \delta_1, \text{PK}_A, \text{SK}_B) = 1 \\ \wedge \text{DVDAPVer}(m_0, p_2, \delta_2, \text{PK}_A, \text{SK}_C) = 1 \end{array} \right\} \leq \varepsilon(\kappa).$$

5. Our Lattice-Based SEAPA Construction

In this section, we will introduce our lattice-based SEAPA construction in detail. Our protocol consists of the following five phases.

5.1. KeyGen. $s_i \leftarrow R_{-d,d}^m$ for the patient A (name Alice) and directly authorized physicians B (name Bob) and C (name Eve) in different hospitals:

- (1) $\mathbf{a} \leftarrow R_q^m$, $\mathbf{t} \leftarrow \mathbf{as}$, and cryptographically collision-resistant hash functions $H: \{0, 1\}^* \rightarrow \{v: v \in R_{-1,1}, \|v\|_1 \leq t\}$.
- (2) Compute $\mathbf{y}_A = \mathbf{a}^T \mathbf{s}_A$, $\mathbf{y}_B = \mathbf{a} \mathbf{s}_B$, and $\mathbf{y}_C = \mathbf{a} \mathbf{s}_C$. Let $(\mathbf{a}, H, d, m, R, t)$ be public parameters.
- (3) Alice's public/private pair is $(\mathbf{y}_A, \mathbf{s}_A)$, Bob's public/private pair is $(\mathbf{y}_B, \mathbf{s}_B)$, and Eve's public/private pair is $(\mathbf{y}_C, \mathbf{s}_C)$.

5.2. *DVDAPSig*. Given the colliding patient's personal health information (m_0, p_1) and (m_0, p_2) which can only be verified and recovered by B and C , respectively, Alice computes a signature σ_1 on (m_0, p_1) by using $\mathbf{s}_A, \mathbf{y}_B$ and a signature σ_2 on (m_0, p_2) by using $\mathbf{s}_A, \mathbf{y}_C$ in the following:

- (1) $\mathbf{r} \leftarrow D_\sigma^m$
- (2) if \mathbf{r} is irreversible, goto step 1
- (3) $\mathbf{y} \leftarrow D_\sigma^m$
- (4) $c_1 \leftarrow H(m_0 \mathbf{y}_B^T \mathbf{y}, p_1)$
- (5) $c_2 \leftarrow H(m_0 \mathbf{y}_C^T \mathbf{y}, p_2)$
- (6) For $i = 1, 2$, compute
- (7) $\mathbf{z}_i \leftarrow \mathbf{s}_A c_i + m_0 \mathbf{y} \mathbf{r}^{-1}$
- (8) Output $(\mathbf{z}_i, c_i, \mathbf{r})$ with probability $\min(D_\sigma^m(\mathbf{z}_i)/MD_{s_{c_i}, \sigma}^m(\mathbf{z}_i), 1)$, where $M_i = O(1)$ and $M = \max\{M_1, M_2\}$
- (9) Then, Alice sends $\sigma_1 = (\mathbf{z}_1, c_1, \mathbf{r})$ to Bob and sends $\sigma_2 = (\mathbf{z}_2, c_2, \mathbf{r})$ to Eve

5.3. *DVDAPVer*. After receiving signatures, physicians Bob and Eve working in the different local healthcare providers do the following things respectively:

- (i) Bob accepts iff $\|\mathbf{z}_1\| \leq 2\sigma\sqrt{m}$ and $c_1 = H(\mathbf{s}_B^T(\mathbf{a}\mathbf{z}_1 - \mathbf{y}_A c_1)\mathbf{r}, p_1)$
- (ii) Eve accepts iff $\|\mathbf{z}_2\| \leq 2\sigma\sqrt{m}$ and $c_2 = H(\mathbf{s}_C^T(\mathbf{a}\mathbf{z}_2 - \mathbf{y}_A c_2)\mathbf{r}, p_2)$

5.4. *Sim*

- (i) Since $\mathbf{s}_B^T(\mathbf{a}\mathbf{z}_1 \mathbf{r} - \mathbf{y}_A c_1 \mathbf{r}) = \mathbf{s}_B^T(\mathbf{a}\mathbf{z}'_1 - s_A c'_1)$, Bob computes simulated signature $\sigma'_1 = (\mathbf{z}'_1, c'_1, \mathbf{r})$, where $\mathbf{z}'_1 = \mathbf{z}_1 \mathbf{r}, c'_1 = c_1 \mathbf{r}$
- (ii) Since $\mathbf{s}_C^T(\mathbf{a}\mathbf{z}_2 \mathbf{r} - \mathbf{y}_A c_2 \mathbf{r}) = \mathbf{s}_C^T(\mathbf{a}\mathbf{z}'_2 - \mathbf{y}_A c'_2)$, Eve computes simulated signature $\sigma'_2 = (\mathbf{z}'_2, c'_2, \mathbf{r})$

5.5. *Extract*

- (i) When there exists a medical dispute between authorized physicians and the patient, they provide their valid signature $(\mathbf{z}_1, c_1, \mathbf{r})$ and $(\mathbf{z}_2, c_2, \mathbf{r})$ to the public and anyone can compute the patient's secret key $\mathbf{s}_A = \mathbf{z}_1 - \mathbf{z}_2/c_1 - c_2$.

6. Security Proof

In this section, we use the random oracle model to prove the security of our proposed scheme based on the security model of SEAPA.

Theorem 1. *The proposed SEAPA is correct.*

Proof. For $i = B, j = 1$ and $i = C, j = 2$, the correctness is given as follows:

$$\begin{aligned} & \mathbf{s}_i^T(\mathbf{a}\mathbf{z}_j - \mathbf{y}_A c_j)\mathbf{r} \\ &= \mathbf{s}_i^T(\mathbf{a}(\mathbf{s}_A c_j + m_0 \mathbf{y} \mathbf{r}^{-1}) - \mathbf{y}_A c_j)\mathbf{r} \\ &= (\mathbf{s}_i^T \mathbf{a} \mathbf{s}_A c_j + m_0 \mathbf{s}_i^T \mathbf{a} \mathbf{y} \mathbf{r}^{-1} - \mathbf{s}_i^T \mathbf{a} \mathbf{y}_A c_j)\mathbf{r} \\ &= \mathbf{y}_i^T \mathbf{y} m_0. \end{aligned} \quad (15)$$

Hence, it follows that

$$H(\mathbf{y}_B^T \mathbf{y} m_0, p_1) = c_1 = H(\mathbf{s}_B^T(\mathbf{a}\mathbf{z}_1 - \mathbf{y}_A c_1)\mathbf{r}, p_1), \quad (16)$$

$$H(\mathbf{y}_C^T \mathbf{y} m_0, p_2) = c_2 = H(\mathbf{s}_C^T(\mathbf{a}\mathbf{z}_2 - \mathbf{y}_A c_2)\mathbf{r}, p_2). \quad (17) \quad \square$$

Theorem 2. *The proposed SEAPA is unforgeable against chosen message attack under the hardness of the Ring-SIS.*

Proof. Suppose that a PPT adversary \mathcal{A} is able to produce a valid signature $\sigma^* = (\mathbf{z}_1^*, c_1^*, \mathbf{z}_2^*, c_2^*, \mathbf{r}^*)$. According to EUF-CMA game, σ^* can be correctly verified which means that \mathcal{A} can compute the following equation.

For $i = B, j = 1$ and $i = C, j = 2$,

$$\begin{aligned} & \mathbf{s}_i^T(\mathbf{a}\mathbf{z}_j^* - \mathbf{y}_A c_j^*)\mathbf{r}^* \mathbf{r}^{-*} - \mathbf{y}_i^T \mathbf{z}_j^* \\ &= -\mathbf{s}_i^T \mathbf{y}_A c_j^* \bmod q \\ &= -\mathbf{y}_i^T \mathbf{s}_A c_j^* \bmod q. \end{aligned} \quad (18)$$

Let $\mathbf{X} = -\mathbf{y}_i^T \mathbf{s}_A c_j^* \bmod q$. If $\|\mathbf{s}_A c_j^*\| \leq \beta$, \mathcal{A} obtains a solution for Ring-SIS problem.

If an adversary \mathcal{A} could obtain a valid SEAPA signature by EUF-CMA game in time t , he can solve the Ring-SIS problem in polynomial time. Hence, we have

$$\begin{aligned} & \Pr[\text{DVDAPVer}_{\text{SEAPA}, \mathcal{A}}^{\text{EUF-CMA}}((m_0, \mu_1^*), (m_0, \mu_2^*), \sigma_1^*, \sigma_2^*) = 1] \\ &= \Pr[\mathbf{X} = -\mathbf{y}_i^T \mathbf{s}_A c_j^*: \|\mathbf{s}_A c_j^*\| \leq \beta] \leq \varepsilon(\kappa). \end{aligned} \quad (19) \quad \square$$

Theorem 3. *The proposed SEAPA shows anonymity for the patient.*

Proof. According to the proposed scheme, anonymity for the patient means that any valid signature on a message produced by the Sim algorithm in SEAPA is indistinguishable from the signature produced by the DVDAPSign algorithm. That is to say, the probability of the signature produced by the two algorithms are the same.

Let $\bar{\sigma} = (\bar{\mathbf{z}}_1, \bar{c}_1, \bar{\mathbf{z}}_2, \bar{c}_2, \bar{\mathbf{r}})$ be a valid signature, and some signatures are chosen randomly from the set of DVDAPSign. The probability of the signature $\sigma = (\mathbf{z}_1, c_1, \mathbf{r})$ produced by the DVDAPSign is given by

$$\Pr \left[\begin{array}{c} \mathbf{r} \leftarrow D_\sigma^m, \\ \mathbf{y} \leftarrow D_\sigma^m, \\ \bar{\sigma} = \sigma: c_1 \leftarrow H(m_0 \mathbf{y}_B^T \mathbf{y}, p_1) \\ \mathbf{z}_1 \leftarrow \mathbf{s}_A c_1 + m_0 \mathbf{y} \mathbf{r}^{-1} \end{array} \right] = \frac{1}{\gamma^m (\gamma^m - 1)}. \quad (20)$$

For randomly selected $y' \leftarrow D_\sigma^m$, the signature $\sigma' = (\mathbf{z}'_1, v, c'_1, \mathbf{r})$ produced by the Sim is given by

$$\Pr \left[\begin{array}{c} \mathbf{z}'_1, c'_1 \leftarrow D_\sigma^m, \\ \bar{\sigma} = \sigma' : c_1 \leftarrow H(\mathbf{s}_B^T (\mathbf{a} \mathbf{z}'_1 - \mathbf{y}_A c'_1), p_1) \\ \mathbf{z}_1 = \mathbf{z}'_1 \mathbf{r}^{-1} \end{array} \right] = \frac{1}{\gamma^m (\gamma^m - 1)}. \quad (21)$$

In a similar way,

$$\Pr \left[\begin{array}{c} \mathbf{r} \leftarrow D_\sigma^m, \\ \mathbf{y} \leftarrow D_\sigma^m, \\ \bar{\sigma} = \sigma: c_2 \leftarrow H(m_0 \mathbf{y}_C^T \mathbf{y}, p_2) \\ \mathbf{z}_2 \leftarrow \mathbf{s}_A c_2 + m_0 \mathbf{y} \mathbf{r}^{-1} \end{array} \right], \quad (22)$$

$$\Pr \left[\begin{array}{c} \mathbf{z}'_2, c'_2 \leftarrow D_\sigma^m, \\ \bar{\sigma} = \sigma': c_2 \leftarrow H(\mathbf{s}_C^T (\mathbf{a} \mathbf{z}'_2 - \mathbf{y}_A c'_2), p_2) \\ \mathbf{z}_2 = \mathbf{z}'_2 \mathbf{r}^{-1} \end{array} \right].$$

Therefore,

$$\text{Adv}_{\text{SEAPA}, \mathcal{D}} = \left| \Pr[b' = b] - \frac{1}{2} \right| \leq \varepsilon(\kappa). \quad (23)$$

That is, the proposed SEAPA scheme shows anonymity for the patient. \square

Theorem 4. *The proposed SEAPA scheme shows self-enhancement for the patient.*

Proof. It is easy to verify the extractability from the algorithm Extract based on Theorem 1. So, if the signature private keys are important for her, she will not make some ambiguous assertions which means that there does not exist colliding personal health information (m_0, p_1) and (m_0, p_2) ; that is to say, $p_1 = p_2$. To some extent, there will not occur some medical disputes. Hence, our proposed scheme shows self-enhancement for the patient.

By going through the criteria of Wang et al., Hussain et al., and Bonneau et al. [30–32], we propose some major categories that our scheme satisfies as shown in Table 1, where “Must Have” category is related to providing robust security, “May or May Not Have” category is dealing with user experience, and “Nice to Have” category has one criterion related to user experience which is sound repairability while the other is related to security. Besides these criteria, our scheme also satisfies postquantum security. \square

TABLE 1: Criterion categorization.

Criterion	Category
C1-No verifier table	Must have
C2-Password friendly	Must Have
C3-No password exposure	Must Have
C4-No smart card loss attack	Must Have
C5-Resistance to known attacks	Must Have
C7-Provision of key agreement	Must Have
C10-Mutual authentication	Must Have
C11-User anonymity	Must Have
C6-Sound repairability	Nice to Have
C12-Forward secrecy	Nice to Have
C8-No clock synchronization	Nice to Have
C9-Timely typo detection	May or May Not Have

TABLE 2: Parameter size in the SEAPA scheme.

	512	512	1024
n	512	512	1024
m	1024	1024	2048
q	2^{24}	2^{33}	8380417
d	1	31	1
k	1024	1024	2048
κ s.t. $2^\kappa \binom{n}{\kappa} \geq 2^{100}$	14	14	14
$\sigma \approx 12 \cdot d \cdot \kappa \cdot \sqrt{m}$	5376	166656	5376
$M \approx \exp(12 d \kappa \sqrt{m} / \sigma + (d \kappa \sqrt{m} / 2 \sigma)^2)$	2.72	2.72	2.72
Size of signature $\approx 2m \log(12\sigma)$ bits	33000	41100	82200
Size of secret key $\approx 2m^2 \log(2d+1)$ bits	$2^{20.5}$	$2^{22.5}$	$2^{23.7}$
Size of public key $\approx 1/2m^2 \log(q)$ bits	$2^{21.5}$	2^{22}	$2^{25.5}$

7. Concrete Parameters Analysis

7.1. Communication Cost. In our SEAPA scheme, we set up some parameters $q, n, m, \eta, \Phi, d, \beta$, and γ for postquantum computational security [33]. The security of our scheme is based on the hardness of the Ring-SIS $_{q,n,d,m,\beta}$ problem. In the scheme, we set $m = 2n$ and $4 d \beta \leq q$, and the Ring-SIS $_{q,n,d,m}$ problem can reduce to ℓ_2 -Ring-SIS $_{q,n,d,m,\beta}$. The definition of its parameters are listed in Table 2.

From Table 2, we can see that the signature size of the SEAPA scheme is about 4 KB, 5 KB, and 10 KB for different parameters, respectively.

7.2. Computational Cost. We execute our algorithms on Intel Core i7-4710 processor with 12 GB memory and Ubuntu Linux operating system. Some important cryptographic operations are implemented with NTLlib, which is a NTT-based fast lattice library. By statistics, these important algorithm operations mainly consist of one polynomial addition, one polynomial multiplication, and one polynomial Gaussian. Since the implementation of any hash function is not included in NTLlib, we test the running time of the hash function by a HMAC based on SM3 algorithm. The execution time of each cryptographic operation is shown in Table 3.

We use KG, Sig, Ver, Sim, and Ext to represent the five algorithms KeyGen, DVDAPSig, DVDAPVer, Sim, and Extract, respectively. The degree of polynomial we choose is 8, 128, 1024, 8192, and 32768, and the corresponding size of

TABLE 3: The cryptographic operation time in the SEAPA scheme (microseconds).

n	q	m	Poly addition	Poly subtraction	Poly multiplication	Poly Gaussian	Hash function
8	60	16	0.00456	0.00250	0.00333	0.00243	0.105
128	14	216	0.00335	0.00230	0.00243	0.00262	0.1100
1024	60	2048	0.00248	0.00369	0.00227	0.00256	0.175
8192	124	16384	0.00454	0.00475	0.00652	0.00474	0.177
32768	124	65536	0.0120	0.0120	0.02678	0.0375	0.287

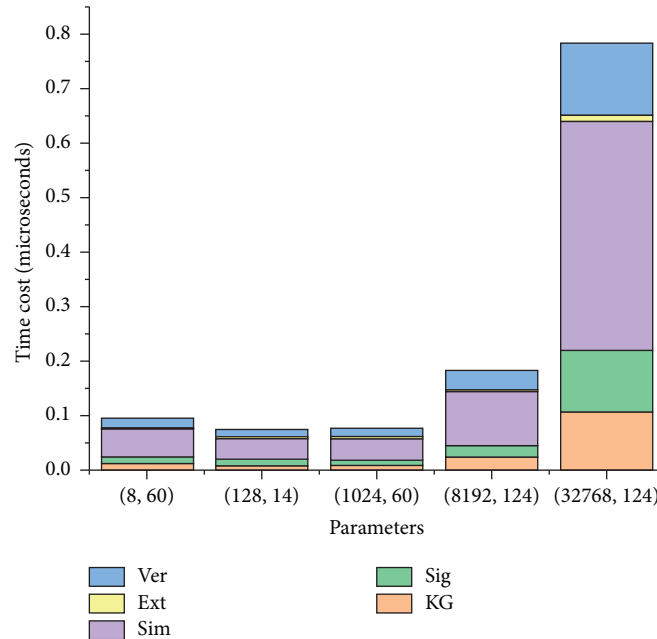


FIGURE 3: Time cost of each algorithm in SEAPA.

integer ring is 14 bits, 60 bits, 124 bits, and 124 bits. So the total running time of our algorithm for different parameters is about 0.096711 ms in keygen phase, 0.076315 ms in signature phase, 0.078821 ms in simulation phase, 0.184251 ms in extraction phase, and 0.784562 ms in verification phase. The execution result is depicted in Figure 3.

8. Conclusion and Future Work

In this paper, we presented an authorized accessible privacy model and provided a concept of the patient self-enhancement privacy-preserving authentication scheme. Our construction is derived from strong designated verifier signatures and double-authentication-preventing signatures based on lattice. Security proof shows that our construction satisfies different levels of security requirements in the HCPS system model. Concrete parameters analysis and performance evaluation demonstrated that our construction has reasonable efficiency for real applications. In future work, on the basis of lattice-based strong designated verifier signatures, we will provide some comparisons on the concrete parameters and the communication cost.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61872229, 61802239, 62062019, and 62074131), Key Research and Development Program of Shaanxi Province (2020ZDLGY09-06, 2021ZDLGY06-04, and 2021ZDLGY05-01), Natural Science Basic Research Plan in Shaanxi Province of China (2019JQ-667 and 2020JQ-422), and Shenzhen Fundamental Research Program (20210317191843003).

References

- [1] Y. Zhang, M. Qiu, C. W. Tsai, M. M. Hassan, and A. Alamri, "Health-CPS: healthcare cyber-physical system assisted by cloud and big data," *IEEE Systems Journal*, vol. 11, no. 1, pp. 88–95, 2015.
- [2] J. Xu, L. Wei, W. Wu, A. Wang, Y. Zhang, and F. Zhou, "Privacy-preserving data integrity verification by using lightweight streaming authenticated data structures for healthcare cyber-physical system," *Future Generation Computer Systems*, vol. 108, pp. 1287–1296, 2020.

- [3] S. Wang, H. Wang, J. Li et al., "A fast cp-abe system for cyber-physical security and privacy in mobile healthcare network," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4467–4477, 2020.
- [4] R. Agarwal and M. Hussain, "Generic framework for privacy preservation in cyber-physical systems," in *Proceedings of the Progress in Advanced Computing and Intelligent Engineering*, pp. 257–266, Springer, Odisha, India, October 2021.
- [5] H. Sedjelmaci, F. Guenab, S.-M. Senouci, H. Moustafa, J. Liu, and S. Han, "Cyber security based on artificial intelligence for cyber-physical systems," *IEEE Network*, vol. 34, no. 3, pp. 6–7, 2020.
- [6] C. Kannan, M. Dakshinamoorthy, M. Ramachandran, R. Patan, H. Kalyanaraman, and A. Kumar, "Cryptography-based deep artificial structure for secure communication using IoT-enabled cyber-physical system," *IET Communications*, vol. 15, no. 6, pp. 771–779, 2021.
- [7] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications," in *Proceedings of the Theory and Applications of Cryptographic Techniques*, vol. 143–154 Berlin, Heidelberg, Springer, May 1996.
- [8] L. Wei, H. Zhu, Z. Cao et al., "Security and privacy for storage and computation in cloud computing," *Information Sciences*, vol. 258, pp. 371–386, 2014.
- [9] Y. Zhang, Q. Liu, C. Tang, and H. Tian, "A lattice-based designated verifier signature for cloud computing," *International Journal of High Performance Computing and Networking*, vol. 8, no. 2, pp. 135–143, 2015.
- [10] S. Hou, X. Huang, J. K. Liu, J. Li, and L. Xu, "Universal designated verifier transitive signatures for graph-based big data," *Information Sciences*, vol. 318, pp. 144–156, 2015.
- [11] X. D. Yang, L. K. Xiao, C. L. Chen, and C. F. Wang, "A strong designated verifier proxy re-signature scheme for IoT environments," *Symmetry*, vol. 10, no. 11, pp. 1486–1491, 2018.
- [12] Y. Yu, Y. Ding, Y. Zhao et al., "LRCoin: leakage-resilient cryptocurrency based on bitcoin for data trading in IoT," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4702–4710, 2019.
- [13] L. Zuo, N. Kumar, H. Tu, A. Singh, N. Chilamkurti, and S. Rho, "Detection and analysis of secure intelligent universal designated verifier signature scheme for electronic voting system," *The Journal of Supercomputing*, vol. 70, no. 1, pp. 177–199, 2014.
- [14] P. Grontas, A. Pagourtzis, A. Zacharakis, and B. Zhang, "Towards everlasting privacy and efficient coercion resistance in remote electronic voting," in *Proceedings of the Financial Cryptography and Data Security*, pp. 210–231, Springer, Nieuwpoort, Curaçao, March 2018.
- [15] J. Zhou, X. Lin, X. Dong, and Z. Cao, "PSMPA: patient self-controllable and multi-level privacy-preserving cooperative authentication in distributed healthcare cloud computing system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1693–1703, 2014.
- [16] Z. Jun, C. Zhenfu, D. Xiaolei, L. Xiaodong, and A. V. Vasilakos, "Securing m-healthcare social networks: challenges, countermeasures and future directions," *IEEE Wireless Communications*, vol. 20, no. 4, pp. 12–21, 2013.
- [17] S. Saeednia, S. Kremer, and O. Markowitch, "An efficient strong designated verifier signature scheme," in *Proceedings of the ISC2003*, pp. 40–54, Springer, Seoul, Korea, November 2003.
- [18] L. Deng, Y. Yang, and R. Gao, "Certificateless designated verifier anonymous aggregate signature scheme for healthcare wireless sensor networks," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8897–8909, 2021.
- [19] Y. Li, W. Susilo, Y. Mu, and D. Pei, "Designated verifier signature: definition, framework and new constructions," in *Proceedings of the Ubiquitous Intelligence and Computing*, pp. 1191–1200, Springer, Hong Kong, China, July 2007.
- [20] Q. Huang, G. Yang, D. S. Wong, and W. Susilo, "Efficient strong designated verifier signature schemes without random oracle or with non-delegatability," *International Journal of Information Security*, vol. 10, no. 6, p. 373, 2006.
- [21] J.-G. Li, N. Qian, X.-Y. Huang, and Y.-C. Zhang, "Certificate-based strong designated verifier signature scheme," *Chinese Journal of Computers*, vol. 35, no. 8, pp. 1579–1587, 2012.
- [22] Y. Chen, Y. Zhao, H. Xiong, and F. Yue, "A certificateless strong designated verifier signature scheme with non-delegatability," *International Journal on Network Security*, vol. 19, no. 4, pp. 573–582, 2017.
- [23] Y. Shi, H. Fan, and Q. Liu, "An obfuscatable designated verifier signature scheme," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 2, pp. 271–285, 2017.
- [24] J. Cai, H. Jiang, P. Zhang, Z. Zheng, G. Lyu, and Q. Xu, "An efficient strong designated verifier signature based on \mathcal{R} -SIS assumption," *IEEE Access*, vol. 7, pp. 3938–3947, 2019.
- [25] J. Cai, H. Jiang, P. Zhang et al., "ID-based strong designated verifier signature over R-SIS assumption," *Security and Communication Networks*, vol. 2019, Article ID 9678095, 8 pages, 2019.
- [26] X. Huang, W. Susilo, Y. Mu, and F. Zhang, "Short (Identity-Based) strong designated verifier signature schemes," in *Proceedings of the Information Security Practice and Experience (ISPE)*, pp. 214–225, Springer, Hangzhou, China, April 2006.
- [27] L. He, J. Ma, L. Shen, and D. Wei, "Certificateless designated verifier proxy signature scheme for unmanned aerial vehicle networks," *Science China Information Sciences*, vol. 64, no. 1, pp. 1–15, 2021.
- [28] M. Zheng, K. Xue, S. Li, and N. Yu, "A practical quantum designated verifier signature scheme for E-voting applications," *Quantum Information Processing*, vol. 20, no. 7, pp. 1–22, 2021.
- [29] V. Lyubashevsky, "Lattice signatures without trapdoors," in *Proceedings of the Advances in Cryptology - EUROCRYPT 2012 TACT*, pp. 738–755, Springer, Cambridge, UK, April 2012.
- [30] J. Bonneau, C. Herley, P. Oorschot, and F. Stajano, "The quest to replace passwords: a framework for comparative evaluation of web authentication schemes," *IEEE Symposium on Security and Privacy*, pp. 1–15, 2012.
- [31] K. Hussain, N. Jhanjhi, H. M. ur-Rahman, J. Hussain, and M. Hasan Islam, "Using a systematic framework to critically analyze proposed smart card based two factor Authentication schemes," *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 4, pp. 417–425, 2021.
- [32] D. Wang and P. Wang, "two birds with one stone: two-factor Authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708–722, 2016.
- [33] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review*, vol. 41, no. 2, pp. 303–332, 1999.

Research Article

Anomaly Detection of System Call Sequence Based on Dynamic Features and Relaxed-SVM

Xiaoyao Liao,¹ Changzhi Wang ,^{1,2} and Wen Chen¹

¹School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China

²Zhihuiyuntian Technology, Chengdu 610031, China

Correspondence should be addressed to Changzhi Wang; wchzhi@gmail.com

Received 17 September 2021; Revised 29 November 2021; Accepted 21 December 2021; Published 1 February 2022

Academic Editor: Chunhua Su

Copyright © 2022 Xiaoyao Liao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The system call sequences of processes are important for host-based anomaly detection. However, the detection accuracy can be seriously degenerated by the subsequences which simultaneously appeared in the call sequences of both normal and abnormal processes. Furthermore, the detection may be obstructed especially when the normal/abnormal distributions of subsequences are extremely imbalanced along with many ambiguous samples. In the paper, the system call sequences are divided into weighted subsequences with fixed-length. Secondly, a suffix tree of each system call sequence is constructed to automatically extract the variable-length subsequence from the longest repeated substring of the tree. The frequencies of the fixed-and variable-length subsequences that appeared in each system call sequence constitute its feature vector. Finally, vectors are input into a cost-sensitive and relaxed support vector machine, in which the penalty-free slack of the relaxed SVM is split independently between the two classes with different weights. The experimental results on two public datasets ADFA-LD and UNM showed that the AUC of the proposed method can reach 99%, while the false alarm rate is only 2.4%.

1. Introduction

System calls provide interfaces between system functions and the user applications. And the call sequences can reflect the targets of process actions. Accordingly, the system call sequences stored in various auditing and logging systems are important intrusion detection objects [1, 2].

Usually, system call sequences are broken into subsequences and submitted to classification models [3]. Jewell and Beaver [4] proposed that the unique sequences of system calls are the basis for discriminating normal and abnormal behaviors of processes. Helman and Bhangoo [5] et al. defined the priority of system call sequence based on the probability of the occurrence of system calls to get typical features. Lee and Stolfo [6] performed machine learning tasks on operating system call sequences of normal and abnormal executions of the UNIX Sendmail program. Xie et al. [7–9] attempted to reduce the dimension of the frequency vectors of subsequences based on PCA to enhance the computational efficiency. Haider et al. [10] proposed to

take both the rarest repeating subsequence and the most frequent repeating subsequence in the system call sequence as statistical features. Kan et al. [11] proposed a novel IoT network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network (APSO-CNN). Bian et al. [12] extracted graph-based features from host authentication logs, which are then employed in the detection of APT targets in the network. Shin and Kim [13] preprocessed the collected data using n-gram to overcome the limitations of the sequence time delay embedding (STIDE) algorithm for host intrusion detection system (HIDS).

Recently inspired by linguistics mining in NLP (Nature language process), researchers started to analyze the semantic relations between subsequences. Forrest and Hofmeyr [1] defined the sequences as phrases composed of words (system calls), and then utilized artificial immune systems to classify the phrases. Creech and Hu [14] proposed to draw semantic features of the system call sequences (phrases) using context-free grammar and built an extreme

learning machine for the classifications. Liao and Vemuri [15] calculated the TF-IDF scores of system call sequences and input them into the K-NN model for abnormal detection. Zhang and Shen [16] built an improved TF-IDF model of subsequences, which takes both the time information and the correlation between the processes into consideration. Marteau [17] proposed a similarity measurement method to evaluate the similarity between symbolic sequences. Ambusaidi et al. [18] proposed a supervised feature selection algorithm, which is able to handle both linearly and nonlinearly related data features. Shams et al. [19] designed a new context-aware feature extraction method for convolutional neural network (CNN)-based multiclass intrusion detections. Subba [20] combined TF-IDF vectorizer and singular value decomposition (SVD) to design a novel HIDS framework for anomalous system processes detection.

However, it is difficult to select appropriate subsequences of system calls to discover the real purpose of the calling actions. Laszka et al. [21] proved that the optimal length of subsequence is highly dependent on data and applications and needs to be carefully fine-tuned. For example, if the subsequence is too short then we may get an incomplete calling trace. On the other hand, if the subsequence is too long, then the malicious calls are mixed with many normal system calls, and the extracted features may be disturbed. Finally, there are many normal call sequences while the abnormal calls are extremely few, and many unusual API sequences, along with incomplete sequences may have a strong influence on the classification model. How to deal with the imbalance and noise data is also a big challenge.

In view of the above problems, not only the semantic information contained in short subsequence but also the representative features in variable-length subsequences are considered to generate combined features. Usually, system call sequences contain repeated subsequences, which are regarded as program-specific behavior patterns. And the length of repeated subsequences is different among call sequences. In order to generate variable-length repeating sub-sequences, a suffix tree is constructed for each system call sequence and the longest repeating substring is automatically extracted from the subtrees. Furthermore, to address the imbalance and noisy subsequences of normal and abnormal calls, the widely used relaxed support vector machine (RSVM) is improved by assigning different weights and free slack amounts to the positive and negative classes, which are scaled by the sizes of the two classes to reduce the influence of data imbalance and outliers.

2. Feature Extraction

In order to automatically extract the call subsequences and related features, which can reflect the real target of the system calls, we present a dynamic feature extraction method for system call sequences. As shown in Figure 1 the features are extracted from both fixed-length and variable-length subsequences. In the first step, the training sequences are split into subsequences by n-gram [22], and each subsequence is weighted by TF-IDF [23]. Then, the first n

subsequences with big weight values are selected to composite a fixed-length subsequence set. In the second step, suffix trees are constructed for training sequences. The longest repeating substrings in the suffix trees are selected as variable-length subsequences. In the third step, both the fixed-length and variable-length subsequences are combined to get a corpus set. In step 4, the occurrence frequencies of the corpus subsequences in each system call sequences are counted to constitute feature vectors. Furthermore, in step 5, an autoencoder [24] is also utilized to reduce the vectors' dimension before being submitted to the classifiers in step 6.

2.1. Fixed-Length Subsequence Based on Semantics. The TF-IDF (term frequency-inverse document frequency) [23] is commonly utilized in text mining to evaluate the importance of every single word or phrase in a document. In this paper, the TF-IDF is employed to evaluate the system calls, which have been coded into word sequences. As described in [1], firstly, each system call is represented by a unique word (or number), accordingly, the call sequences become word sequences. As defined in (1), \overrightarrow{seq}_i is the j^{th} element in the set.

Definition 1. Inverse ratio of sequence frequency. The inverse ratio of a sequence frequency is the IDF reverse file frequency defined in TF-IDF. As shown in equation (1), N is the number of training sequences, and n_{t_i} is the number of fixed-length subsequence t_i that appeared in the training set.

$$idf(t_i, \overrightarrow{seq}_i) = \frac{N}{n_{t_i}}. \quad (1)$$

Definition 2. Vocabulary frequency of single sequence. As shown in equation (2), fre_{ij} represents the frequency of the fixed-length subsequence t_i in a system call sequence \overrightarrow{seq}_i . And the vector Fre_j represents the frequency of all the fixed-length subsequences $t = \{t_1, t_2, \dots, t_m\}$ in the system call sequence \overrightarrow{seq}_j .

$$fre_{ij} = tf(t_i, \overrightarrow{seq}_j) = \frac{\text{the frequency of } t_i}{\text{the number of word in the } \overrightarrow{seq}_j}. \quad (2)$$

$$Fre_j = [fre_{1j}, fre_{2j}, \dots, fre_{mj}]. \quad (3)$$

Definition 3. Process behavior weight. It is the combination of inverse ratio and vocabulary frequency of every single subsequence, as shown in (4), to evaluate the importance of a single vocabulary (subsequence) t_i to one system call sequence \overrightarrow{seq}_j .

$$\text{weight}(t_i, \overrightarrow{seq}_j) = tf(t_i, \overrightarrow{seq}_j) \times \log\left(\frac{N}{n_{t_i}} + 0.01\right). \quad (4)$$

Definition 4. Fixed length corpus of system call sequences. The subsequences with the first three highest weights, calculated by (4), in a single process are included in the fixed-length sequence corpus. In equation (5), where t_{ji} represents

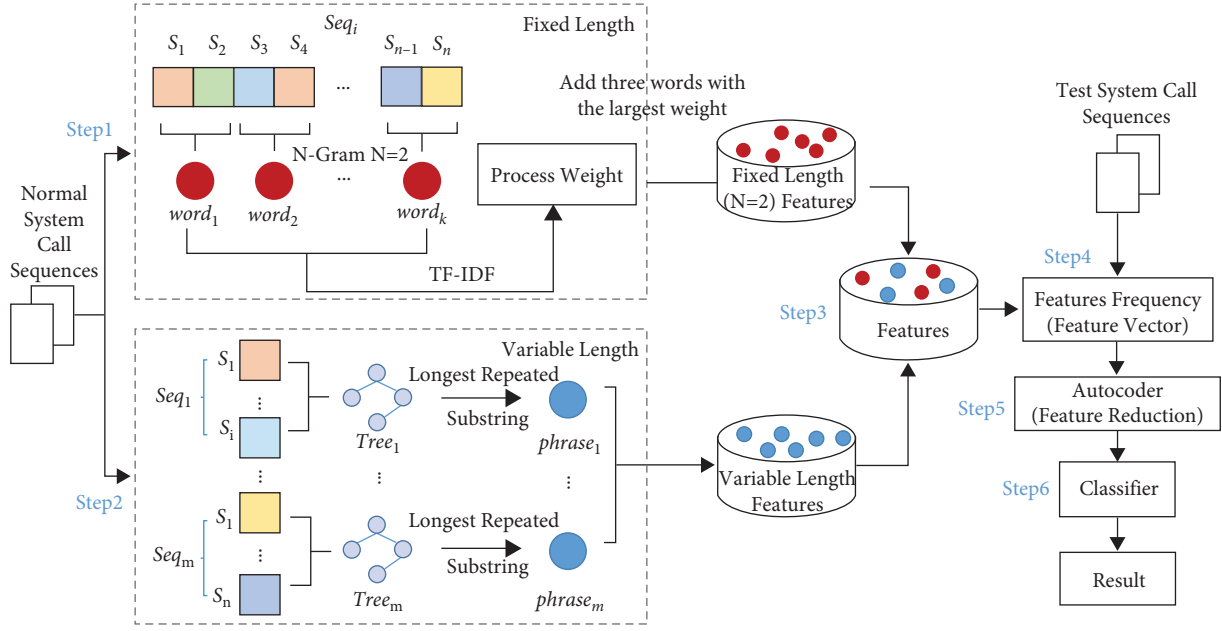


FIGURE 1: Dynamic feature extraction.

the i^{th} fixed-length subsequence in the system call sequence \vec{seq}_j .

$$\text{corpus}_{\text{fixed}} = [t_{11}, t_{12}, t_{13}, \dots, t_{n1}, t_{n2}, t_{n3}]. \quad (5)$$

2.2. Variable-Length Subsequence. Usually, the repeated subsequence in a system call sequence can reflect the behavior patterns of processes, which are much useful for abnormal detection. In the paper, these sequence fragments are defined as the representative features with variable lengths.

2.2.1. Constructing Suffix Tree. Ukkonen [25] is a commonly used suffix tree construction algorithm based on path compression and suffix chain. In the paper, Ukkonen is utilized to construct suffix trees from single system call sequences. Let a sequence $seq = "6, 4, 1, 4, 1, 4, 3."$ Firstly, the suffix tree for initial string $S = "6"$, $c = \text{null}$ (S is the current string for constructing suffix tree, $p = "1, 4"$ is the next character) is shown in Figure 2(a); and then, $S = "6," c = "4,"$ so all possible suffixes of $S + c$ are $seq_1 = "4," seq_2 = "6, 4,"$ as shown in Figure 2(b); finally, $S = "6, 4," c = "1,"$ so all possible suffixes of $S + c$ are $seq_1 = "1," seq_2 = "4, 1,"$ and $seq_3 = "6, 4, 1"$ as shown in Figure 2(c). In the same way, the result of the suffix tree for $seq = "6, 4, 1, 4, 1, 4, 3"$ is shown in Figure 3.

2.2.2. Search the Longest Repeating Substring. After the establishment of a suffix tree, the longest repeating substring p_k of the tree is selected to represent the behavior patterns of a system call sequence. As shown in Figure 4, for $seq = "6, 4, 1, 4, 1, 4, 3"$ the deepest nonleaf node is node "4" and the

longest repeating substring is $p = "1, 4,"$ which is incorporated into the variable-length sequence set $\text{corpus}_{\text{variable}}$.

$$\text{corpus}_{\text{variable}} = [p_1, p_2, \dots, p_n]. \quad (6)$$

2.2.3. Segmenting Long System Call Sequence. To alleviate the effect of long system call sequences on the suffix trees' generation efficiency, the long sequences are segmented into subsequences. As shown in Figure 5, for the system call sequence $seq_i = \{s_1, s_2, \dots, s_{500}, \dots, s_{len}\}$ with length $len > 500$. The sequence seq_i is divided into subsequence $\{seq_{i1}, seq_{i2}, \dots, seq_{ij}\}$. Let seq_{ij} represents the j^{th} subsequence of seq_i . The suffix trees $\{Tree_{i1}, Tree_{i2}, \dots, Tree_{ij}\}$ are constructed for each subsequence $\{seq_{i1}, seq_{i2}, \dots, seq_{ij}\}$.

2.2.4. Generating Corpus. Both the fixed-length subsequence set $\text{corpus}_{\text{fixed}}$ and the variable-length subsequence set $\text{corpus}_{\text{variable}}$ constitute a combination set Corpus to represent the behavior patterns of the system calls.

$$\text{Corpus} = \text{corpus}_{\text{fixed}} \cup \text{corpus}_{\text{variable}}. \quad (7)$$

2.2.5. Feature Extraction. Firstly, the frequency of the subsequences defined in (8) that appeared in each system call sequence is counted by AC automaton (Aho–Corasick automaton) [24]. Let $fre_{t_{ij}}$ represents the occurrence frequency of fixed-length subsequence t_i in a system call sequence seq_j . $fre_{p_{ij}}$ represents the frequency of a variable-length subsequence p_i in seq_j . The frequency vectors of all the call sequences constitute a feature matrix.

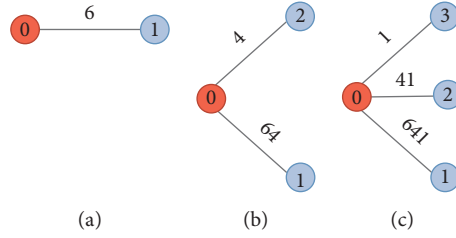


FIGURE 2: Constructing suffix tree.

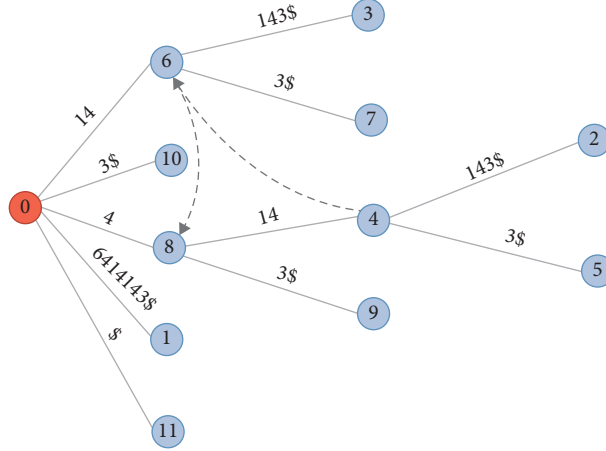


FIGURE 3: Constructed suffix tree.

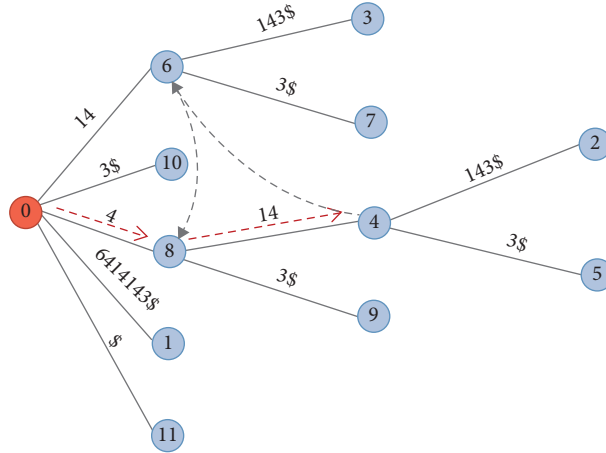


FIGURE 4: The longest repeating substring.

$$Vec = \begin{bmatrix} fre_{t_{11}} & \cdots & fre_{t_{1k}} & fre_{p_{11}} & \cdots & fre_{p_{1m}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ fre_{t_{j1}} & \cdots & fre_{t_{jk}} & fre_{p_{j1}} & \cdots & fre_{p_{jm}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ fre_{t_{n1}} & \cdots & fre_{t_{nk}} & fre_{p_{n1}} & \cdots & fre_{p_{nm}} \end{bmatrix}. \quad (8)$$

With the increasing of system call sequences, the number of subsequences in corpus also increases dramatically. In order to control the dimension of the feature vectors and fascinate mining of the potential features in the matrix Vec ,

in (8), the autoencoder [26] is utilized to reduce the dimension of Vec . Finally, Vec is submitted to weighted relaxed support vector machines described in the next section.

3. Weighted Relaxed Support Vector Machines

The widely used RSVM [27] is an extension of SVM-L2 with an additional penalty-free slack variable for each sample, which allows influential support vectors to be relaxed, such that a restricted amount of penalty-free slack is used to relax support vectors and push them towards their respective classes. In the paper, based on WRSVM [28], we modify

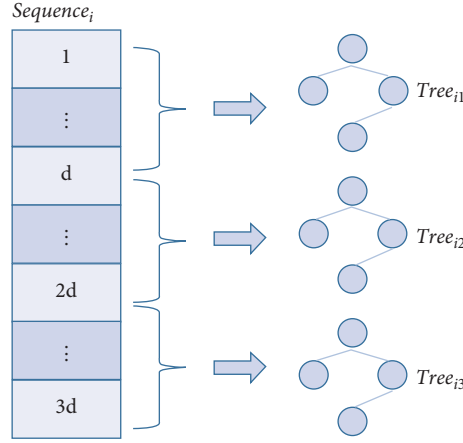


FIGURE 5: The suffix tree for substrings.

RSVM by assigning different weights and free slack amounts to the two classes, normal and abnormal call sequences, which are scaled by the positive and negative class sizes with different penalty control factors, C_1 and C_2 .

The enhanced weighted relaxed support vector machines (EWR-SVM) model is given in equation (10), which is an

extension of the well-known SVM formulation RSVM. However, the constructed model is different from RSVM in such a way that, it differentiates between positive and negative classes, and considers different weights inversely proportional to the class sizes.

$$\min_{\omega, b, \xi, v} \frac{1}{2} \langle \omega \bullet \omega \rangle + \frac{C_1}{2n^+} \sum_{i \in I^+} \xi_i^2 - \frac{C_2}{2n^-} \sum_{i \in I^-} \xi_i^2. \quad (9)$$

$$\begin{aligned} \text{s.t. } & \langle \omega \bullet \omega \rangle + b \geq 1 - \xi_i - v_i, \forall i \in I^+ - \langle \omega \bullet \omega \rangle - b \geq 1 - \xi_i - v_i, \forall i \in I^-, \\ & \sum_{i \in I^+} v_i \leq n^+ \gamma, \\ & \sum_{i \in I^-} v_i \leq n^- \gamma, \\ & v_i \geq 0, \end{aligned} \quad (10)$$

where n^+ and n^- are the sizes of the majority (normal) and minority (abnormal) class, I^+ and I^- , respectively. Free slack is denoted by the variable v_i in the constraints for the i^{th} sample. Due to imbalance, we provide separate amounts of total free slack for the normal and the abnormal classes in the constraints and v_i is parameterized by γ , which is the free slack provided per sample.

Then calculate the first order partial derivative of the Lagrangian function of equation (10) with respect to the related Lagrangian multipliers, we can get the Wolfe dual of equations (10) and (12) can be efficiently solved using the sequential minimal optimization (SMO) algorithm, and the dot products $\langle x_i \cdot x_j \rangle$ in equation (12) can be replaced by a kernel $k(x_i, x_j)$ for nonlinear classification.

$$\min_{\alpha, \beta^+, \beta^-} \sum_{i \in I} \alpha_i - \frac{1}{2} \sum_{i \in I} \sum_{j \in I} y_i y_j \alpha_i \alpha_j \langle x_i \cdot x_j \rangle - \frac{n^+}{2C_1} \sum_{i \in I^+} \alpha_i^2 - \frac{n^-}{2C_2} \sum_{i \in I^-} \alpha_i^2 - n^+ \gamma \beta^+ - n^- \gamma \beta^-, \quad (11)$$

$$\begin{aligned} \text{s.t. } & - \sum_{i \in I^+} \alpha_i + \sum_{i \in I^-} \alpha_i = 0, \\ & 0 \leq \alpha_i \leq \beta^+ \quad \forall i \in I^+, \\ & 0 \leq \alpha_i \leq \beta^- \quad \forall i \in I^-. \end{aligned} \quad (12)$$

4. Experiment

4.1. Datasets Description. In this section, a group of comparisons is carried out to test the performance of the proposed method based on two public system call sequence datasets ADFA-LD and UNM. The ADFA-LD [29] was released by the Australian defense force academy. It contains thousands of system call traces collected from modern Linux local servers. The UNM [30] was released by the University of New Mexico, which contains system call sequences of normal applications in the Linux system and sequences from attacking progress. The detailed descriptions of the two datasets are shown in Tables 1–3. And we can see that the UNM is an obviously imbalanced dataset.

Firstly, the sequence features are extracted by counting the number of the appearance of the fixed-length and variable-length subsequences in each system call sequence, and the abstracted features are verified by mathematical tests including information gain and Mann–Whitney U [31] test in the attachment. And then the features are input into classification models including the proposed EWR-SVM, naive Bayes, logistic regression, random forest, and gradient descent, and the performance is also compared with some traditional methods [8, 10, 32].

The results are evaluated by the index of AUC, F_1 -Score, false alarm rate, and ROC, which is also known as the receptivity curve. The ROC curve takes both the false positive rate and the true positive rate into consideration. It represents a curve drawn by the subjects under specific stimulus conditions due to different results obtained in different conditions. And AUC is the area under the ROC curve.

$$F_1 = \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \quad (13)$$

$$FA = \frac{FP}{FP + TP}.$$

4.2. Experimental Result

4.2.1. Experimental Result. The experimental data set ADFA consists of 833 normal system call sequences and 719 attack system call sequences. The experimental data set UNM consists of 6 normal sample files and 16 attack sample files. The system call sequences' features are extracted from the data set ADFA-LD and UNM datasets. Fixed length features are extracted from system call sequences by a sliding window with length 2. Variable-length features are extracted from system call sequences by a suffix tree. The anomaly detection results based on the classifiers, including EWR-SVM, naive Bayes, logistic regression, random forest, and gradient descent tree are shown in Table 4 and Figures 6 and 7.

Obviously, the performance of EWR-SVM is better than the others on the two datasets. The results demonstrated that the features extracted by our method contain useful information to reflect the behavior patterns behind the call sequences. And the features are useful for model recognition of the abnormal program behaviors. In order to further

TABLE 1: Detail of UNM and ADFA-LD dataset.

Sequence length	ADFA-LD	UNM
Min	75	7
Max	4494	183 018
Mean	462	9031
Deviation	522	25 111

demonstrate the effectiveness of our method, a feature validation is carried out on the datasets ADFA-LD and UNM.

4.2.2. Feature Validation Verification Experiment. The occurrence frequencies of fixed- and variable-length subsequence in ADFA-LD and UNM are shown in Figures 8–15. From the figures, we can see that the distributions of the word (call) frequency are different between the normal and attacking traces (system call sequences). The information gain and Mann–Whitney U test is carried out to demonstrate the effectiveness of the extracted features.

The information gain of both the selected fixed-length sequences and the left ones are shown in Table 5. The table showed that the information gain of the selected fixed-length sequence feature is obviously higher than that of the others. The results demonstrated that fixed-length sequences with the high TF-IDF weight are useful, which can be utilized to extract the behavior features of system call sequences.

According to the statistical results, we found that the frequency of the sub-sequences of normal sequences and attacking sequences does not follow the normal distribution in two datasets, so a nonparametric method is employed to conduct Mann–Whitney U test [31]. The following hypotheses are proposed:

H_0 : there is no significant difference between the sub-sequence frequency of the normal sequence and the sub-sequence frequency of the abnormal sequence

H_1 : opposite hypothesis of H_0

The results of the Mann–Whitney U test is shown in Table 6. According to the law of statistics, we know that when P – value > 0.05 , the original hypothesis H_0 is true, while P – value < 0.05 , the original hypothesis H_0 is rejected, and the hypothesis H_1 is true. From Table 6, we can see that the appearing frequency of fixed-length subsequence of normal sequences in ADFA-LD and UNM datasets is significantly different from that of the abnormal sequences. Similarly, the frequency of variable-length subsequences of normal sequences in ADFA and UNM datasets is also obviously different from that of the abnormal ones.

4.2.3. Comparison with Traditional Methods. In this section, we compared the EWR-SVM with other abnormal traces detection methods [8, 10, 32], and the results are shown in Table 7. From the table, we can see that our method is obviously superior to others. In [8], system call sequences are classified according to the appeared abnormal frequencies in the tested traces. In their method, the optimal window width

TABLE 2: The information for the ADFA-LD dataset.

Data type	Data size	Label
Training	833	Normal
Hydra-FTP	162	Attack
Hydra-SSH	148	Attack
Adduser	91	Attack
Java-meterpreter	125	Attack
Meterpreter	75	Attack
Webshell	118	Attack

TABLE 3: UNM dataset.

Data type	Sample files	Num
Normal	bounce.int, bounce1.int, plus.int, queue.int, sendmail.int, sendmail1.int	6
Code intrusion	decode-280.int, decode-314.int	2
Circular forwarding error	fwd-loops-1.int, fwd-loops-2.int, fwd-loops-3.int, fwd-loops-4.int, fwd-loops-5.int	5
forwarding error	sscp-10763.int, sscp-10814.int, sscp-10801.int	3
syslogd	syslog-local-1.int, syslog-local-2.int, syslog-remote-1.int, syslog-remote-2.int	4
Failed invasion	cert-sm5x-1.int, cert-sm565a-1.int	2

TABLE 4: ADFA-LD and UNM dataset, the results for different models.

Dataset	Algorithm	AUC	F1-score	False alarm rate
ADFA-LD	EWR-SVM	99%	0.93	2.4%
	Naive Bayes	94%	0.90	8%
	Logistic regression	96%	0.94	3%
	Random forest	98%	0.92	7%
	GBDT	98%	0.94	4%
UNM	EWR-SVM	97%	0.83	0%
	Naive Bayes	89%	0.36	0%
	Logistic regression	95%	0.72	10%
	Random forest	97%	0.83	0%
	GBDT	97%	0.8	0%

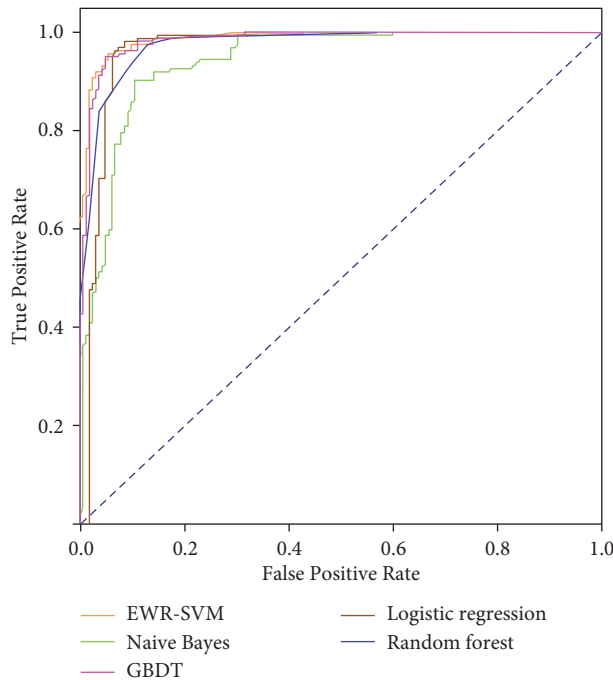


FIGURE 6: ADFA-LD dataset: Area under the ROC curve (AUC) for different models.

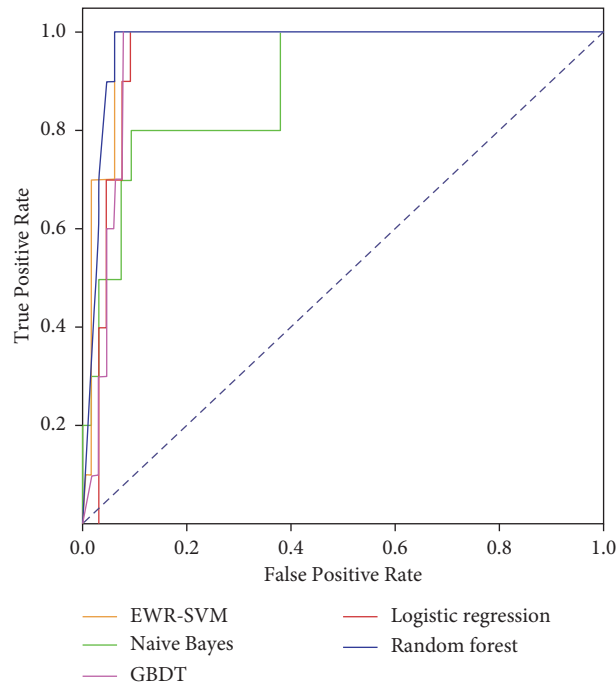


FIGURE 7: UNM dataset: Area under the ROC curve (AUC) for different models.

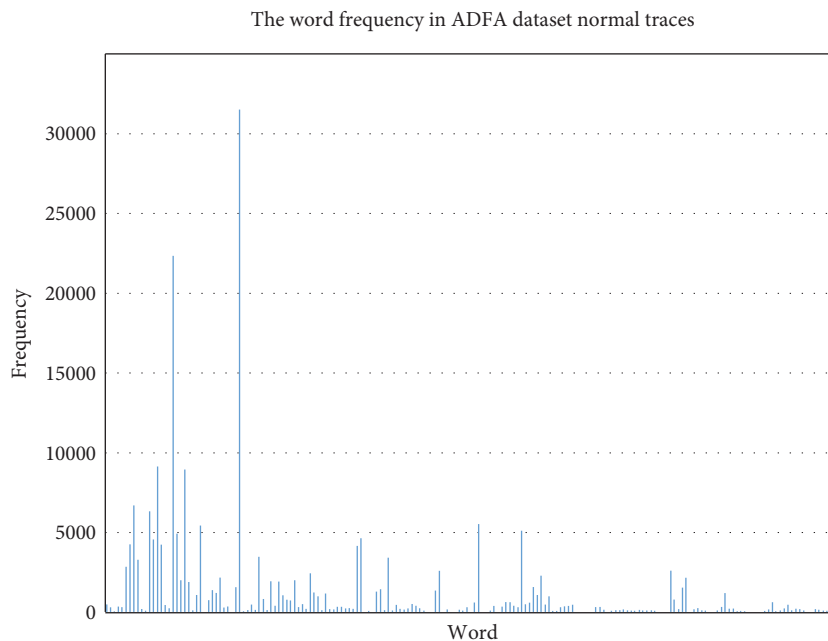


FIGURE 8: The word frequency in ADFA normal traces.

is set by empirical test. However, the optimal length of the subsequence varies among different datasets, resulting in unstable performance. Haider [10] proposed to utilize the rarest repeating subsequence, the most frequently repeated subsequence, along with the maximum and minimum system calls in a single sequence to extract the trace features. However, the correlative contextual semantic information in the sequence is ignored. Anandapriya and Lakshmanan [32] brought the conception of contextual semantic information

into trace detection with fixed-length windows. The results in Table 7 showed that the AUC of the proposed EWR-SVM is much larger than the others, while its false-positive rate is lower than the comparison targets. This is because our method considers not only semantic information of the sequences but also the behavior patterns of processes behind the calls. Furthermore, we proposed to assign different weights and free slack amounts to the two classes (normal and attacking traces), which are scaled by the normal and

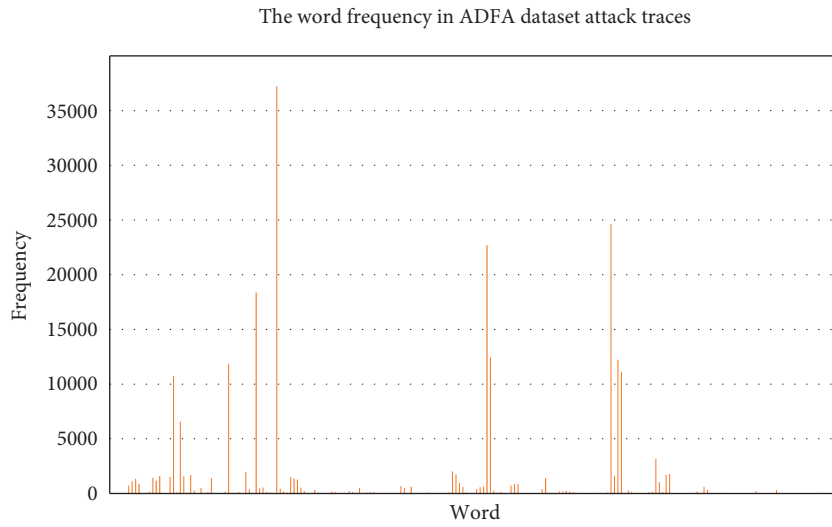


FIGURE 9: The word frequency in ADFA attacking traces.

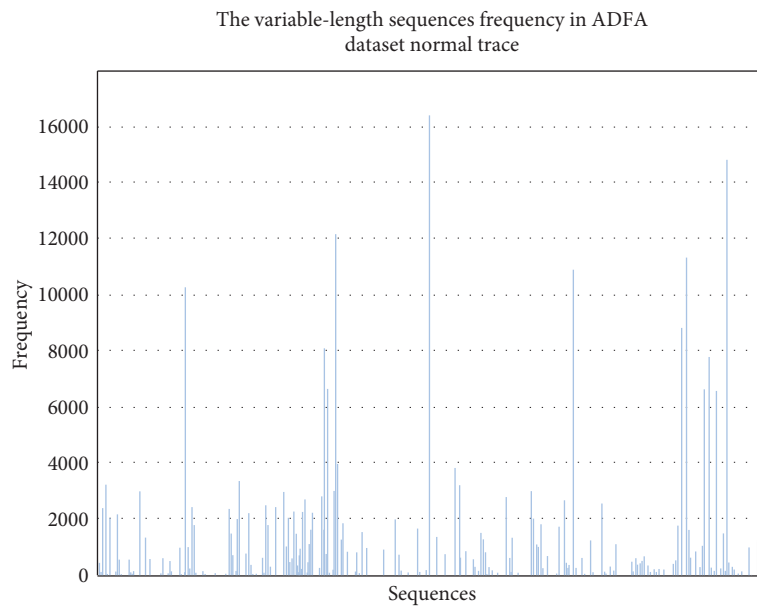


FIGURE 10: The variable-length sequences frequency in ADFA normal traces.

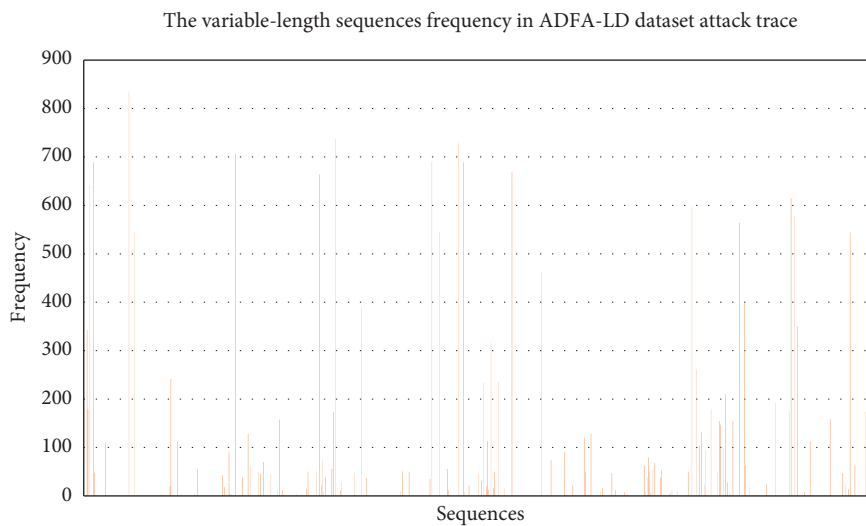


FIGURE 11: The variable-length sequences frequency in ADFA attacking traces.

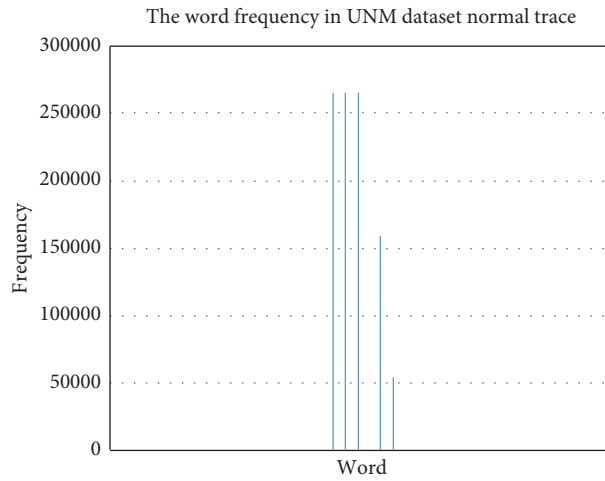


FIGURE 12: The word frequency in UNM normal traces.

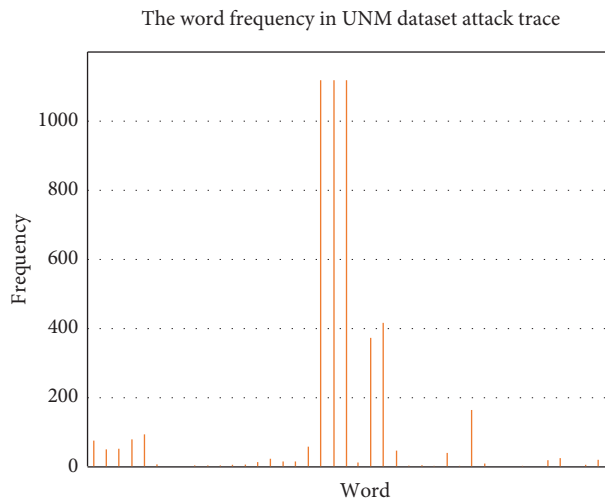


FIGURE 13: The word frequency in UNM attacking traces.

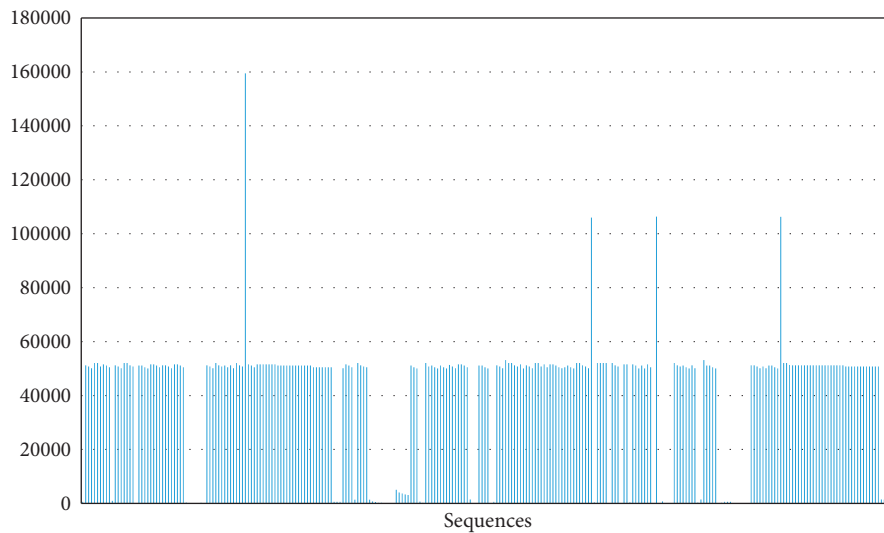


FIGURE 14: The variable-length sequences frequency in UNM normal traces.

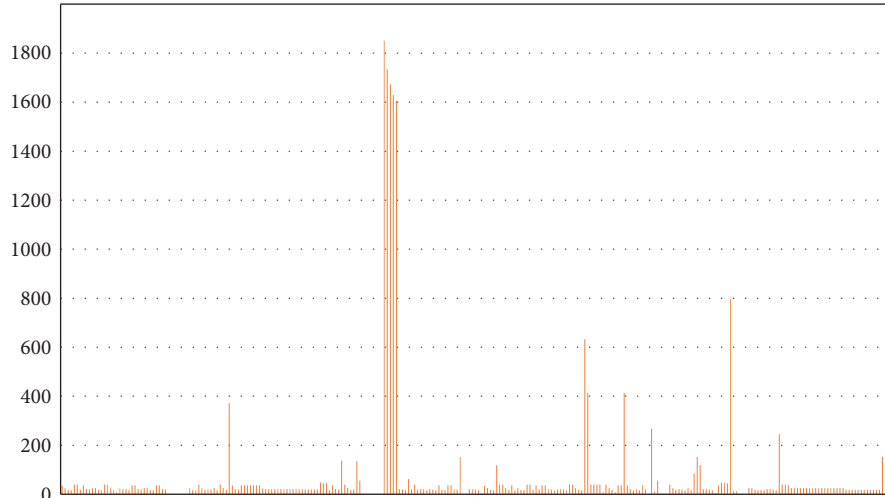


FIGURE 15: The variable-length sequences frequency in UNM attacking traces.

TABLE 5: ADFA-LD and UNM dataset: the information gain results for fixed-length sequences.

Dataset	Sequences type	Average information gain
ADFA-LD	Selected	0.185
	Unselected	0.014
UNM	Selected	0.025
	Unselected	0.0153

TABLE 6: ADFA and UNM datasets the results for Mann-Whitney U .

Dataset	Sequences type	Statistic	P -Value
ADFA-LD	TF-IDF fixed-length sub-sequences	14692.5	3.31×10^{-10}
	Variable-length sub-sequences	144565.5	1.15×10^{-109}
UNM	TF-IDF fixed-length sub-sequences	875.5	4.18×10^{-2}
	Variable-length sub-sequences	2915.5	4.26×10^{-79}

TABLE 7: Comparison results.

Algorithm	AUC	False positive rate
One class SVM [8]	70%	20%
Data feature retrieval and decision engine [10]	60%	23%
Semantic-based method [25]	82%	4.2%
EWR-SVM	99%	2.4%

abnormal class sizes to release the affection of imbalanced data distribution between the two classes. And the slack factors in EWR-SVM make the outliers have less influence on the optimal hyperplane and ensure a large margin between the two classes.

5. Conclusions

In the paper, both fixed-length and variable-length subsequence of system calls are taken into consideration for the

host-based intrusion detection. The semantic weight is incorporated into the selection process of fixed-length sub-sequences. On the other hand, the variable-length subsequence is automatically selected from the suffix tree of each call sequence. In order to deal with the imbalance data distribution and samples outliers of the call sequences, a cost-sensitive relaxed support vector machine EWR-SVM is proposed, in which the restricted penalty-free slack is split independently between the two classes in proportion to the number of samples in each class with different weights. Both

the theoretical analysis and experimental comparison results demonstrated our methods are more suitable for the detection of attacking traces in system call sequences.

Data Availability

Host intrusion detection datasets in this article are public datasets. Web pages are as follows: (1) <http://www.cs.unm.edu/immsec/systemcalls.htm>. (2) <https://research.unsw.edu.au/projects/adfa-ids-datasets>.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (020YFB1805405, and 2019QY0800), and the Natural Science Foundation of China Nos. 61 872 255, U19A2068, and U1736212.

References

- [1] S. Forrest and S. A. Hofmeyr, "Sense of self for unix processes," in *Proceedings of the Security and Privacy*, pp. 120–128, IEEE, Oakland, CA, USA, May 1996.
- [2] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, 1998.
- [3] C. Warrender and S. Forrest, "Detecting intrusions using system calls: alternative data models," in *Proceedings of the Security and Privacy*, pp. 1–20, IEEE, Oakland, CA, USA, May 1999.
- [4] B. Jewell and J. Beaver, "Host-based data exfiltration detection via system call sequences," in *Proceedings of the Conf. Inform. Warf. Secur.*, pp. 134–137, Columbia University, New York, NY, USA, January 2011.
- [5] P. Helman and J. Bhangoo, "A statistically based system for prioritizing information exploration under uncertainty," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 27, no. 4, pp. 449–466, 1997.
- [6] W. Lee and S. J. Stolfo, "Learning patterns from unix process execution traces for intrusion detection," in *Proceedings of the AAAI Work AI Approaches to Fraud Detect Risk Manag*, pp. 50–56, Columbia University, New York, NY, May 1997.
- [7] M. Xie and J. Hu, "Evaluating host-based anomaly detection systems: a preliminary analysis of adfa-ld," in *Proceedings of the Int. Congr. Image Signal Process.*, pp. 1711–1716, IEEE, Hangzhou, China, December 2013.
- [8] M. Xie, J. Hu, and J. Slay, "Evaluating host-based anomaly detection systems: application of the one-class svm algorithm to adfa-ld," in *Proceedings of the Int. Conf. Fuzzy Syst. Knowl. Discovery*, pp. 978–982, IEEE, Xiamen, China, August 2014.
- [9] M. Xie, J. Hu, and E. Chang, "Evaluating host-based anomaly detection systems: application of the frequency-based algorithms to adfa-ld," *Network and System Security*, vol. 8792, pp. 542–549, 2014.
- [10] W. Haider, J. Hu, and M. Xie, "Towards reliable data feature retrieval and decision engine in host-based anomaly detection systems," in *Proceedings of the 10th IEEE Conf. Ind. Electron. Appl.*, pp. 513–517, IEEE, Auckland, New Zealand, June 2015.
- [11] X. Kan, Y. Fan, Z. Fang et al., "A novel iot network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network," *Information Sciences*, vol. 568, pp. 147–162, 2021.
- [12] H. Bian, T. Bai, M. A. Salahuddin, N. Limam, A. A. Daya, and R. Boutaba, "Uncovering lateral movement using authentication logs," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 1049–1063, 2021.
- [13] Y. Shin and K. Kim, "Comparison of anomaly detection accuracy of host-based intrusion detection systems based on different machine learning algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 2, 2020.
- [14] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguous and discontinuous system call patterns," *IEEE Transactions on Computers*, vol. 63, no. 4, pp. 807–819, 2014.
- [15] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & Security*, vol. 21, no. 5, pp. 439–448, 2002.
- [16] Z. Zhang and H. Shen, "Application of online-training svms for real-time intrusion detection with different considerations," *Computer Communications*, vol. 28, no. 12, pp. 1428–1442, 2005.
- [17] P.-F. Marteau, "Sequence covering for efficient host-based intrusion detection," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 994–1006, 2019.
- [18] M. A. Ambusaidi, X. He, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.
- [19] E. A. Shams, A. Rizaner, and A. H. Ulusoy, "A novel context-aware feature extraction method for convolutional neural network-based intrusion detection systems," *Neural Computing & Applications*, vol. 33, no. 20, pp. 13647–13665, 2021.
- [20] B. Subba, "A tfidfvectorizer and singular value decomposition based host intrusion detection system framework for detecting anomalous system processes," *Computers & Security*, vol. 100, p. 102084, 2020.
- [21] A. Laszka, W. Abbas, and S. Shankar, "Optimal thresholds for intrusion detection systems," in *Proceedings of the Symp. Bootcamp Sci. Secur.*, pp. 72–81, ACM, New York, NY, USA, April 2016.
- [22] J. R. Ullmann, "A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words," *The Computer Journal*, vol. 20, no. 2, pp. 141–147, 1977.
- [23] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.
- [24] A. V. Aho and M. J. Corasick, "Efficient string matching," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.
- [25] M. Crochemore and W. Rytter, "On-line construction of suffix trees," *Jewels Of Stringology: Text Algorithms*, vol. 14, 2015.
- [26] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [27] O. Şeref, W. A. Chaovalitwongse, and J. P. Brooks, "Relaxing support vectors for classification," *Annals of Operations Research*, vol. 216, no. 1, pp. 229–255, 2014.
- [28] O. Şeref, T. Razzaghi, and P. Xanthopoulos, "Weighted relaxed support vector machines," *Annals of Operations Research*, vol. 249, no. 1-2, pp. 1–37, 2017.

- [29] G. Creech and J. Hu, "Generation of a new ids test dataset: time to retire the kdd collection," in *Proceedings of the Wireless Communications and Networking Conference (WCNC), 2013*, pp. 4487–4492, IEEE, Shanghai, China, April 2013.
- [30] Computer Science Department, "University of New Mexico intrusion detection dataset oct 2020," 2020, <http://www.cs.unm.edu/immsec/systemcalls.htm>.
- [31] J. Whitney, "Testing for differences with the nonparametric mann-whitney u test," *The Journal of Wound, Ostomy and Continence Nursing: Official Publication of The Wound, Ostomy and Continence Nurses Society*, vol. 24, no. 1, p. 12, 1997.
- [32] M. Anandapriya and B. Lakshmanan, "Anomaly based host intrusion detection system using semantic based system call patterns," in *Proceedings of the IEEE International Conference on Intelligent Systems & Control*, pp. 1–5, Coimbatore, India, January 2015.

Research Article

Black-Box Adversarial Attacks against Audio Forensics Models

Yi Jiang  and Dengpan Ye 

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education,
School of Cyber Science and Engineering, Wuhan University, Wuhan, China

Correspondence should be addressed to Dengpan Ye; yedp@whu.edu.cn

Received 16 September 2021; Revised 28 December 2021; Accepted 30 December 2021; Published 17 January 2022

Academic Editor: Ding Wang

Copyright © 2022 Yi Jiang and Dengpan Ye. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Speech synthesis technology has made great progress in recent years and is widely used in the Internet of things, but it also brings the risk of being abused by criminals. Therefore, a series of researches on audio forensics models have arisen to reduce or eliminate these negative effects. In this paper, we propose a black-box adversarial attack method that only relies on output scores of audio forensics models. To improve the transferability of adversarial attacks, we utilize the ensemble-model method. A defense method is also designed against our proposed attack method under the view of the huge threat of adversarial examples to audio forensics models. Our experimental results on 4 forensics models trained on the LA part of the ASVspoof 2019 dataset show that our attacks can get a 99% attack success rate on score-only black-box models, which is competitive to the best of white-box attacks, and 60% attack success rate on decision-only black-box models. Finally, our defense method reduces the attack success rate to 16% and guarantees 98% detection accuracy of forensics models.

1. Introduction

Speech synthesis technologies have advanced significantly in recent years [1, 2]. Speech synthesis generally refers to the process of converting text into speech. At present, the mainstream speech synthesis system generally consists of two parts: spectrogram prediction network and vocoder. The spectrogram prediction network converts the text into the mel spectrograms. Shen et al. [3], for example, use a Seq2Seq network with an attention mechanism to map text to mel spectrograms, Ren et al. [4] and Lancucki et al. [5] use the transformer structure [6] for this purpose. The vocoder is used to convert the mel spectrograms into speech. Van et al. [7] use several dilated convolution layers to achieve this function. Prenger et al. [8] use a generative model that generates audio by sampling from a distribution [9]. Of course, there are some end-to-end models, such as FastSpeech2s [10]. These technologies have been widely utilized in the Internet of things [11, 12] like a smart speaker, personal voice assistant, etc.

However, these technologies also have been abused. They appear in telecom fraud, creating rumors and spoofing

automatic speaker verification (ASV) systems. To detect these fake audios, the researchers designed several methods. Lai et al. [13] accumulate discriminative features in frequency and time domains selectively, Lai et al. [14] adaptively recalibrate channel-wise feature responses by explicitly modeling interdependencies between channels, Jung et al. [15] use the convolutional layer to extract frame-level embedding and the GRU layer to aggregate extracted frame-level features into a single utterance-level feature. Related competitions [16] were also organized to promote research in this field.

Previous researches show that the image classification neural networks [17–19] are vulnerable to attacks from adversarial examples, and audio models are no exception [20–27]. Generally, adversarial attacks are divided into two categories: white-box attacks and black-box attacks. A white-box attack means that the attacker can access the complete structure, parameters, and input and output of the model, the black-box attack means that the attacker can only obtain external input and output information but cannot access the internal structure and parameters of the model [28, 29]. Current researches on adversarial attacks on audio forensics

models mainly focus on white-box attacks [30]. Although there are studies on using the transferability of adversarial examples to achieve black-box attacks, it still relies on white-box models to generate adversarial examples [31]. In this paper, we will only rely on the output distribution to conduct black-box adversarial attacks.

The main contributions of this article can be summarized as follows:

- (i) To the best of our knowledge, we are the first to propose a black-box adversarial attack method only relying on the output distribution of audio forensics models and we use the ensemble-model method to increase the transferability of adversarial examples to implement decision-only black-box attacks.
- (ii) We propose a defense method based on low-sensitivity features in view of the huge threat of black-box adversarial examples.
- (iii) Our proposed black-box method can get the attack success rate equivalent to the best of white-box attacks and our defense method significantly reduces the threat.

The rest of the paper is organized as follows: Section 2 introduces several audio forensics models, which are the victim models in this paper; Section 3 describes the proposed adversarial attacks and defense methods; Section 4 introduces the experimental setup and results; and Section 5 gives the conclusion and future work.

2. Audio Forensics Models

Current speech synthesis technologies have developed to a high level. Once they are used by criminals in the fields of telecommunications fraud, network rumors, etc., and it will bring great harm to society. Therefore, people have designed a variety of audio forensics models, which aim to reveal the difference between real voice and fake voice from various angles. The following will introduce several current mainstream audio forensics models, whose detection accuracy is among the best in the audio forensics competition ASVspoof 2019. Therefore, they will serve as the victim models in this paper.

2.1. Attentive Filtering Network Model (AFnet) [13]. Attentive filtering (AF) accumulates discriminative features in frequency and time domains selectively. AF augments every input feature map S with an attention heatmap A_s . The augmented feature map S^* is then treated as the new input for the dilated residual network (DRN). For S^* , $S^* \in \mathbb{R}^{(F \times T)}$, AF is described as

$$S^* = A_s \circ S + \bar{S}, \quad (1)$$

where F and T are the frequency and time dimensions, \circ is the element-wise multiplication operator, $+$ is the element-wise addition operator, and \bar{S} is the residual S . To learn the attention heatmap, A_s contains similar bottom-up and top-down processing as [32, 33], and is described as

$$A_s = \phi(U(S)), \quad (2)$$

where ϕ is a nonlinear transform such as sigmoid or softmax, U is a U -net-like structure, composed of a series of downsampling and upsampling operations, and S is the input.

2.2. Squeeze-Excitation ResNet Model (SEnet) [14]. The squeeze-and-excitation (SE) block [34] is a computational unit that can be constructed for any given transformation $F_{tr}: X \rightarrow U, X \in \mathbb{R}^{H' \times W' \times C'}, U \in \mathbb{R}^{H \times W \times C}$. The features U are first passed through a squeeze operation F_{sq} and get a statistic $z \in \mathbb{R}^C$, where the c^{th} element of z is calculated by

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W u_c(i, j). \quad (3)$$

This is followed by an excitation operation F_{ex} .

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z)), \quad (4)$$

where δ refers to the ReLU function, $W_1 \in \mathbb{R}^{C/r \times C}$ and $W_2 \in \mathbb{R}^{C \times C/r}$. The final output of the block is obtained by rescaling the transformation output U with the activations

$$\tilde{x}_c = F_{scale}(u_c, s_c) = s_c \cdot u_c, \quad (5)$$

where $\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_C]$ and $F_{scale}(u_c, s_c)$ refers to channel-wise multiplication between the feature map $u_c \in \mathbb{R}^{H \times W}$.

It will be easy to apply the SE block, which adaptively recalibrates channel-wise feature responses by explicitly modeling interdependencies between channels to ResNet and get the squeeze-excitation ResNet (SEnet).

2.3. CNN-GRU Model [15]. The DNNs used in this model include convolutional neural network (CNN), gated circulation unit (GRU), and fully connected layer (CNN-GRU). In this architecture, the convolutional layer is first used to process input features to extract frame-level embedding. The convolutional layer includes residual blocks with identity mapping [35] to facilitate the training of deep architectures. Specifically, the first convolution layer of this model deals with the local adjacent time and frequency domains and gradually aggregates them through the repeated pooling operations to extract frame-level embedding. Then, the GRU layer is used to aggregate the extracted frame-level features into a single utterance-level feature. Fully connected layers are used to convert utterance-level features. An output layer with two nodes indicates whether the input utterance is a spoof or bona fide.

3. Audio Adversarial Examples Generation

3.1. Threat Model. In this paper, the adversarial attack is to craft adversarial voice $x = x + \delta$ by finding a perturbation δ such that (1) x is an original voice classified as the spoof by the audio forensics model, (2) δ is as human-imperceptible as possible, and (3) the audio forensics model classifies the voice $x = x + \delta$ as the bonafide. To be as human-

imperceptible as possible, our attack following the FAKE-BOB [22] adopts L_∞ norm to measure the similarity between the original and adversarial voices and ensures that the L_∞ distance $\|\dot{x}, x\|_\infty = \max_i \{|\dot{x}(i) - x(i)|\}$ is less than the given maximal amplitude threshold ϵ of the perturbation, where i denotes the sample point of the audio waveform. So, we can formalize the problem of finding an adversarial voice \dot{x} for a voice x as the following constrained minimization problem:

$$\operatorname{argmin}_\delta f(x + \delta), \text{ such that } \|\dot{x}, x\|_\infty < \epsilon, \quad (6)$$

where f is a loss function. To ensure the success rate of the attack, we minimize the loss function rather than minimizing the perturbation δ . When f is minimized, $x + \delta$ is recognized as the bonafide.

According to the attacker's mastery of the model, the adversarial attack can be divided into a white-box attack and a black-box attack. The white-box attack generally means that the attacker can fully understand all the information of the victim model, including the external input and output information of the model and the internal structure and parameters. Attackers can efficiently perform gradient descent by differentiating the loss function to launch an iteration-based adversarial attack. Previous researches on adversarial examples against audio forensics models mostly focus on white-box attacks [30] or using the adversarial examples generated from white-box models to conduct transferable adversarial attacks [31]. However, in the real environment, users of the audio forensics model generally do not disclose the internal structure and parameters of the model, which significantly limits the application scenarios and the threat of white-box attack. It also leads some people to mistakenly believe that protecting the internal information of the model can prevent them from adversarial attacks.

Therefore, in this paper, we will focus on black-box adversarial attacks. Black-box adversarial attacks mean that the attackers can only access the input and external output of the model. The attacker cannot directly use the internal information to obtain the gradient of the loss function and launch the adversarial attack. Compared to the white-box model, black-box adversarial attacks can be further subdivided into score-only black-box adversarial attacks and decision-only black-box adversarial attacks. The score-only black-box adversarial attack refers to that the attacker can access the confidence scores of the model for each input, while a decision-only black-box attack means a direct attack that solely relies on the final decision of the model [36].

In the remainder of this section, we will present methods for launching adversarial attacks in these two black-box scenarios and attempt to defend against score-only black-box adversarial attacks.

3.2. Score-Only Black-Box Attack Algorithm. As shown in Figure 1, we will introduce the whole attack process in the remainder of this subsection, especially the loss function and algorithm to solve the optimization problem.

3.2.1. Loss Function. The key to carrying out the adversarial attack is that the score $[S(x)]_b$ of the bonafide voice should be greater than $[S(x)]_s$ of the spoof voice. Therefore, the loss function f is defined as follows:

$$f(x) = \max\{[S(x)]_s - [S(x)]_b, -\kappa\}, \quad (7)$$

where the parameter $-\kappa$, is to control the intensity of adversarial examples, so we can increase κ to enhance the robustness of the adversarial examples.

3.2.2. Optimization Algorithm. We use the basic iterative method (BIM) [18] with the estimated gradients to craft adversarial examples. Therefore, the i^{th} iteration voice x_i can be defined as

$$\dot{x}_i = \operatorname{clip}_{x,\epsilon} \left\{ \dot{x}_{i-1} - \eta \cdot \operatorname{sign}(g_i) \right\}, \quad (8)$$

where η is the learning rate, g_i is the i^{th} iteration gradient.

To compute the estimated gradients, we use the natural evolution strategy (NES) [37], because the NES-based gradient estimation is proved to require much fewer queries than finite difference gradient estimation. In detail, we first create m (must be even) Gaussian noises (u_1, \dots, u_m) on the i^{th} iteration, and generate m new voices x_{i-1}, \dots, x_{i-1} , where $x_{i-1} = x_{i-1} + \sigma \times u_j$. Then we compute the loss values $f(x_{i-1}), \dots, f(x_{i-1})$. Finally, the gradient $\nabla_x f(x_{i-1})$ can be computed as

$$\nabla_x f(\dot{x}_{i-1}) = \frac{1}{m \times \sigma} \sum_{j=1}^m f(\dot{x}_{i-1}^j) \times u_j. \quad (9)$$

We also use the momentum [38] to speed up the convergence and increase the transferability of adversarial examples, therefore the i^{th} iteration gradient g_i can be defined as

$$g_i = \mu \cdot g_{i-1} + (1 - \mu) \cdot \nabla_x f(\dot{x}_{i-1}), \quad (10)$$

where the μ is the decay factor.

3.3. Decision-Only Black-Box Attack Algorithm. Although the NES-based gradient estimation attack has no need to touch the internal structure and parameters of the model, it still needs to obtain the distribution of result scores through a large number of queries. Once the model limits the number of queries or returns only positive or negative results without the score, this attack will be impossible to implement. In this regard, the transferable adversarial attack method can be used to achieve a decision-only black box attack. Specifically, the transferable adversarial attack is generating adversarial examples through known methods and then using these examples to attack the decision-only black-box model.

To improve the transferability of adversarial examples, an obvious idea is to increase the attack intensity κ . However, if we only increase the κ , the adversarial examples may overfit and it will decrease the transferability. So, an ensemble-based method will be used to conduct the decision-only black-box attack. In [39], authors argue that the

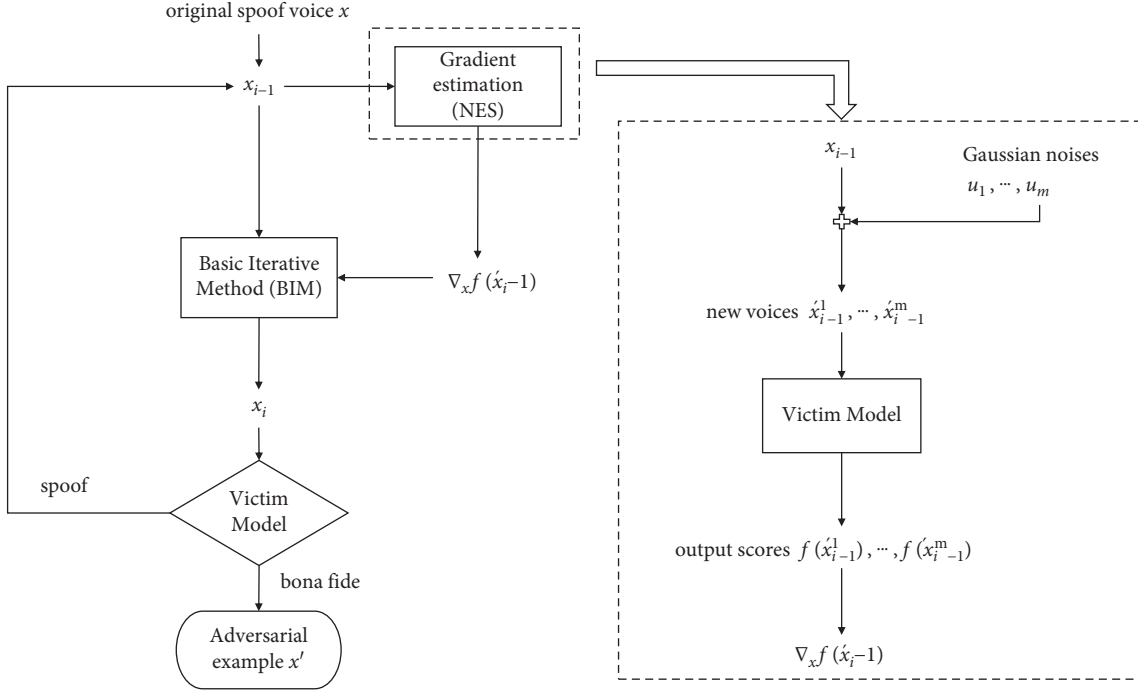


FIGURE 1: Score-only black-box attack process.

adversarial examples are more likely to transfer to other models if they could fool various models simultaneously, as shown in Figure 2. We follow this strategy and made a weighted sum of the scores of multiple models. To attack K score-only black-box models simultaneously, we fuse the loss function as

$$f(x) = \sum_{k=1}^K w_k \cdot f(x_k), \quad (11)$$

where $f(x_k)$ represents the loss of k_{th} score-only black-box model and w_k is the ensemble weight, where $\sum_{k=1}^K w_k = 1$.

3.4. Defense. Audio forensics models aim to reduce the harm of speech synthesis technology; however, several adversarial attacks have made these efforts in vain. So we need some methods to defend against this adversarial attack so that the model can be reinforced.

In previous experiments, we noticed that although the models have the same structure and are trained on the same dataset, they show different detection accuracy when trained by features of different sizes and types. We deem audio forensics models have different sensitivity to different features. Because models trained by high-sensitivity features show better performance than those trained by low-sensitivity features, we consider the reason is the model can obtain more information from original audio information through particular features. Therefore, we think that if we use the low-sensitivity features, the models will suffer less impact from the adversarial perturbation, and we will attempt to use these low-sensitivity features to reinforce audio forensics models.

4. Experiments

4.1. Dataset and Victim Models. Following the setting in [30], we use the LA part of the ASVspoof 2019 dataset [16]. We use the 2048 fast Fourier transform (FFT) bins energy spectrum as input for all models. Only the first 400 frames of each utterance are used to extract acoustic features.

We use the LA training set to train our audio forensics models and the LA developing set to evaluate the models. The details of the models can be found in Table 1.

4.2. Score-Only Black-Box Attack. We randomly selected 500 spoof audio examples from the trn set to conduct our adversarial proposed attacks. All the selected samples are classified correctly by our victim audio forensics models before the attacks. We only generate adversarial examples from spoof examples, because we consider there's no real value in converting a bonafide sample to a spoof one. All of the attacks are conducted under $\varepsilon = 0.001$ in equation (6), $\kappa = 0$ in equation (7), $m = 500$, $\sigma = 0.001$ in equation (9). As shown in Table 2, our proposed method gets a 99% attack success rate, which is comparable to the MI-FGSM, the most powerful white-box attack method.

We can conclude that our proposed score-only black-box attack method is extremely threatening to mainstream audio forensics models. Tiny adversarial perturbation can almost completely invalidate them. This also shows that if only the internal structure and parameters are hidden from the attackers, it is almost impossible to defend against the attack. It is necessary to find other more effective defense methods.

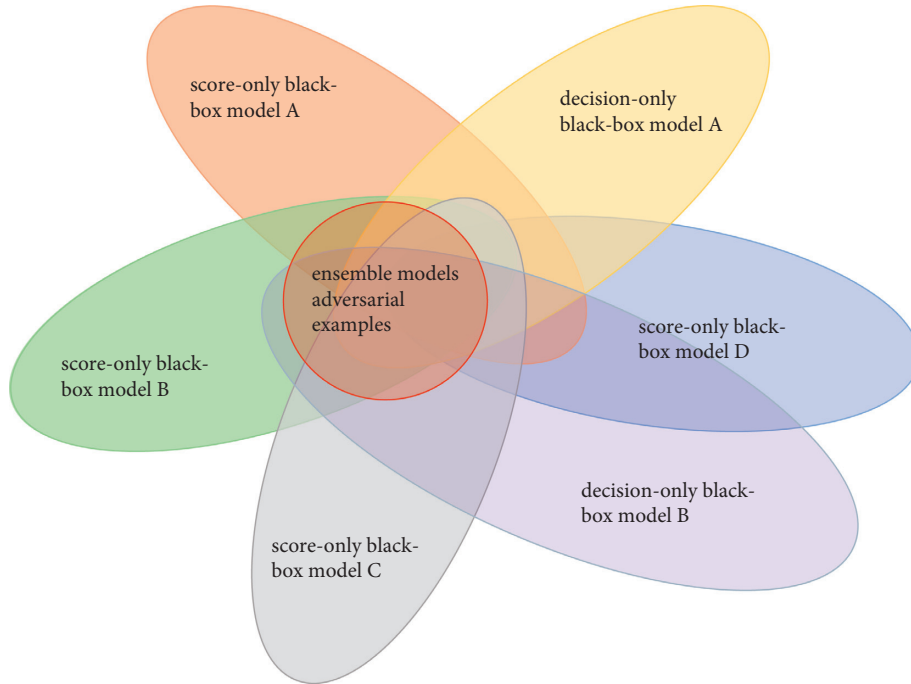


FIGURE 2: Ensemble-model.

TABLE 1: Detection accuracy of victim models (%).

Victim models	Trn	Dev
ResNet	99.99	99.90
SEnet	99.82	99.85
AFnet	99.74	99.57
CNN-GRU	99.99	99.96

TABLE 2: Adversarial attack success rate on score-only black-box models (%).

Victim models	Proposed method
ResNet	99.6
SEnet	99.9
AFnet	98.9
CNN-GRU	97.9

4.3. Decision-Only Black-Box Attack. We use the 100 spoof audio examples used in the previous subsection to conduct the decision-only black-box attack. We use 3 of the models to generate the adversarial examples and use the remaining one to evaluate the adversarial examples. In order to evaluate the ensemble-model method and the effect of intensity factor κ , we also generate adversarial examples through single-model with $\kappa = 2$ and multi-models with $\kappa = 0$. All of the results can be found in Table 3 and Figure 3.

We find that if we only increase the intensity factor κ or use the ensemble-model method, the improvement of the transferability of the adversarial examples is limited. So we need to combine these methods to get the best attack effect.

4.4. Defense. In the previous part of the paper, We have discussed how to enhance the defense capabilities of the model against our proposed adversarial attack. Here, we will

test the method using low-sensitivity features. After conducting a series of experiments, we found that the models, which are trained by the log-power spectrum of 512 FFT bins, get a balance between the accuracy of detecting spoof samples and the defense capabilities against adversarial attacks.

We used the LA training set to train the audio forensics models and the LA developing set to evaluate the models. The detection accuracy of the original models and the reinforced models are shown in Table 4.

We randomly select 100 spoof audio examples from the trn set to conduct the adversarial attack and evaluate the defense capabilities of the reinforced models, we also conduct the attack on original models. The results of the examples can be seen in Table 5 and Figure 4.

By comparing the two types of models, we find that the average detection accuracy of original models on original examples is slightly higher than that of the reinforced

TABLE 3: Adversarial attack success rate on decision-only black-box models (%).

Victim models	ResNet, $\kappa = 2$	SEnet, $\kappa = 2$	AFnet, $\kappa = 2$	CNN-GRU, $\kappa = 2$	Muti-models, $\kappa = 0$	Muti-models, $\kappa = 2$
ResNet	*	46	54	46	54	60
SEnet	40	*	38	42	53	62
AFnet	56	45	*	45	65	78
CNN-GRU	47	44	45	*	48	50

The bold values are the best results from our proposed method.

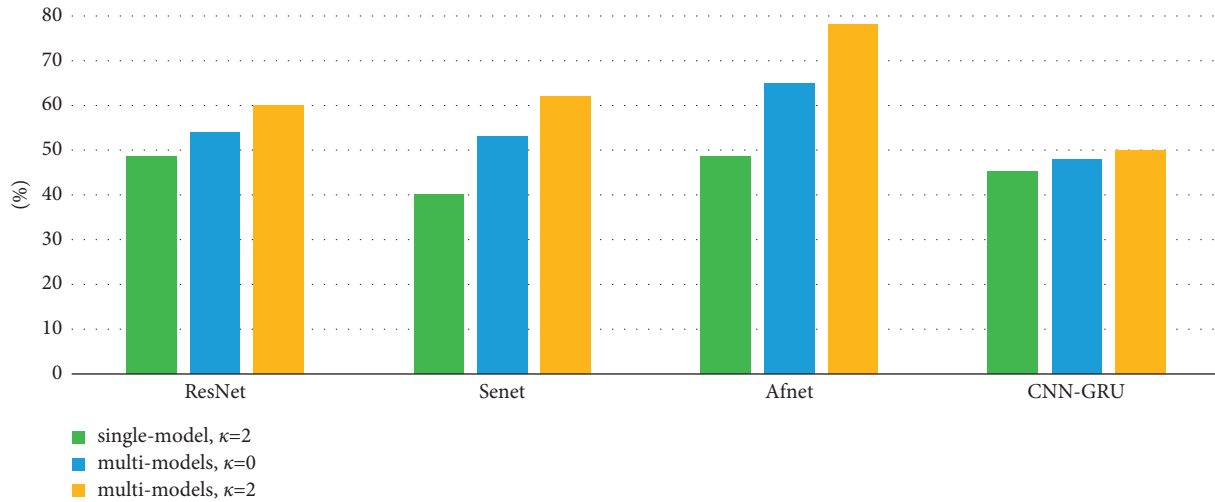


FIGURE 3: Adversarial attack success rate on decision-only black-box models.

TABLE 4: Detection accuracy of original models and reinforced models on original examples (%).

Victim models	Original models	Reinforced models
ResNet	99.90	94.12
SEnet	99.85	99.57
AFnet	99.57	99.38
CNN-GRU	99.96	99.27
Average	99.82	98.09

TABLE 5: Score-only black-box attack success rate on original models and reinforced models.

Victim models	Original models	Reinforced models
ResNet	99	1
SEnet	100	28
AFnet	98	0
CNN-GRU	97	36
Average	98.5	16.25

The bold values are the best results from our proposed method.

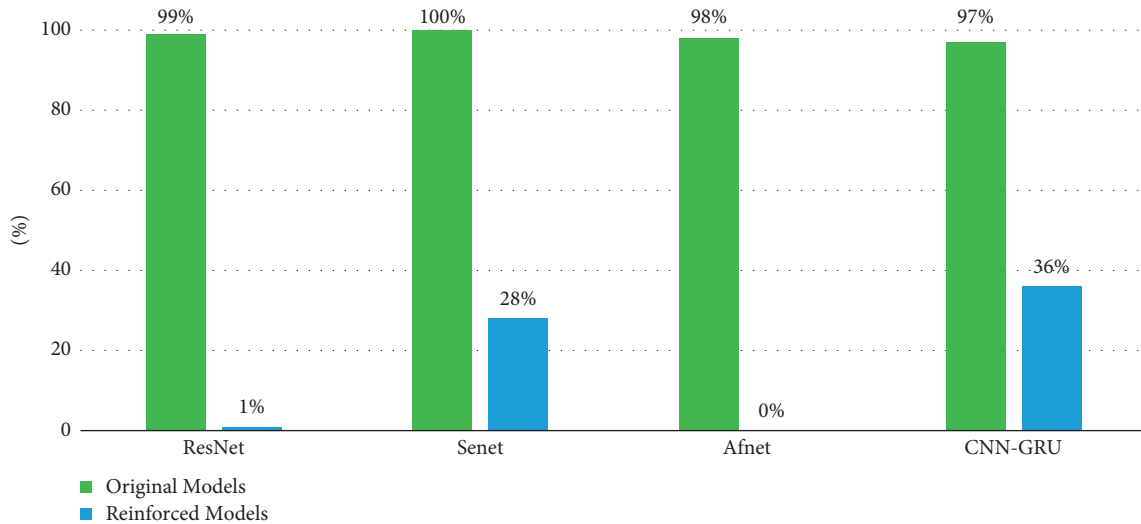


FIGURE 4: Score-only black-box attack success rate on original models and reinforced models.

models. However, the reinforced models we proposed significantly reduce the success rate of adversarial attacks.

5. Conclusion

In this paper, the black-box attack method we proposed achieves an attack success rate equivalent to the best of white-box attacks, which shows that hiding the internal structure and parameters of the model from the attacker cannot effectively protect the model. The success rate of the decision-only black-box attack also shows that the method of limiting the number of queries has scant protection capabilities for the model. Therefore, it is necessary to do more research on exploring more effective methods of model reinforcement.

Although the method proposed in this paper has reached a similar success rate to that of the white-box attack, however, there is still a large gap between the black-box method and the white-box method in terms of the generation efficiency of adversarial examples. Therefore, further research is needed on improving the generation efficiency of black-box adversarial examples.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] Y. Yan, X. Tan, B. Li et al., "Adaspeech 3: adaptive text to speech for spontaneous style," 2021, <https://arxiv.org/abs/2107.02530>.
- [2] I. Elias, H. Zen, J. Shen et al., "Parallel tacotron: non-autoregressive and controllable tts," in *Proceedings of the ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5709–5713, Toronto, Canada, June 2021.
- [3] J. Shen, R. Pang, R. J. Weiss et al., "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4779–4783, IEEE, Calgary, Canada, April 2018.
- [4] Y. Ren, Y. Ruan, X. Tan et al., "Fastspeech: fast, robust and controllable text to speech," 2019, <https://arxiv.org/abs/1905.09263>.
- [5] A. Łańcucki, "Fastpitch: parallel text-to-speech with pitch prediction," 2020, <https://arxiv.org/abs/2006.06873>.
- [6] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.
- [7] A. van den Oord, S. Dieleman, H. Zen et al., "Wavenet: a generative model for raw audio," in *Proceedings of the 9th ISCA Speech Synthesis Workshop*, p. 125, Sunnyvale, CA, USA, September 2016.
- [8] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: a flow-based generative network for speech synthesis," in *Proceedings of the ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621, Brighton, UK, May 2019.
- [9] D. P. Kingma and P. Dhariwal, "Glow: generative flow with invertible 1×1 convolutions," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 10236–10245, Montréal, Canada, December 2018.
- [10] Y. Ren, C. Hu, X. Tan et al., "Fastspeech 2: fast and high-quality end-to-end text to speech," 2020, <https://arxiv.org/abs/2006.04558>.
- [11] Q. Wang, D. Wang, C. Cheng, and D. He, "Quantum2fa: efficient quantum-resistant two-factor authentication scheme for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, no. 1, p. 1, 2021.
- [12] D. Wang and P. Wang, "Two birds with one stone: two-factor authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708–722, 2016.
- [13] C. I. Lai, A. Abad, K. Richmond, J. Yamagishi, N. Dehak, and S. King, "Attentive filtering networks for audio replay attack detection," in *Proceedings of the ICASSP 2019-2019 IEEE*

- International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6316–6320, Brighton, UK, May 2019.
- [14] C. I. Lai, N. Chen, J. Villalba, and N. Dehak, “Assert: anti-spoofing with squeeze-excitation and residual networks,” in *Proceedings of the Interspeech 2019*, pp. 1013–1017, Graz, Austria, September 2019.
- [15] J. w. Jung, H. j. Shim, H. S. Heo, and H. J. Yu, “Replay attack detection with complementary high-resolution information using end-to-end DNN for the ASVSpooF 2019 challenge,” in *Proceedings of the Interspeech 2019*, pp. 1083–1087, Graz, Austria, 2019.
- [16] M. Todisco, X. Wang, V. Vestman et al., “AsvspooF 2019: Future horizons in spoofed and fake audio detection,” 2019, <https://arxiv.org/abs/1904.05441>.
- [17] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” 2015, <https://arxiv.org/abs/1412.6572>.
- [18] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” 2016, <https://arxiv.org/abs/1607.02533>.
- [19] C. Szegedy, W. Zaremba, I. Sutskever et al., “Intriguing properties of neural networks,” <https://arxiv.org/abs/1312.6199>.
- [20] H. Abdullah, W. Garcia, C. Peeters, P. Traynor, K. R. Butler, and J. Wilson, “Practical hidden voice attacks against speech and speaker recognition systems,” 2019, <https://arxiv.org/abs/1904.05734>.
- [21] M. Alzantot, B. Balaji, and M. Srivastava, “Did you hear that? adversarial examples against automatic speech recognition,” 2018, <https://arxiv.org/abs/1801.00554>.
- [22] G. Chen, S. Chen, L. Fan et al., “Who is real bob? adversarial attacks on speaker recognition systems,” 2019, <https://arxiv.org/abs/1911.01840>.
- [23] Y. Chen, X. Yuan, J. Zhang et al., “Devil’s whisper: a general approach for physical adversarial attacks against commercial black-box speech recognition devices,” in *Proceedings of the 29th USENIX Security Symposium Security 20*, pp. 2667–2684, Boston, MA, USA, August 2020.
- [24] T. Du, S. Ji, J. Li, Q. Gu, T. Wang, and R. Beyah, “Sirenattack: generating adversarial audio for end-to-end acoustic systems,” in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pp. 357–369, Taipei, Taiwan, October 2020.
- [25] Q. Wang, B. Zheng, Q. Li, C. Shen, and Z. Ba, “Towards query-efficient adversarial attacks against automatic speech recognition systems,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 896–908, 2020.
- [26] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, “Dolphinattack: inaudible voice commands,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 103–117, Dallas, TX USA, November 2017.
- [27] B. Zheng, P. Jiang, Q. Wang et al., “Black-box adversarial attacks on commercial speech platforms with minimal information,” in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 86–107, New York, NY, USA, November 2021.
- [28] W. Brendel, J. Rauber, and M. Bethge, “Decision-based adversarial attacks: reliable attacks against black-box machine learning models,” in *Proceedings of the International Conference on Learning Representations*, Vancouver, Canada, May 2018.
- [29] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, “Black-box adversarial attacks with limited queries and information,” in *Proceedings of the International Conference on Machine Learning*, pp. 2137–2146, Macau, China, February 2018.
- [30] S. Liu, H. Wu, H. . y. Lee, and H. Meng, “Adversarial attacks on spoofing countermeasures of automatic speaker verification,” in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 312–319, Singapore, December 2019.
- [31] Y. Zhang, Z. Jiang, J. Villalba, and N. Dehak, “Black-box attacks on spoofing countermeasures using transferability of adversarial examples,” in *Proceedings of the Interspeech 2020*, pp. 4238–4242, Shanghai, China, 2020.
- [32] F. Wang, M. Jiang, C. Qian et al., “Residual attention network for image classification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164, Honolulu, HI, USA, July 2017.
- [33] O. Ronneberger, P. Fischer, and T. Brox, “U-net: convolutional networks for biomedical image segmentation,” in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, Munich, Germany, October 2015.
- [34] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, UT, USA, June 2018.
- [35] J. W. Jung, H. S. Heo, I. H. Yang, H. J. Shim, and H. J. Yu, “A complete end-to-end speaker verification system using deep neural networks: from raw signals to verification result,” in *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5349–5353, IEEE, Calgary, Canada, April 2018.
- [36] W. Brendel, J. Rauber, and M. Bethge, “Decision-based adversarial attacks: reliable attacks against black-box machine learning models,” 2017, <https://arxiv.org/abs/1712.04248>.
- [37] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber, “Natural evolution strategies,” in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3381–3387, IEEE, Hongkong, China, June 2008.
- [38] Y. Dong, F. Liao, T. Pang et al., “Boosting adversarial attacks with momentum,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9185–9193, Salt Lake, UT, USA, June 2018.
- [39] Y. Liu, X. Chen, C. Liu, and D. Song, “Delving into transferable adversarial examples and black-box attacks,” 2016, <https://arxiv.org/abs/1611.02770>.

Research Article

Network Embedding-Based Approach for Detecting Collusive Spamming Groups on E-Commerce Platforms

Jinbo Chao ^{1,2}, Chunhui Zhao ^{1,2} and Fuzhi Zhang ^{1,2}

¹School of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei, China

²The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Qinhuangdao, Hebei, China

Correspondence should be addressed to Fuzhi Zhang; xjzfz@ysu.edu.cn

Received 17 September 2021; Accepted 21 December 2021; Published 11 January 2022

Academic Editor: Chunhua Su

Copyright © 2022 Jinbo Chao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Information security is one of the key issues in e-commerce Internet of Things (IoT) platform research. The collusive spamming groups on e-commerce platforms can write a large number of fake reviews over a period of time for the evaluated products, which seriously affect the purchase decision behaviors of consumers and destroy the fair competition environment among merchants. To address this problem, we propose a network embedding based approach to detect collusive spamming groups. First, we use the idea of a meta-graph to construct a heterogeneous information network based on the user review dataset. Second, we exploit the modified DeepWalk algorithm to learn the low-dimensional vector representations of user nodes in the heterogeneous information network and employ the clustering methods to obtain candidate spamming groups. Finally, we leverage an indicator weighting strategy to calculate the spamming score of each candidate group, and the top-k groups with high spamming scores are considered to be the collusive spamming groups. The experimental results on two real-world review datasets show that the overall detection performance of the proposed approach is much better than that of baseline methods.

1. Introduction

With the development of Web 2.0, it has become fashionable for users to shop online [1]. E-commerce IoT platforms often collect and analyze users' shopping and review information from edge devices such as smartphones. This information, especially user reviews, directly affects the purchase decision behaviors of potential consumers, as users usually look at other users' reviews on the product before purchasing and then decide whether to buy it. However, consumers' dependence on product reviews has catalyzed the emergence of fake reviews. Driven by economic interests, some merchants tend to attract consumers to write favorable reviews for their products by presenting gifts or small vouchers. The reviews that are unrealistic advocacy or defamatory for the products of merchants to influence the purchase decision of consumers are called fake reviews [2]. Fake reviews are very deceptive and many consumers are misled by such reviews [1]. Therefore, identifying the authenticity of user reviews has become an important research topic in the field of e-commerce IoT information security.

To make higher profits, some merchants might hire a group of reviewers working together to promote their products by fabricating product reviews. A reviewer who writes fake reviews is termed a spammer, and a reviewer group whose members collude to fabricate product reviews is called a collusive spamming group [3, 4]. The collusive spamming groups are more harmful to e-commerce platforms than individual spammers because they can produce fake reviews in batches for a period of time and will affect or even control the reputation of the evaluated products [5]. Therefore, how to identify the collusive spamming groups on e-commerce platforms has become an urgent problem to be solved.

To detect the collusive spamming groups on e-commerce platforms, a variety of approaches have been proposed. These approaches can be roughly categorized as group content and behavior analysis-based methods and graph-based methods [5, 6]. Group content and behavior analysis-based detection methods [3, 4, 7–9] employ the frequent itemset mining (FIM) technique to discover candidate spamming groups, which are only applicable to detecting

tightly coupled spamming groups. Moreover, FIM-based candidate group discovery methods are sensitive to the setting of support threshold. Graph-based detection methods [10–16] need to construct a user relationship graph by mining the relationships between reviewers in the dataset. However, these methods do not deeply mine the hidden relationships between reviewers when constructing the user relationship graph, which cannot fully detect the collusive spamming behaviors between reviewers. Furthermore, these methods tend to discard users whose correlation strength is less than a certain threshold during the process of constructing the user relationship graph, thus causing a negative impact on the detection performance.

To address the limitations above, we propose an approach for collusive spamming group detection based on network embedding, which is named CSGD-NE. We first model the given user review dataset as a heterogeneous information network (HIN) based on the idea of meta-graph. Then, we use the modified DeepWalk algorithm to perform random walks on the HIN to learn the low-dimensional vector representations of user nodes and employ the Canopy and K-means clustering algorithms to generate candidate spamming groups. Finally, we obtain the top-k groups with high spamming scores as the collusive spamming groups.

The contributions of this paper are summarized below:

- (1) Borrowing from the idea of meta-graph, we extract the key attributes such as users, products, and ratings required by the meta-graph from the given user review dataset to construct a HIN, which preserves more abundant structural and semantic information.
- (2) We modify the DeepWalk algorithm to learn the low-dimensional, dense, and real-value vector representations of user nodes in the HIN. Based on this, we perform the Canopy and K-means clustering algorithms in the embedding space to obtain candidate spamming groups.
- (3) We propose three indicators to measure the group spamming behaviors and combine the proposed indicators with two existing indicators to calculate the spamming score of each candidate group by using an indicator weighting strategy.
- (4) To show the performance of the proposed approach, we conduct extensive experiments on two real-world review datasets and compare our approach with four baseline methods.

The remainder of this paper is organized as follows. Section 2 introduces the related work on collusive spamming group detection. Section 3 describes the proposed approach, which includes the construction of HIN, generation of candidate collusive spamming groups, and detection of collusive spamming groups. The experimental results are reported in Section 4, and conclusions are drawn in Section 5.

2. Related Work

To detect individual spammers on e-commerce platforms, researchers have done in-depth work [5, 17–20]. In recent years, some researchers have focused on the collusive

spamming behaviors among reviewers and tried to detect collusive spamming groups on e-commerce platforms. In this section, we will discuss the related studies on collusive spamming group detection following the previously mentioned two categories. Table 1 summarizes the methods for collusive spamming group detection in line with these two categories.

2.1. Group Content and Behavior Analysis-Based Detection Methods. In the group content and behavior analysis-based detection methods, the correlation between group members is mined by analyzing the group review contents and behavior characteristics, and then the collusive spamming groups are identified through ranking or clustering methods. Mukherjee et al. [3] proposed the concept of the spamming group; they employed the FIM technique to discover candidate spamming groups and evaluated these candidate groups by analyzing the group members' review contents and the group members' unusual behaviors, and used a ranking method to obtain the collusive spamming groups. Xu et al. [4] calculated pairwise similarity between reviewers based on the review features and behavior features of reviewers and employed the k-nearest neighbors' method to predict top-k most similar reviewers. Xu and Zhang [7] presented an unsupervised framework to detect colluders, which used multiple heterogeneous pairwise features to capture the reviewers' collusive behaviors. In [8], a statistical model was proposed to model the collusive spamming behaviors among reviewers and an expectation maximization (EM) algorithm was employed to calculate the collusiveness of reviewers in the candidate groups that are obtained with the FIM technique. Zhang et al. [9] proposed a semisupervised method to identify spamming groups. They used the FIM technique to discover the candidate spamming groups and exploited the naive Bayesian model and EM algorithm to train a classifier for spamming group detection.

The abovementioned detection approaches mainly use the FIM technique to find candidate spamming groups, which are only suitable for tightly coupled spamming groups and are not applicable to loosely coupled spamming groups. Moreover, these approaches are sensitive to the setting of the support threshold. In addition, the semisupervised method needs a priori knowledge of spamming groups, which is often difficult to obtain in practice.

2.2. Graph-Based Detection Methods. Graph-based detection methods considered the specificity of collusive spamming groups in network structure and used the group structure features to identify the collusive spamming groups. Xu et al. [4] proposed a pairwise Markov network-based graph model to detect colluders, which took reviewers as nodes to construct a colluder graph model and used a probability graph model to predict the collusive spamming groups. Ye and Akoglu [10] used a 2-hop subgraph to detect the unusual behaviors of reviewers by analyzing their network footprints and employed the hierarchical clustering method to discover opinion spammer groups. Choo et al. [11] built user

TABLE 1: Summary of methods for collusive spamming group detection.

Category	Method	Data modeling	Candidate group discovery method	Spamming behavior evaluation	Detection method
Group content and behavior analysis-based methods	Mukherjee et al. [3]	Construct behavioral and relation models	Frequent itemset mining	The average of spam indicators	Ranking
	Xu et al. [4]	Measure the pairwise similarity of groups	Frequent itemset mining	No	KNN-based method
	Xu and Zhang [7]	Use pairwise features to model the relations among colluders	No	The weighted sum of all the pairwise features	Ranking
	Xu and Zhang [8]	Use homogeneity-based behavior features to model the collusive review fraud	Frequent itemset mining	Similarity-based measure	Unified probabilistic model
	Zhang et al. [9]	Use a number of features to model the group spamming behavior	Frequent itemset mining	No	Semi-supervised classification
Graph-based methods	Ye and Akoglu [10]	Model the dataset as a bipartite graph	No	Reviewers' network footprint scores	Hierarchical clustering
	Choo et al. [11]	Model the dataset as a user relationship graph	No	No	Community-based method
	Wang et al. [12]	Model the dataset as a bipartite graph	Graph partitioning	The average of spam indicators	Ranking
	Do et al. [13]	Model the dataset as a review graph	No	The average of spam indicators	k-means clustering
	Do et al. [14]	Model the dataset as a review graph	No	The average of spam indicators	Fuzzy k-means clustering
	Han et al. [15]	Model the dataset as a user relationship graph	No	No	Graph spectrum analysis
	Wang et al. [16]	Model the dataset as a user relationship graph	Graph partitioning	The average of spam indicators	Ranking
	Cao et al. [21]	Model the dataset as a bipartite graph	No	The average of spam indicators	Hierarchical agglomerative clustering
	Zhang et al. [22]	Model the dataset as a user relationship graph	Label propagation	Linearly weighted sum of spam indicators	Ranking

relationship graphs through the sentiment analysis of user interactions and used the strong positive communities of spammers to detect opinion spammer groups. Wang et al. [12] used a divide and conquer strategy to discover candidate spamming groups from the constructed reviewer relationship graph and introduced several spam indicators for detecting the review spammer groups. Do et al. [13] presented a review graph-based method to detect group spamming, which assigned a suspicious score to each user and employed the k-means algorithm to obtain the spamming groups. In [14], a network-based approach was utilized to identify individual spammers, and then a fuzzy k-means clustering algorithm was employed to find the group to which they belong. Han et al. [15] proposed an approach for detecting spammer groups based on graph spectrum analysis, which identified the anomaly structure in the constructed user relationship graph through analyzing the spectrum features. Wang et al. [16] used the minimum cut algorithm to split the user relationship graph to obtain candidate groups and employed a set of spam indicators to detect the spammer groups. Cao et al. [21] extracted the multiview anomalous features of collusive spammers and employed a hierarchical clustering algorithm to detect the spammer groups in location-based

social networks. In our recent work [22], we used a Label propagation-based approach for detecting review spammer groups.

The above detection approaches are mainly based on the homogeneous graph. As these methods do not deeply mine the implicit relationship between reviewers in the review dataset when constructing the user relationship graph, they cannot fully detect the collusive spamming behaviors between reviewers. Moreover, these methods tend to discard some users during the construction of the user relationship graph, causing a negative impact on the detection performance. In addition, these methods take the average of all spam indicator values as the suspicious score of the candidate group, ignoring the difference among spam indicators.

3. The Proposed Detection Approach

The framework of the proposed detection approach is illustrated in Figure 1 and includes three phases. Figure 2 gives the scenario diagram for our detection approach. In the first phase, the key attributes such as users, products, and ratings required by the meta-graph are extracted from the user review dataset to construct a HIN. In the second phase, the modified DeepWalk algorithm is employed to perform equal probability

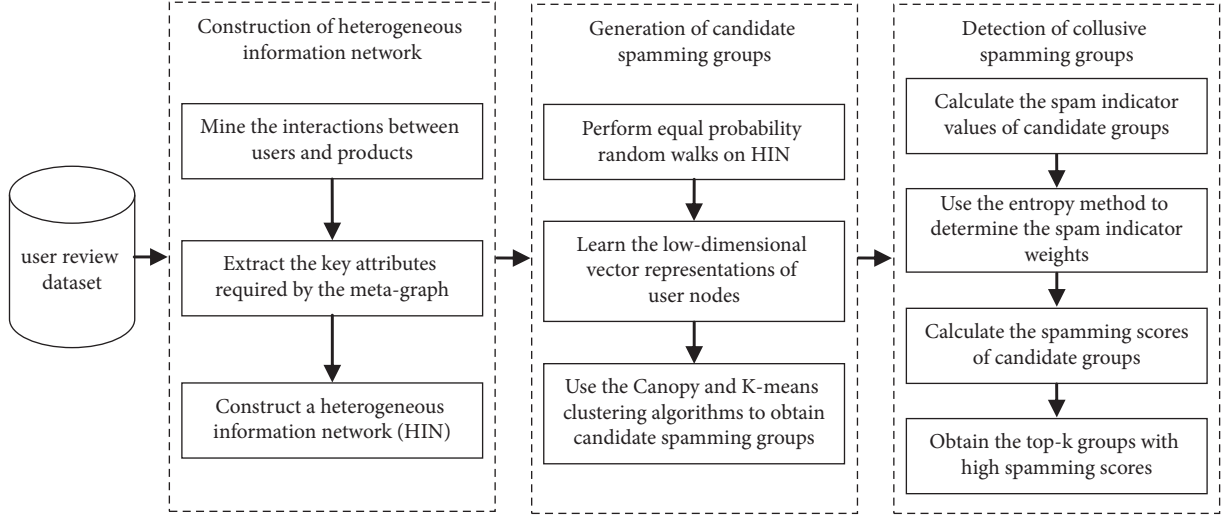


FIGURE 1: The framework of the proposed detection approach.

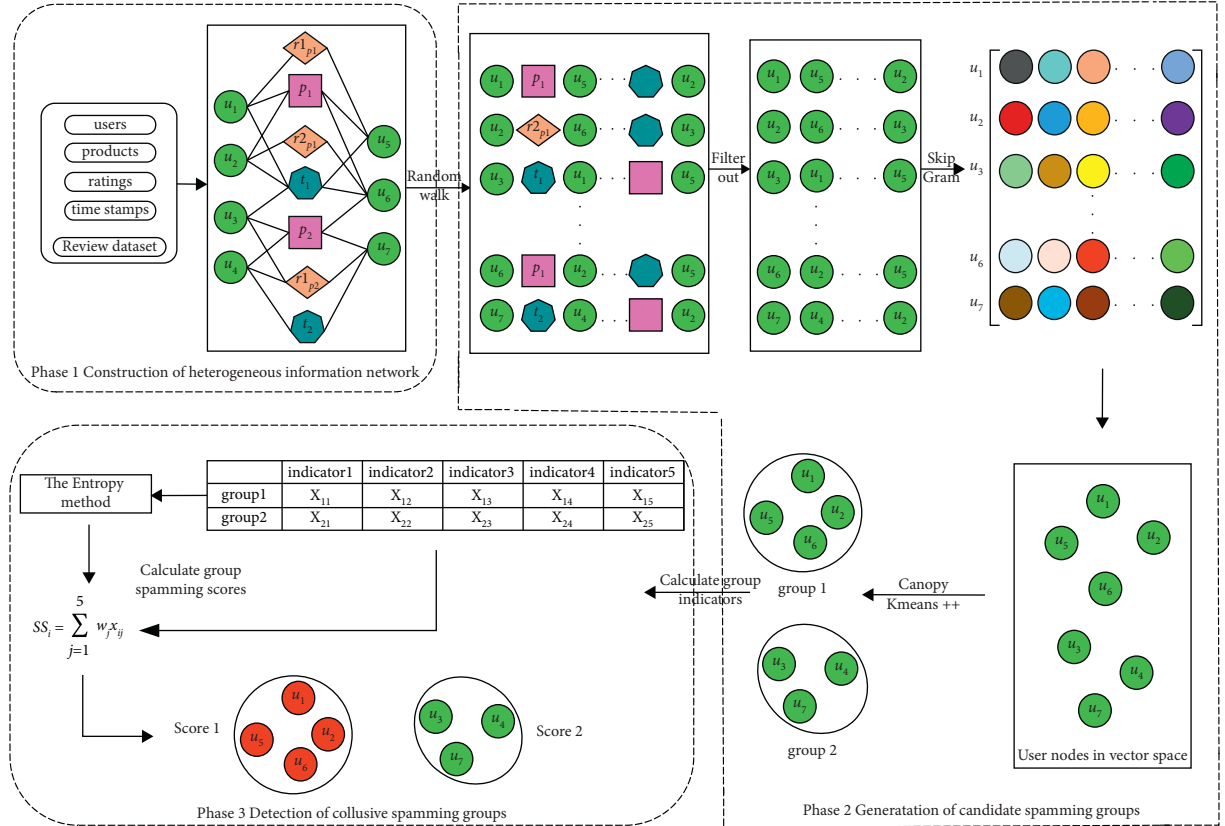


FIGURE 2: The scenario diagram of the proposed detection approach.

random walks on the HIN to learn the low-dimensional vector representations of user nodes, and the Canopy and K-means clustering algorithms are applied to generate candidate spamming groups. In the last phase, the spamming score of each candidate group is calculated by weighting all spam indicator values, and the top-k groups with high spamming scores are treated as the collusive spamming groups.

3.1. Construction of Heterogeneous Information Network. In this section, we model the given user review dataset as a HIN. Particularly, we extract the key attributes, including users, products, ratings, and timestamps required by the meta-graph from the user review dataset, and construct the HIN based on the idea of meta-graph. First, we introduce some concepts related to the HIN.

Definition 1 (heterogeneous information network [23]). Let V , E , A , and \mathcal{R} denote sets of nodes, edges, node types, and edge types, respectively; a HIN is defined as a graph $G = (V, E)$ with a node type mapping function $\phi: V \rightarrow A$ and an edge type mapping function $\psi: E \rightarrow \mathcal{R}$, where each node $v \in V$ belongs to one particular node type $\phi(v) \in A$; each edge $e \in E$ belongs to one particular relation $\psi(e) \in \mathcal{R}$, and at least one of $|A|$ and $|\mathcal{R}|$ is greater than 1.

Definition 2 (network schema [24]). Given a HIN $G = (V, E)$ with two mapping functions $\phi: V \rightarrow A$ and $\psi: E \rightarrow \mathcal{R}$, the network schema of G is a directed graph defined over node types A and edge types \mathcal{R} , which is denoted as $T_G = (A, \mathcal{R})$.

The network schema of a HIN serves as a template for the HIN, indicating how many objects are in the HIN and whether links may exist between these objects.

Definition 3 (meta-path [24]). Given a HIN $G = (V, E)$ with a network schema $T_G = (A, \mathcal{R})$, a meta-path P is a path represented in the form of $P = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_i} A_{i+1}$, where $A_i \in A$ and R_i denotes the relation between node types A_i and A_{i+1} .

A meta-path represents the relationship between nodes in a HIN, and different types of meta-paths represent different meanings between nodes.

Definition 4 (meta-graph [25]). Given a HIN $G = (V, E)$ with a network schema $T_G = (A, \mathcal{R})$, a meta-graph $S = (N, M, n_s, n_t)$ is a directed acyclic graph defined on T_G , where $N \subseteq V$ and $M \subseteq E$, n_s and n_t are the source and target nodes, respectively.

Unlike meta-paths, meta-graphs have a more complex and flexible structure, which can be used to represent more complex relationships. Based on the idea of meta-graph, the steps for constructing the HIN from the given user review dataset are as follows.

First, we extract the key attributes and mine the interactions between users and products from the review dataset to construct a meta-graph. The extracted attributes include users, products, ratings, and time-stamps, which are treated as user nodes, product nodes, rating nodes, and time nodes in the HIN. To avoid ambiguity, the four types of nodes are renumbered uniformly in this paper. The rating node contains the product ID information because the product reviewed by the user may not be the same when the user gives the same rating. Based on the extracted attributes and interactions, we can construct three meta-paths, i.e., user-product-user, user-rating-user, and user-time-user, which represent that two users reviewed the same product, gave the same rating for a product, and reviewed a product at the same time, respectively. A meta-graph can be built by connecting the three meta-paths together.

Second, we construct the HIN by extending the meta-graph. Data objects in the given user review dataset can be connected through the meta-graph and not just a single path. Even if two users have not reviewed the

same product, there may be a correlation between them if their review time is the same. Therefore, the meta-graph can be used to explore the potential relationship between users from three products, rating, and review time.

3.2. Generation of Candidate Spamming Groups. In this section, we modify the DeepWalk algorithm to learn the low-dimensional vector representations of user nodes in the HIN and perform clustering in the embedding space to obtain candidate spamming groups.

3.2.1. The Modified DeepWalk Algorithm. DeepWalk [26] is a network embedding method based on random walks and skip-gram modeling. Based on the idea of Word2vec [27], it takes the sequence of nodes obtained by random walks as a sentence, obtains the local information of the network from the truncated random walk sequence, and then learns the latent representations of nodes through the local information.

DeepWalk is applicable to learning the representations of nodes in the homogeneous graph, while the HIN we construct in this paper belongs to a heterogeneous graph that includes four types of nodes (i.e., user node, product node, rating node, and time node). Therefore, we need to modify the DeepWalk algorithm to make it suitable for learning the representations of user nodes in the constructed HIN in order to divide candidate spamming groups. The main steps of the modified DeepWalk algorithm are as follows.

Step 1. Each user node is taken as the starting node in turn, and any node is selected from the set of adjacent nodes of the current node to walk randomly with equal probability.

Step 2. At the end of each random walk, the node type is judged. If the node belongs to a user node, it will be added to the random walk sequence, and then the next random walk will be conducted.

Step 3. Repeat the above steps until the length of the random walk reaches a given threshold. Put the obtained sequence of user nodes into the skip-gram model for training and generate the user node vector representation.

Based on the above steps, the modified DeepWalk algorithm is described below.

Algorithm 1 generates the low-dimensional vector representations of user nodes in the HIN through equal probability random walks. Particularly, it first takes each user node as the starting node for random walks and obtains the walk sequence composed of all types of nodes (Lines 4–6). Then the user nodes are extracted from the walk sequence W_{vi} and the user node sequence S_u is generated (Lines 7–12). Finally, the generated user node sequence S_u is used as the input of the skip-gram model and the vector representation of each user node is obtained (Line 13).

```

Input: HIN  $G$ , window size  $ws$ , embedding size  $d$ , the number of walks per node  $n$ , walk length  $l$ , set of user nodes  $V_u$ 
Output: matrix of user node representations  $\Phi \in \mathbb{R}^{|V_u| \times d}$ 
(1) Initialization: Sample  $\Phi$  from  $U^{|V_u| \times d}$ 
(2) Build a binary tree from  $V_u$ 
(3) for  $i=0$  to  $n$  do
(4)    $O \leftarrow \text{Shuffle}(V_u)$ 
(5)   for each  $v_i \in O$  do
(6)      $W_{v_i} \leftarrow \text{RandomWalk}(G, v_i, l)$ 
(7)      $S_u \leftarrow \emptyset$ 
(8)     for each node  $v \in W_{v_i}$  do
(9)       if  $v \in V_u$  then
(10)        add  $v$  to  $S_u$ 
(11)       end if
(12)     end for
(13)     SkipGram ( $\Phi, S_u, ws$ )
(14)   end for
(15) end for
(16) return  $\Phi$ 

```

ALGORITHM 1: The modified DeepWalk algorithm.

3.2.2. *Generating Candidate Spamming Groups Based on the Canopy and K-Means Clustering.* Based on the user vector representations obtained with the modified DeepWalk algorithm, we employ a combination of the Canopy clustering and the K-means clustering to generate candidate spamming groups. Canopy clustering [28] is a fast approximate clustering algorithm, which does not need to specify the number of clusters in advance. While this clustering algorithm cannot produce accurate clustering results, it can give the optimal number of clusters. The K-means clustering is a simple, efficient, and effective clustering algorithm, but it has limitations such as sensitivity to the selection of initial cluster centers and need to specify the number of clusters in advance. Therefore, we can use the Canopy clustering algorithm as the pre-processing of the K-means clustering algorithm to determine the optimal number of clusters and the best initial cluster centers for the K-means clustering.

The basic idea of generating candidate spamming groups based on the Canopy and K-means clustering algorithms is as follows. First, we treat the user vector representations as a set of sample objects and employ the Canopy clustering algorithm to produce a number of Canopy sets (or clusters). Second, we take the number of clusters and the cluster centers obtained with the Canopy clustering algorithm as the cluster number K and initial cluster centers of the K-means algorithm to perform clustering on the set of sample objects to generate the candidate spamming groups. The specific algorithm is described as follows.

Algorithm 2 first calculates the average Euclidean distance between sample objects (Lines 2–7), which is used as a distance threshold of the Canopy algorithm, where Function `calculateEuclideanDistance()` is used to calculate the Euclidean distance between two objects. Then it performs the Canopy clustering algorithm to obtain the canopy sets and cluster centers (Lines 8–23), where Function `getClusterCenter()` is used to obtain the center of a cluster. Finally, the

number of clusters $|C|$ and the cluster centers are used as the cluster number and the initial cluster centers of the K-means clustering algorithm to generate the candidate spamming groups (Line 24).

3.3. *Detection of Collusive Spamming Groups.* In this section, we propose three spamming group detection indicators and combine them with two existing detection indicators to calculate the spamming scores of the candidate groups. Based on this, we rank the candidate groups according to their spamming scores and consider the top- k most suspicious groups as the collusive spamming groups.

3.3.1. *The Spamming Group Detection Indicators.* For a product, if most of the product’s review information comes from the same group, it is more likely that the merchant hires a spamming group for profit. If there exists a significant difference in the review ratio of products within and outside a group, then it is likely that the group is hired by a merchant, and the more likely it is to be a spamming group.

Definition 5. (review ratio of products within and outside a group, RRP): The review ratio of products within and outside a group g refers to the mean of review ratio values of products within and outside group g , whose normalized value is calculated as follows:

$$RRP(g) = \left(\frac{2}{1 + e^{-1/|P_g| \left| \sum_{p \in P_g} |R_{p,g}| / |R_{p,D}| \right|}} - 1 \right) \times L(g), \quad (1)$$

where P_g denotes the set of products that are reviewed by the reviewers in group g , $R_{p,D}$ and $R_{p,g}$ denote the sets of reviewers who have reviewed product p in the dataset D and in group g , respectively. $L(g)$ is used to reduce the possibility

```

Input: set of sample objects S
Output: candidate spamming groups CPG
(1) dis←0, C←∅, Cluster Centers←∅
(2) for each object oi ∈ S do
(3)   for each object oj ∈ S do
(4)     dis←dis + calculateEuclideanDistance(oi, oj)
(5)   end for
(6) end for
(7) T2←dis÷(|S| × (|S| - 1)÷2)
(8) while S is not empty do
(9)   choose a sample object s from S to initialize a canopy set c
(10)  S←S - {s}
(11)  for each object o in S do
(12)    dis←calculateEuclideanDistance(s, o)
(13)    if dis < T2 + 1 then
(14)      c←c ∪ {o}
(15)    end if
(16)    if dis < T2 then
(17)      S←S - {o}
(18)    end if
(19)  end for
(20)  C←C ∪ c
(21)  Center←getClusterCenter(c)
(22)  Cluster Centers←Cluster Centers ∪ {Center}
(23) end while
(24) CPG←doK-means(S, |C|, Cluster Centers)
(25) return CPG

```

ALGORITHM 2: Generating the candidate spamming groups.

that group g is formed by accident, which is calculated as follows [12]:

$$L(g) = \frac{1}{1 + e^{-\left(|R_g| + |P_g| - 3\right)}}, \quad (2)$$

where R_g denotes the set of reviewers in group g and P_g has the same meaning as in equation (1).

In order to promote the target product (s), the members in a spamming group are often required to review the product (s) in a short time. Therefore, the reviews of the spamming group have the characteristics of time concentration or outburst. We can use a single-day review outburst indicator to measure this spamming behavior of a group. The higher the single-day review outburst of a group, the more likely it is to be a spamming group.

Definition 6 (single-day review outburst, SRO). The single-day review outburst of a group g refers to the average ratio of the highest number of reviews written by a user in a single day to the total number of reviews written by users in group g , whose normalized value is calculated as follows:

$$SRO(g) = \frac{1}{1 + e^{-\text{avg}_{u \in R_g} R_u^h / R_g^t}} \times L(g), \quad (3)$$

where R_u^h denotes the highest number of reviews written by user u in a single day, R_g^t denotes the total number of reviews written by the users in group g , R_g and $L(g)$ have the same meaning as in equations (2) and (1), respectively.

Generally, users in a spamming group tend to give extreme ratings for the target products, while normal reviewers tend to give relatively scattered ratings. If there is a significant difference in the rating of the target product between the users in a group and the normal reviewers, it indicates that the users in the group have strong intentionality in the product reviewed. Thus it can be inferred that the group is likely to be a spamming group. We can use the rating deviation indicator to measure this spamming behavior of a group.

Definition 7 (rating deviation, RD). The rating deviation of a group g is the mean of the products' average rating differences between users within and outside group g , whose normalized value is calculated as follows:

$$RD(g) = \frac{\text{avg}_{p \in P_g} |i\text{avg}(p) - o\text{avg}(p)|}{r_{\max} - r_{\min}} \times L(g), \quad (4)$$

where $i\text{avg}(p)$ and $o\text{avg}(p)$ denote the average ratings of users within and outside group g for product p , respectively, r_{\max} and r_{\min} denote the maximum and minimum ratings in the dataset, respectively, and P_g and $L(g)$ have the same meaning as in equation (1).

Considering the purpose of spamming group, the members in a spamming group often review the preplanned products to be promoted or demoted and rarely or never review other products. Therefore, if the members in a group review too closely on the products, the group is likely to be a

spamming group. We can use the product tightness indicator to measure this spamming behavior of a group. The greater the product tightness of a group, the more likely it is to be a spamming group.

Definition 8 (product tightness, PT [16]). The product tightness of a group g refers to the ratio of the number of products co-reviewed by the reviewers in group g to the total number of products reviewed by the reviewers in group g , that is,

$$PT(g) = \frac{|\cap_{u \in R_g} P_u|}{|\cup_{u \in R_g} P_u|}, \quad (5)$$

where P_u denotes the set of products reviewed by reviewer u and R_g has the same meaning as in equation (2).

Normal reviewers usually make an objective evaluation according to the actual situation of a product, so they may give different ratings for the same products. At the same time, reviewers in a spamming group tend to have consistent ratings on the target products to be promoted or demoted. Therefore, we can use the variance of the target product's ratings provided by the reviewers in a group to measure the spamming behavior of the group.

Definition 9 (rating variance, RV [16]). The rating variance of a group g refers to the average of rating variances of the target products and is calculated as follows:

$$RV(g) = 2 \left(1 - \frac{1}{1 + e^{-avg_{p \in P_g} S^2(p,g)}} \right), \quad (6)$$

where $S^2(p, g)$ denotes the variance of product p 's ratings provided by the reviewers in group g .

3.3.2. Obtaining the Collusive Spamming Groups. Based on the spamming group detection indicators described in Section 3.3.1, we can calculate a spamming score for each candidate group. Considering the differences in the importance of these indicators when measuring the spamming behavior of a group, we employ the entropy method [29,30] to determine the weights of these indicators and take the weighted sum of these indicator values as the group's spamming score.

Suppose we have m candidate groups and n detection indicators, x_{ij} ($i = 1, 2, \dots, m; j = 1, 2, \dots, n$) denotes the j -th indicator value of the i -th group, the steps that employ the entropy method to determine the weights of these indicators are as follows.

Step 1. Calculate the normalized value of x_{ij}

$$z_{ij} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)}, \quad i = 1, 2, \dots, m; j = 1, 2, \dots, n, \quad (7)$$

where x_j represents the list of the j -th indicator values for each group.

Step 2. Calculate the ratio of the i -th group in the j -th indicator

$$r_{ij} = \frac{z_{ij}}{\sum_{i=1}^m z_{ij}}, \quad i = 1, 2, \dots, m; j = 1, 2, \dots, n. \quad (8)$$

Step 3. Calculate the j -th indicator's entropy

$$ie_j = \frac{-\sum_{i=1}^m r_{ij} \ln r_{ij}}{\ln m}, \quad j = 1, 2, \dots, n. \quad (9)$$

Step 4. Calculate each indicator's entropy weight

$$w_j = \frac{1 - ie_j}{\sum_{j=1}^n (1 - ie_j)}, \quad j = 1, 2, \dots, n. \quad (10)$$

The greater the entropy weight of an indicator, the more important it is in measuring the group spamming behavior.

Definition 10 (spamming score of group, SS). The spamming score of a group i is defined as the weighted sum of all indicator values of group i , that is,

$$SS_i = \sum_{j=1}^n w_j x_{ij}, \quad i = 1, 2, \dots, m, \quad (11)$$

where x_{ij} denotes the j -th indicator value of the i -th group and w_j denotes the entropy weight of the j -th indicator. The greater the spamming score of a group, the more likely it is to be a spamming group.

We can use equation (11) to calculate the spamming scores of the candidate groups. We rank the candidate groups according to their spamming scores and take top- k groups with high spamming scores as the collusive spamming groups. The algorithm of obtaining the collusive spamming groups is described below.

Algorithm 3 consists of three parts. The first part (Lines 1–6) calculates all indicator values of the candidate spamming groups and the entropy weights of five indicators. The second part (Lines 7–12) calculates the spamming scores of the candidate groups. The third part (Line 13) obtains the top- k groups with high spamming scores.

4. Experimental Evaluation

4.1. Experimental Datasets. We use two real-world datasets as the experimental data to evaluate our approach.

- (1) Amazon review dataset [4, 31]: this dataset was crawled from Amazon.cn till August 20, 2012. It includes 1205125 ratings/reviews from 645072 reviewers on 136785 products, in which 5055 reviewers have labels indicating normal users or spammers. The ratings in this dataset are integers between 1 and 5, indicating how much users like the products they have evaluated, where 1 indicates disliked and 5 indicates most liked. For the convenience of experimental evaluation, we extract these 5055 reviewers with labels, their ratings on products, and

```

Input: CSG: the candidate spamming groups
Output: TKG: the top-k groups with high spamming scores
(1) for each group  $i=1$  to  $|CSG|$  do
(2)   for each indicator  $j=1$  to 5 do
(3)      $x[i][j] \leftarrow$  calculate the  $j$ -th indicator value of the  $i$ -th group
(4)   end for
(5) end for
(6)  $w \leftarrow$  calculate the entropy weights of five indicators using equations (7)–(10)
(7) for each group  $i=1$  to  $|CSG|$  do
(8)    $SS[i] \leftarrow 0$ 
(9)   for each indicator  $j=1$  to 5 do
(10)     $SS[i] \leftarrow SS[i] + w[j] * x[i][j]$ 
(11)  end for
(12) end for
(13)  $TKG \leftarrow$  getTopkGroups(CSG, SS)
(14) return TKG

```

ALGORITHM 3: Obtaining the collusive spamming groups.

their rating timestamps from the dataset to construct a sampled dataset. The sampled dataset contains 53777 ratings from 5055 reviewers on 17610 products, which includes 3118 normal users and 1937 spammers.

- (2) Yelp_Miami review dataset: in [32], the authors constructed a dataset to classify fake reviews by crawling business reviews in the consumer electronics domain in four American cities (i.e., New York, Los Angeles, Miami, and San Francisco) from Yelp.com till 2017. The reviews in this dataset were labeled as fake or not by Yelp’s antifraud filter, which can be treated as near ground truth. The reviewers corresponding to the fake or not reviews are considered as spammers or normal reviewers. Yelp_Miami review dataset is a subset of this dataset, which includes 3463 ratings from 3311 reviewers on 549 electronic businesses in the city of Miami. The ratings in the Yelp_Miami review dataset are integers between 1 and 5, where 1 indicates disliked and 5 indicates most liked. In the Yelp_Miami review dataset, there are 1394 spammers and 1917 normal reviewers.

4.2. Evaluation Metrics. We use precision@k ($P@k$) and recall@k ($R@k$) metrics to evaluate the performance of our approach, which are defined as follows [33]:

$$P@k = \frac{|\mathcal{U}_{\text{spam}} \cap \mathcal{U}_D|}{|\mathcal{U}_{\text{spam}}|}, \quad (12)$$

$$R@k = \frac{|\mathcal{U}_{\text{spam}} \cap \mathcal{U}_D|}{|\mathcal{U}_D|},$$

where $\mathcal{U}_{\text{spam}}$ denotes the set of top- k reviewers who are considered to be spam users and \mathcal{U}_D denotes the set of spam users in the dataset D .

4.3. Experimental Results and Analysis. To illustrate the effectiveness of the proposed approach (CSGD-NE), we conduct experiments on two real-world review datasets and compare CSGD-NE with the following four methods.

- (1) GSBC [16]: a graph-based spammer group detection method, which models spammer groups as bi-connected graphs and treats the bi-connected components whose spamming scores exceed the given threshold as the spammer groups. In the experiments, the parameters τ , δ , MP , and $MINSPAM$ for GSBC are set to 30, 0.1, 1000, and 0.49 on the Amazon review dataset and 10, 0.4, 10000, and 0.59 on the Yelp_Miami review dataset, respectively.
- (2) FAP [33]: a fraudulent action propagation-based method for spammer detection, which employs the label propagation method to iteratively calculate the spamming probability values of users based on the labeled seed spam users. In the experiments, we randomly select 5 spam users as the seed users. To reduce the randomness of the experimental results, the average values of 10 experiments are used as the final results.
- (3) CONSGD [34]: a spammer group detection method that employs cosine pattern mining to find candidate spammer groups and models the candidate groups and their relations as an HIN. CONSGD uses the label propagation method to calculate the spamming probabilities of the candidate groups based on the labeled instances and treats the groups with high spamming probabilities as the spammer groups. In the experiments, the parameters τ_{AR} , τ_{MS} , α , and k for CONSGD are set to 0.2, 0.01%, 0.3, and 20, respectively.
- (4) GSDB [6]: a burst-based method to detect review spammer groups, which discovers candidate spammer groups in review bursts using the Kernel Density

Estimation algorithm, and then exploits both individual and group spam indicators to classify spammer groups. In the experiments, parameters θ , δ , δ_{TP} , δ_I , δ_G , k , h , and $ISIZE$ used in GSDB are set to 30, 30, 0.1, 0.2, 0.3, 0.5, 0.075, and 7, respectively.

In the experiments, the window size and embedding size for CSGD-NE are set to 5 and 64, respectively. The number of walks per node and walk length for CSGD-NE are set to 100 and 10 on the Amazon review dataset and 10 and 20 on the Yelp_Miami review dataset, respectively.

4.3.1. Comparison of Detection Performance for Five Methods on the Amazon Review Dataset. Figures 3 and 4 show the comparison of $P@k$ and $R@k$ for CSGD-NE, GSBC, CONSGD, FAP, and GSDB with various numbers of top- k reviewers on the Amazon review dataset.

As shown in Figure 3, on the Amazon review dataset, the $P@k$ of CONSGD is the worst on the whole among the five methods and tends to decrease as the number of top- k reviewers increases. FAP has good $P@k$ before the top-330 reviewers. This is because FAP uses some seed spam users to drive the fraudulent action propagation, which leads to very high initial detection precision. However, the $P@k$ of FAP shows a downward trend with the number of top- k reviewers increasing. This indicates that an increasing number of normal reviewers are misjudged as spam users. The $P@k$ of GSDB and GSBC is overall better than that of CONSGD and FAP and gradually becomes stable at about 0.76 and 0.71, respectively, after the top-600 reviewers. Furthermore, the $P@k$ value of GSBC keeps above 0.7 at the top-2000 reviewers. However, the $P@k$ curve of GSDB ends before top-1450, indicating that the total number of spammers obtained by GSDB is less than 1450. The $P@k$ of CSGD-NE increases gradually before the top-330 reviewers and subsequently tends to be stable. Moreover, the $P@k$ value of CSGD-NE still keeps 0.88 at the top-2000 reviewers, which is obviously higher than that of the baseline methods. Therefore, the $P@k$ of CSGD-NE is overall better than that of GSBC, CONSGD, FAP, and GSDB on the Amazon review dataset.

As shown in Figure 4, the $R@k$ of CONSGD and FAP on the Amazon review dataset shows an upward trend before the top-800 reviewers. Subsequently, it has little change as the number of top- k reviewers increases. This phenomenon shows that the number of spam users identified by CONSGD and FAP is basically unchanged after the top-800 reviewers. In contrast, the $R@k$ of CSGD-NE, GSDB, and GSBC tends to increase as the number of top- k reviewers increases, which means more and more spam users are detected by the three methods. Moreover, the $R@k$ value of CSGD-NE and GSBC at the top-2000 reviewers still maintains at about 0.92 and 0.73, respectively. Clearly, the $R@k$ of CSGD-NE is much better on the whole than that of GSBC, CONSGD, FAP, and GSDB on the Amazon review dataset.

Based on the above analysis, we can conclude that CSGD-NE outperforms GSBC, CONSGD, FAP, and GSDB in terms of $P@k$ and $R@k$ metrics when detecting collusive spamming groups in the Amazon review dataset.

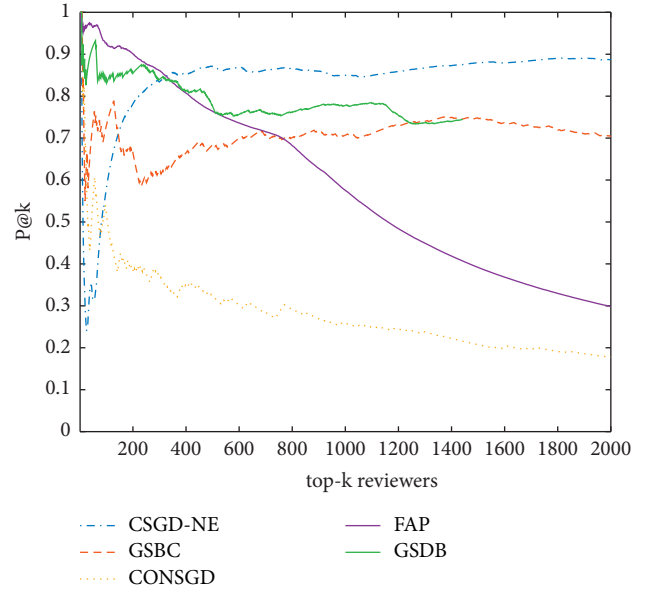


FIGURE 3: Comparison of $P@k$ for five methods on the Amazon review dataset.

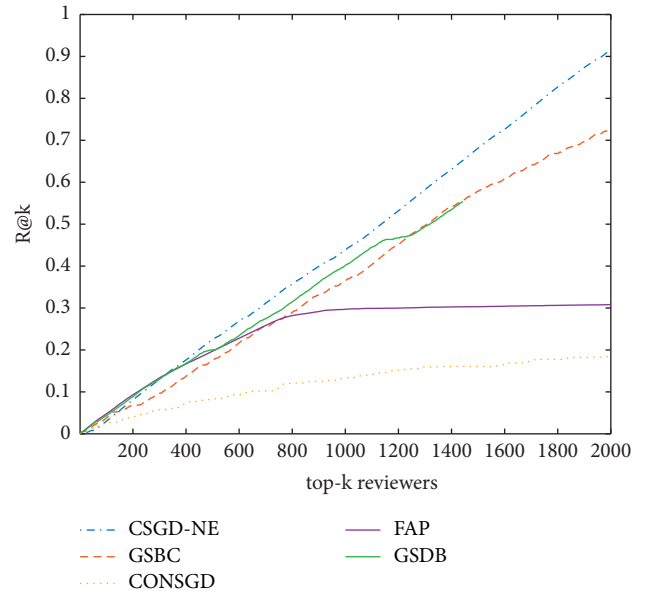


FIGURE 4: Comparison of $R@k$ for five methods on the Amazon review dataset.

4.3.2. Comparison of Detection Performance for Five Methods on the Yelp_Miami Review Dataset. Figures 5 and 6 show the comparison of $P@k$ and $R@k$ for CSGD-NE, GSBC, CONSGD, FAP, and GSDB with various numbers of top- k reviewers on the Yelp_Miami review dataset.

It can be seen from Figure 5 that the $P@k$ curve of GSBC on the Yelp_Miami review dataset ends at the top-70 reviewers, indicating that the number of spamming groups that GSBC can detect on this dataset is very limited. This phenomenon is not difficult to explain. GSBC employs the minimum cut algorithm to divide candidate groups, which easily produces isolated nodes. Moreover, the candidate groups with spamming scores

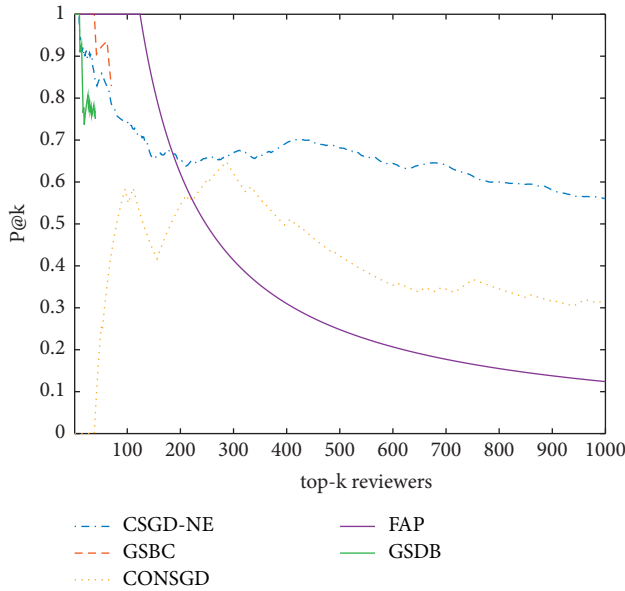


FIGURE 5: Comparison of $P@k$ for five methods on the Yelp_Miami review dataset.

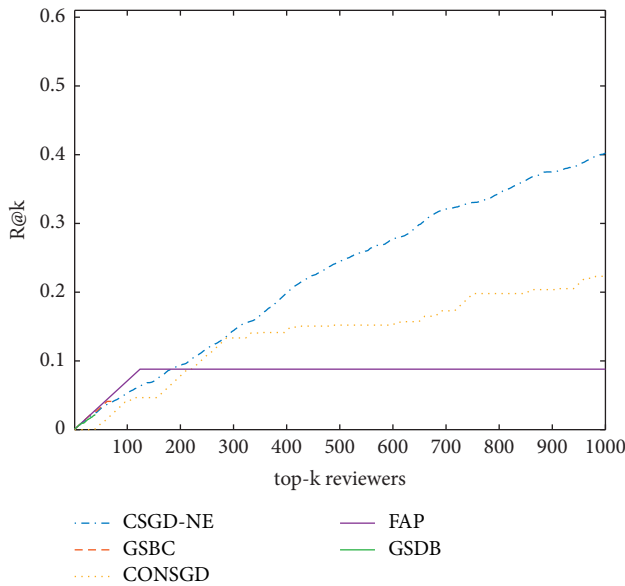


FIGURE 6: Comparison of $R@k$ for five methods on the Yelp_Miami review dataset.

below a given threshold are discarded. GSDB has similar phenomena on this dataset. The $P@k$ of FAP on the Yelp_Miami review dataset is always 1 before the top-125 reviewers. Subsequently, it gradually decreases as the number of top- k reviewers increases and it is only 0.124 at the top-1000 reviewers, which means more and more normal reviewers are misjudged as spam users by FAP on this dataset. The $P@k$ of CONSGD on the Yelp_Miami review dataset is lower than that of FAP before the top-225 reviewers, and subsequently, it is better than that of FAP on this dataset. While CONSGD can detect more spam users than GSBC on the Yelp_Miami review dataset, its $P@k$ is lower than that of GSBC. The $P@k$ of CSGD-

NE on the Yelp_Miami review dataset shows a downward trend before the top-150 reviewers and subsequently tends to be stable. Clearly, the $P@k$ of CSGD-NE is, on the whole, better than that of GSBC, CONSGD, FAP, and GSDB on the Yelp_Miami review dataset.

As shown in Figure 6, the $R@k$ curve of GSDB and GSBC on the Yelp_Miami review dataset end before top-71 reviewers because they cannot spot more spamming groups in this dataset. The $R@k$ of FAP on the Yelp_Miami review dataset increases gradually before the top-125 reviewers. Subsequently, it tends to be stable and maintains at about 0.088, which means no spamming groups are detected by FAP after the top-125 reviewers. The $R@k$ of CONSGD on the Yelp_Miami review dataset tends to increase as more and more spam users are spotted by CONSGD and reaches 0.22 at the top-1000 reviewers, which is, on the whole, better than that of GSBC and FAP. The $R@k$ of CSGD-NE on the Yelp_Miami review dataset also increases gradually as an increasing number of spam users are detected and reaches about 0.4 at the top-1000 reviewers. Therefore, the $R@k$ of CSGD-NE is overall better than that of GSBC, CONSGD, FAP, and GSDB on the Yelp_Miami review dataset.

Based on the above analysis, we can conclude that CSGD-NE performs better than GSBC, CONSGD, FAP, and GSDB when detecting collusive spamming groups in the Yelp_Miami review dataset.

5. Conclusions

The collusive spamming groups can produce fake product reviews in batches, which poses a serious challenge to the credibility of e-commerce platforms. In this paper, we propose a network embedding based approach for detecting collusive spamming groups. Borrowing from the idea of meta-graph, we model the given user review dataset as a HIN. We modify the DeepWalk algorithm to learn the low-dimensional vector representations of user nodes in the HIN and obtain the candidate spamming groups by combining the Canopy and K-means clustering algorithms. We put forward three spamming group detection indicators and combined them with two existing detection indicators to measure the spamming behavior of each candidate group. By ranking the candidate groups according to their spamming scores, we obtain top- k groups with high spamming scores as the collusive spamming groups. The experimental results on two real-world review datasets demonstrate the superiority of the proposed method in detecting the collusive spamming groups.

Data Availability

The data used to support the findings of this study are available from the first author (xjcb@ysu.edu.cn) upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 62072393).

References

- [1] R. K. Dewang and A. K. Singh, "State-of-art approaches for review spammer detection: a survey," *Journal of Intelligent Information Systems*, vol. 50, no. 2, pp. 231–264, 2018.
- [2] A. Heydari, M. a. Tavakoli, N. Salim, and Z. Heydari, "Detection of review spam: a survey," *Expert Systems with Applications*, vol. 42, no. 7, pp. 3634–3642, 2015.
- [3] A. Mukherjee, B. Liu, and N. Glance, "Spotting fake reviewer groups in consumer reviews," in *Proceedings of the 21st International Conference on World Wide Web*, pp. 191–200, Lyon, France, April 2012.
- [4] C. Xu, J. Zhang, K. Chang, and C. Long, "Uncovering collusive spammers in Chinese review websites," in *Proceedings of the 22nd ACM International Conference On Information & Knowledge Management*, pp. 979–988, San Francisco, CA, USA, November 2013.
- [5] L. Li, B. Qin, and T. Liu, "Survey on fake review detection research," *Chinese Journal of Computers*, vol. 41, no. 4, pp. 946–968, 2018, in Chinese.
- [6] S.-j. Ji, Q. Zhang, J. Li et al., "A burst-based unsupervised method for detecting review spammer groups," *Information Sciences*, vol. 536, pp. 454–469, 2020.
- [7] C. Xu and J. Zhang, "Combating Product Review Spam Campaigns via Multiple Heterogeneous Pairwise Features," in *Proceedings of the Siam International Conference On Data Mining*, pp. 172–180, Vancouver, BC, Canada, April 2015.
- [8] C. Xu and J. Zhang, "Towards collusive fraud detection in online reviews," in *Proceedings of the 15th IEEE International Conference On Data Mining*, pp. 1051–1056, Atlantic City, NJ, USA, November 2015.
- [9] L. Zhang, Z. Wu, and J. Cao, "Detecting spammer groups from product reviews: a partially supervised learning model," *IEEE Access*, vol. 6, pp. 2559–2568, 2018.
- [10] J. Ye and L. Akoglu, "Discovering Opinion Spammer Groups by Network Footprints," in *Proceedings of the Joint European Conference On Machine Learning And Knowledge Discovery In Databases*, pp. 267–282, Porto, Portugal, September 2015.
- [11] E. Choo, T. Yu, and M. Chi, "Detecting opinion spammer groups through community discovery and sentiment analysis," in *Proceedings of the 29th Annual IFIP Conference On Data And Applications Security And Privacy*, pp. 170–187, Fairfax, VA, USA, July 2015.
- [12] Z. Wang, T. Hou, D. Song, Z. Li, and T. Kong, "Detecting review spammer groups via bipartite graph projection," *The Computer Journal*, vol. 59, no. 6, pp. 861–874, 2016.
- [13] Q. N. T. Do, A. Zhilin, C. Z. P. Junior, G. Wang, and F. K. Hussain, "A network-based approach to detect spammer groups," in *Proceedings of the International Joint Conference On Neural Networks*, pp. 3642–3648, Vancouver, BC, Canada, July 2016.
- [14] Q. N. T. Do, F. K. Hussain, and T. N. Bang, "A fuzzy approach to detect spammer groups," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 1–6, Naples, Italy, July 2017.
- [15] Z. Han, K. Yang, and X. Tan, "Analyzing spectrum features of weight user relation graph to identify large spammer groups in online shopping websites," *Chinese Journal of Computers*, vol. 40, no. 4, pp. 939–954, 2017, in Chinese.
- [16] Z. Wang, S. Gu, X. Zhao, and X. Xu, "Graph-based review spammer group detection," *Knowledge and Information Systems*, vol. 55, no. 3, pp. 571–597, 2018.
- [17] E. Serra, A. Shrestha, F. Spezzano, and A. Squicciarini, "DeepTrust: an automatic framework to detect trustworthy users in opinion-based systems," in *Proceedings of the Tenth ACM Conference On Data And Application Security And Privacy*, pp. 29–38, New Orleans, LA, USA, March 2020.
- [18] Y. Dou, G. Ma, P. S. Yu, and S. Xie, "Robust spammer detection by nash reinforcement learning," in *Proceedings of the 26th ACM SIGKDD Conference On Knowledge Discovery And Data Mining*, pp. 924–933, California, CA, USA, June 2020.
- [19] Z. Guo, L. Tang, T. Guo, K. Yu, M. Alazab, and A. Shalaginov, "Deep Graph neural network-based spammer detection under the perspective of heterogeneous cyberspace," *Future Generation Computer Systems*, vol. 117, pp. 205–218, 2021.
- [20] Z. Song, F. Bai, J. Zhao, and J. Zhang, "Spammer detection using graph-level classification model of graph neural network," in *Proceedings of the IEEE 2nd International Conference On Big Data, Artificial Intelligence And Internet Of Things Engineering*, pp. 531–538, Nanchang, China, March 2021.
- [21] J. Cao, R. Xia, Y. Guo, and Z. Ma, "Collusion-aware detection of review spammers in location based social networks," *World Wide Web*, vol. 22, no. 6, pp. 2921–2951, 2019.
- [22] F. Zhang, X. Hao, J. Chao, and S. Yuan, "Label propagation-based approach for detecting review spammer groups on e-commerce websites," *Knowledge-Based Systems*, vol. 193, pp. 1–19, 2020.
- [23] Y. Sun, J. Han, C. C. Aggarwal, and N. V. Chawla, "When will it happen?-relationship prediction in heterogeneous information networks," in *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, pp. 663–672, Seattle, WA, USA, February 2012.
- [24] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "PathSim," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.
- [25] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li, "Meta structure: computing relevance in large heterogeneous information networks," in *Proceedings of the 22nd ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*, pp. 1595–1604, San Francisco, CA, USA, August 2016.
- [26] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, New York, NY, USA, August 2014.
- [27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the Advances In Neural Information Processing Systems*, pp. 3111–3119, Lake Tahoe, NV, USA, December 2013.
- [28] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 169–178, Boston, MA, USA, August 2000.
- [29] X. Liu, K. Shi, and Z. Yao, "Evaluation of the M&A effectiveness based on grey systems theory and entropy method," in *Proceedings of the 4th IEEE International Conference On Information Science And Technology*, pp. 268–271, Shenzhen, China, April 2014.
- [30] M. Song, Q. Zhu, J. Peng, E. D. R. Santibanez Gonzalez, and D. R. Ernesto, "Improving the evaluation of cross efficiencies:

- a method based on Shannon entropy weight,” *Computers & Industrial Engineering*, vol. 112, pp. 99–106, 2017.
- [31] W. Li, M. Gao, H. Li, J. Zeng, Q. Xiong, and S. Hirokawa, “Shilling attack detection in recommender systems via selecting patterns analysis,” *IEICE - Transactions on Info and Systems*, vol. E99.D, no. 10, pp. 2600–2611, 2016.
- [32] R. Barbado, O. Araque, and C. A. Iglesias, “A framework for fake review detection in online consumer electronics retailers,” *Information Processing & Management*, vol. 56, no. 4, pp. 1234–1244, 2019.
- [33] Y. Zhang, Y. Tan, M. Zhang, Y. Liu, T. Chua, and S. Ma, “Catch the black sheep: unified framework for shilling attack detection based on fraudulent action propagation,” in *Proceedings of the 24th International Joint Conference On Artificial Intelligence*, pp. 2408–2414, Buenos Aires, Argentina, July 2015.
- [34] L. Zhang, G. He, J. Cao, H. Zhu, and B. Xu, “Spotting review spammer groups: a cosine pattern and network based method,” *Concurrency and Computation: Practice and Experience*, vol. 30, no. 20, pp. 1–15, 2018.

Research Article

Outsourced Mutual Private Set Intersection Protocol for Edge-Assisted IoT

Jing Zhang ¹, Rongxia Qin ¹, Ruijie Mu ¹, Xiaojun Wang ², and Yongli Tang ¹

¹College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454003, China

²Network Innovations Centre (NIC), Dublin City University, Dublin 9, Ireland

Correspondence should be addressed to Yongli Tang; yltang@hpu.edu.cn

Received 30 August 2021; Revised 17 October 2021; Accepted 9 December 2021; Published 31 December 2021

Academic Editor: Shifeng Sun

Copyright © 2021 Jing Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The development of edge computing and Internet of Things technology has brought convenience to our lives, but the sensitive and private data collected are also more vulnerable to attack. Aiming at the data privacy security problem of edge-assisted Internet of Things, an outsourced mutual Private Set Intersection protocol is proposed. The protocol uses the ElGamal threshold encryption algorithm to rerandomize the encrypted elements to ensure all the set elements are calculated in the form of ciphertext. After that, the protocol maps the set elements to the corresponding hash bin under the execution of two hash functions and calculates the intersection in a bin-to-bin manner, reducing the number of comparisons of the set elements. In addition, the introduction of edge servers reduces the computational burden of participating users and achieves the fairness of the protocol. Finally, the IND-CPA security of the protocol is proved, and the performance of the protocol is compared with other relevant schemes. The evaluation results show that this protocol is superior to other related protocols in terms of lower computational overhead.

1. Introduction

The vigorous development of fifth-generation technology (5G), Internet of Things (IoT), edge computing, and other technologies has spawned new medical and life modes such as smart medical treatment [1], smart home [2], and smart bus [3]. Intelligent IoT devices have been widely used in daily life and have brought great changes to people's life. With the assistance of proxy devices and edge servers [4], these data can be outsourced to edge storage for subsequent analysis and use. However, although data outsourcing based on edge computing reduces the storage and computing overhead on the user side, it also exposes the user's sensitive data to the risk of leakage [5]. How to protect the privacy of data stored on edge servers and share data with designated data consumers (such as service and product providers, medical professionals, and educators) has become the focus of research.

Private Set Intersection (PSI), as an efficient encryption technology that allows secret sharing of data information, can ensure the security of the data stored on the edge server

when making full use of the data for intersection calculation. So it has become an important research object to solve the problem of edge-assisted Internet of Things data privacy sharing. PSI protocol [6] refers to the intersection of two participants calculating their private sets. In the edge computing environment, two clients encrypt their respective data sets and outsource them to the edge server. Then the edge server effectively performs the intersection operation but cannot know any information in the set. Then one or two clients can obtain the intersection result, while their respective sets remain private. Usually, only one client obtains the intersection result is the one-way PSI protocol [7], which can be applied in situations such as contact tracing of the novel coronavirus COVID-19 [8]. At this time, the client unilaterally obtains the intersection result to determine whether it belongs to the contact. However, one-way PSI protocol cannot guarantee that both clients obtain the intersection result at the same time. Obviously, one-way PSI cannot satisfy both clients in the situation such as profile matching [9], in which both clients want to obtain the intersection results to realize the medical information sharing

between patients. In this case, mutual PSI (mPSI) protocol [10] is the better choice. That is our focus.

Debnath et al. [11] proposed a secure mutual oblivious pseudorandom function (mOPRF) under the malicious model based on composite order group to maintain the fairness of the mPSI protocol and use homomorphic encryption algorithm to protect data privacy. However, since the protocol is constructed based on composite order, the efficiency is lower than that based on prime order. Based on [11], Debnath et al. [12] proposed another mPSI protocol using prime order groups. The protocol also uses homomorphic encryption algorithm to ensure the security of data and uses semihonest offline arbiter to achieve fairness between two participants. However, the complexity of the protocol is still high. The mPSI protocol in [13] is also constructed based on prime order. It uses the multiplicative homomorphic encryption ElGamal and the distributed ElGamal cryptosystem to ensure the security of data and the offline semihonest arbiter to achieve fairness. But in order to ensure the security under the malicious model, the verifiable Cramer-Shoup cryptographic system used makes the protocol more complicated. Overall, although these mPSI protocols achieve data privacy and fairness, they have high computational complexity and low efficiency.

In this paper, we propose an outsourced mPSI (O-mPSI) protocol in the aid of the edge server. The protocol not only protects the data privacy of parties and achieves the fairness of the results obtained by both parties, but also improves the efficiency. The main contributions are as follows:

- (1) The O-mPSI protocol improves the method of preprocessing set elements in [14] by increasing the number of elements stored in each hash bin instead of using the stash. It only needs to compare the elements in the two hash tables in a bin-to-bin manner to calculate the intersection, which further reduces the number of comparisons of set elements and decreases the computational cost of the protocol. As a result, the computation and communication overhead of this protocol is lower than the existing mPSI protocol and realizes the efficiency of the O-mPSI protocol.
- (2) The protocol adopts ElGamal threshold encryption algorithm to encrypt the elements in the hash table to ensure that the set elements of both parties are compared in the form of ciphertext, so that the users can correctly and safely calculate the intersection and realize the data privacy of the O-mPSI protocol.
- (3) The protocol introduces a semihonest edge server as the third party, and the work of set element comparison is transferred to the server, which further reduces the computational burden of users. At the same time, it enables two users to process collection elements in parallel. This ensures that, after the implementation of the agreement, both parties can know the intersection results at the same time and realize the fairness of O-mPSI protocol.

2. Related Work

Since Agrawal et al. [15] proposed the concept of PSI, a series of work has been done to construct the PSI protocol. Ordinarily, they are divided into one-way PSI and mutual PSI.

2.1. One-Way PSI. At present, there are many types of research on the one-way PSI protocol, and the design methods are mainly based on public key encryption, garbled circuit (GC), oblivious transfer (OT), and cloud computing. Based on the design method of public key encryption system, literature [16] uses Fully Homomorphic Encryption (FHE) to construct the PSI protocol. The protocol uses bloom filters (BF) to process data, so that the complexity of the protocol has nothing to do with the size of the client set. Huang and Evans designed a PSI protocol [17] through GC that can resist semihonest adversaries, which proved to be suitable for the intersection calculation of sets of different sizes. Pinkas et al. realized the PSI protocol [18] with linear communication complexity based on GC and oblivious programmable pseudorandom functions (OPPRF). This protocol is superior to previous circuit-based PSI protocols in terms of efficiency. Literature [14] constructed oblivious pseudorandom function (OPRF) based on OT extension and then proposed a PSI protocol combined with OPRF and hash algorithm, and the security is proved under the semihonest opponent model. Kavousi et al. proposed a PSI protocol in [19], which takes the OPRF and the garbled BF as its main components, avoiding costly operations of computation and having high scalability. The protocol in [20] allows users to store their private data sets on cloud server and also entrust computing of the intersection to the server and use homomorphic encryption (HE) and oblivious polynomial evaluation (OPE) to process the data. It greatly reduces the workload of users and improves the computational efficiency of the protocol. The PSI protocol designed by Abadi et al. [21] also uses cloud storage of private data sets. It is mainly constructed by hash table and OPE without any public key encryption operation. However, all parties need to establish a secure channel in advance; otherwise it is easy for an attacker to eavesdrop on the communication between honest parties. Literature [22] proposed an improved PSI scheme based on [21]. There is no need for any secure channel in the scheme, which is superior to the scheme in [21] in terms of confidentiality and complexity.

2.2. Mutual PSI. The research on mutual PSI protocol began in 2005; Kissner and Song et al. proposed the first mPSI protocol [23], which is based on the mathematical properties of OPE and uses HE to calculate PSI on ciphertexts. The fairness of the protocol depends on the fairness of the threshold encryption scheme used by the protocol. Camenisch and Zaverucha proposed another mPSI authentication set protocol [24] based on Camenisch-Lysyanskaya signature (CLS) and OPE. The disadvantage of this protocol is that its computational overhead is quadratic and the computational complexity is relatively high. Fairness of the protocol in [24] is realized by a fair exchange scheme. However, if the

input is not authenticated, it usually does not work. Kim et al. coupled the prime representation (PR) technology with threshold additive HE and realized an mPSI protocol [25] with linear complexity for the first time in the semihonest security model and through the nature of threshold decryption to achieve the fairness of the protocol. Dong and Chen et al. proposed a fair mPSI protocol [26] with a semihonest arbitrator based on HE and OPE. The arbitrator in the protocol can handle conflicts and cannot know the user's private input and output. The mPSI protocol in [11] is constructed by HE and mutual oblivious pseudorandom function (mOPRF). The fairness of the mOPRF ensures the fairness of the protocol. And in the standard model, it has been proven to resist the attack of malicious adversaries. In [12], an mPSI protocol with linear communication and computational overhead is constructed by using prime order group. The protocol uses a distributed ElGamal encryption algorithm and an offline semitrusted third party to achieve the fairness of the protocol and the security under the malicious adversary model. The mPSI protocol in [13] uses multiplicative HE and a distributed ElGamal cryptosystem to protect data privacy and uses an offline semihonest arbiter to achieve fairness. Under the decisional Diffie-Hellman hypothesis, it is proved that the protocol is safe under the malicious adversary model.

We show the differences between these protocols in Table 1.

3. Preliminaries

3.1. Decisional Diffie-Hellman Assumption

Definition 1. Let G be a cyclic group of prime order p , g is the generator of G , and $a, b \leftarrow_R \mathbb{Z}_p$. For the following two four-tuple distributions: $R = (g, g^a, g^b, g^c) \in G^4$, $D = (g, g^a, g^b, g^{ab}) \in G^4$, and for any probability polynomial time (PPT) adversary A when distinguishing between R and D , its advantage $\text{Adv}_{\mathcal{A}}^{\text{DDH}}(\lambda) = |\Pr[\mathcal{A}(R) = 1] - \Pr[\mathcal{A}(D) = 1]|$ is negligible with security parameter λ , where R is a random four-tuple and D is a Diffie-Hellman four-tuple.

3.2. Security Model

3.2.1. Adversary Model. We consider the environment where a PPT adversary A exists; the definition and model follow [27]. In this setting, the adversary A can eavesdrop on the messages transmitted through the communication channel and allow A to corrupt participant to obtain private key. The capabilities of the adversary A are shown in Table 2.

3.2.2. Formal Security Model. We mainly describe the formal security model of the O-mPSI protocol in this section. It is based on the model in [28] and we made some modifications according to the specific requirements of the O-mPSI protocol.

Initialization. In a two-party set intersection scheme O-mPSI, three entities are involved, that is, two participants

$P_1, P_2 \in \text{User}$ and an edge server $S \in \text{Server}$. Each of them may have several instances called oracle, which involve different, concurrent executions of O-mPSI. We denote user instances as $P_1^i, P_2^i (i \in \mathbb{Z})$, server instances as S^i , and any kind of instance as $U \in \text{User} \cup \text{Server}$. In addition, P_1 holds key pair (pk_1, sk_1) , P_2 has key pair (pk_2, sk_2) , and pk is the system public key.

Queries. The adversary A only interacts with the participants of the protocol through oracle queries, which simulates the capabilities of adversary A in a real attack. And all possible oracle queries are shown below:

- (1) Execute (P_1^i, P_2^i, S^i) : The adversary's eavesdropping attack is simulated in this oracle query (see C1 in Table 2), and the messages exchanged during the actual implementation of the O-mPSI protocol are returned.
- (2) Send (U, m) : A sends a message to instance U in this query; then the response generated by U processing m based on O-mPSI protocol is returned.
- (3) Corrupt (U, a) : The corruption ability of the adversary is simulated in this query (see C2 in Table 2). A can only corrupt one of the two users, not both. If $a = 1$, it returns the private key sk_1 and data set X of P_1 . If $a = 2$, it returns the private key sk_2 and data set Y of P_2 .
- (4) Collude (S) : The collusion ability of adversary A is simulated in this query (i.e., C3 in Table 2). Any information stored on the server is returned.
- (5) Reveal (U) : The system private key shared by the instances P_1^i and P_2^i is returned to the adversary in this query. But instances P_1^i, P_2^i and their partner S^i must not have been queried by corrupt-query and test-query; otherwise it returns \perp .
- (6) Challenge (U, m_0, m_1) : In this query, A selects two messages m_0 and m_1 of equal length and sends them to instance U . The instance U selects $b \leftarrow_R \{0, 1\}$ randomly, then encrypts m_b , and returns the ciphertext c . Among them, P_1^i, P_2^i , and S^i in U have not been inquired by corrupt-query and reveal-query; otherwise it returns \perp . And this query is called only once during execution.
- (7) Test (U) : After querying the oracle, if the guessed bit $b' = b$, return 1; otherwise return 0. This query can only point to the instance in the challenge-query and is called only once during execution.

IND-CPA security. During the execution of the O-mPSI protocol, A can require polynomial degree to execute, send, corrupt, collude, and reveal queries. A can also send a challenge-query and a test-query to an instance that has not been queried. At the end of the game execution, for the bit b in the challenge-query, A outputs a guess bit b' in the test-query. If $b' = b$, it means that A wins and it is recorded as Succ. The advantage that the adversary A can destroy the IND-CPA security of O-mPSI protocol is defined as

TABLE 1: Difference between the mentioned protocols.

Type	Reference	Structure	Adversary model	Fairness
Public key	[12]	HE	Malicious	Yes
	[13]	HE	Malicious	Yes
	[16]	FHE + BF	Semihonest	No
	[23]	HE + OPE	Malicious	Yes
	[24]	HE + OPE + CLS	Malicious	Yes
	[25]	HE + PR	Semihonest	Yes
	[26]	HE + OPE	Malicious	Yes
Garbled circuit	[17]	GC	Semihonest	No
	[18]	GC + OPPRF	Semihonest	No
Oblivious transfer	[11]	mOPRF + HE	Malicious	Yes
	[14]	hash + OPRF	Semihonest	No
	[19]	OPRF + garbled BF	Semihonest	No
Cloud computing	[20]	HE + OPE	Semihonest	No
	[21]	Hash + OPE	Semihonest	No
	[22]	Hash + OPE	Semihonest	No

TABLE 2: Capabilities of the adversary.

Types	Descriptions
C1	A can learn the messages transmitted in the communication channel.
C2	A can corrupt one of the two participants to obtain its secret data and private key.
C3	A can collude with dishonest edge server S to obtain the information stored in the server.

$$\text{Adv}_{O\text{-mPSI}}^{\text{IND-CPA}}(\mathcal{A}) = \Pr[\text{Succ}(\mathcal{A})] - \frac{1}{2} = \Pr[b' = b] - \frac{1}{2}. \quad (1)$$

Definition 2. For any PPT adversary A , if there is a negligible function $\varepsilon(\cdot)$ such that $\text{Adv}_{O\text{-mPSI}}^{\text{IND-CPA}}(\mathcal{A})\Pr[\text{Succ}(\mathcal{A})] - 1/2 \leq \varepsilon(\lambda)$, then the O-mPSI protocol is IND-CPA secure.

3.3. Hash Algorithm. Cuckoo hashing. Cuckoo hashing [29] is a method to solve hash conflict, which is widely used in PSI protocols. In this hashing technique, two hash functions $h_1, h_2: \{0, 1\}^* \rightarrow \{1, \dots, m\}$ are used to map n elements into m bins, where each bin has at most b elements. When storing an element x in the cuckoo hash table, calculate the two bin positions $h_1(x)$ and $h_2(x)$ corresponding to x . If both bins are not full, insert x into either of them; if one of the two bins is full and the other is not full, then insert x into the bin that is not full; if both bins are full, then randomly select a position to kick out one of the elements y and then insert x into it; the kicked y is reinserted into the cuckoo table using the same algorithm. This process is called relocation, and the relocation process is executed recursively until all elements are stored in the cuckoo hash table.

Simple hashing. This hashing technique is similar to cuckoo hashing. Two hash functions $h_1, h_2: \{0, 1\}^* \rightarrow \{1, \dots, m\}$ are used to map n elements into m bins, and each bin also has at most b elements. However, in simple hash mapping, when the element x is mapped to its corresponding two positions $h_1(x)$ and $h_2(x)$, the elements are stored in both $h_1(x)$ and $h_2(x)$.

3.4. ElGamal Threshold Encryption Algorithm. The ElGamal threshold encryption algorithm [30] is implemented according to the additive homomorphism of the ElGamal encryption algorithm, which is composed of *KeyGen*, *Encrypt*, *Decrypt*, and *Rerandomize* four algorithms. The specific algorithm is as follows:

KeyGen. Given a cyclic group G of order p and its generator g , the security parameter is λ . Parties P_1 and P_2 randomly select $sk_i \leftarrow \mathbb{Z}_p$ ($i = 1, 2$) and then compute $pk_i = g^{sk_i} \bmod p$, where sk_i is their respective private key, and pk_i is their respective public key. Then the public key of the threshold encryption scheme is $pk = pk_1 \cdot pk_2$.

Encrypt. For a message m , map m to $H(m)$ using the hash function $H(\cdot)$, where $H(\cdot)$ is defined as $\{0, 1\}^* \rightarrow G$. Then select a random number r_1 , and compute the ciphertext of the message m as $ct = (g^m \cdot pk^{r_1} \bmod p)$.

Decrypt. For a ciphertext ct , P_1 half-decrypts it as $ct' = (g^m \cdot pk^{r_1} / (g^{r_1})^{sk_1} \bmod p)$; then P_2 fully decrypts ct' as $g^m = ct' / (g^{r_1})^{sk_2} \bmod p$.

Rerandomize. For the given ciphertext ct , select a random number r_2 to rerandomize it to $ct^* = (g^m \cdot pk^{r_1} \cdot pk^{r_2}) = (g^m \cdot pk^{r_1+r_2})$.

4. Outsourced Mutual PSI Protocol

4.1. Overview. The system model of O-mPSI protocol is shown in Figure 1. There are three entities involved, two participating users P_1, P_2 and a semihonest edge server S . Among them, P_1 has private data set $X = \{x_1, \dots, x_{n_1}\}$ and user P_2 has private data set $Y = \{y_1, \dots, y_{n_2}\}$. They hope to calculate the intersection $X \cap Y$ of X and Y through the server S without revealing their own set information.

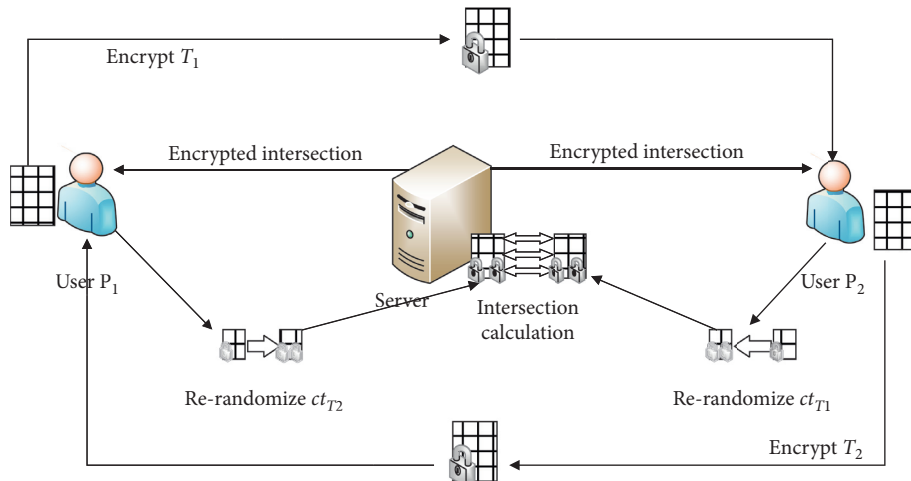


FIGURE 1: System model of O-mPSI protocol.

The design idea of our protocol is based on [14], where P_1 and P_2 select three hash functions $\{h_1, h_2, h_3\}: \{0, 1\}^* \rightarrow \{1, \dots, m\}$ to generate three hash bin positions corresponding to the set elements in the hash table. P_1 uses the cuckoo hashing to select one of the three bins to map each element in the set X to hash table T_1 , each bin stores one element, and the remaining elements are stored in the stash ST . P_2 uses simple hashing to map each element in set Y to the corresponding three bins in hash table T_2 , and each bin stores b elements. Then, the elements in the hash bin of T_1 are compared with the elements in the corresponding hash bin of T_2 . And each element in the stash ST is compared to all elements in set Y . In this way, the comparison times of elements are reduced from $O(n^2)$ to $O(n)$, and the intersection $X \cap Y$ is all the equal elements obtained by comparison.

But we diverge from the protocol of [14] in the method. In our protocol, parties P_1 and P_2 agree on two hash functions $h_1, h_2: \{0, 1\}^* \rightarrow \{1, \dots, m\}$ to generate the two hash bin positions corresponding to the set elements in the hash table. In this way, the elements in the set Y of P_2 only need to be stored twice, which saves storage space and reduces the subsequent computational burden. Moreover, our protocol sets the size of each bin of hash table T_1 to b . This avoids that the elements cannot be stored in the hash table T_1 due to too many relocations when hash conflicts occur, so there is no need to increase the stash to store the elements. Therefore, it is only necessary to compare the elements in the bin of T_1 with the elements in the corresponding bin of T_2 to obtain the intersection. This further decreases the number of comparisons and reduces the complexity of the protocol.

However, there is still a problem of how to protect the privacy of user set information while calculating the intersection correctly. We use the ElGamal threshold encryption algorithm to solve this problem, as shown in Section 4.2.1. Participant P_a ($a = 1, 2$) uses the public key pk to encrypt the elements in the hash table T_a and then sends it to the other party to rerandomize. Combined with the hash algorithm, we can see that, for $\forall x_i \in X$ in the hash bin of T_1 , if the same element $y_j \in Y$ in T_2 is stored in the

corresponding hash bin, there must be $E^*(x_i) = E^*(y_j)$ in the two hash bins with the same bin number. Therefore, the use of ElGamal threshold encryption algorithm enables the comparison operation of elements to be performed in the form of ciphertext and ensures that the encrypted results of equal elements on both sides are the same. It creates conditions for the smooth implementation of intersection calculation.

In addition, in order to decrease the computational burden of both parties, the element comparison work is handed over to the edge server. Specifically, P_a ($a = 1, 2$) sends the hash table encrypted by the ElGamal encryption algorithm to the server. Then the server compares the elements in the two hash tables in a bin-to-bin manner in the form of ciphertext. That is, it compares the elements in the hash bin of T_1 with the elements in the corresponding hash bin of T_2 . In this way, all the equal ciphertext elements in the m bins are the intersection $X \cap Y$ in the ciphertext form. Then P_1 and P_2 cooperate to decrypt the intersection in the form of ciphertext in parallel, and the plaintext intersection $X \cap Y$ can be obtained at the same time. Among them, the comparison work of m bins can be performed in parallel, which reduces the time overhead of O-mPSI protocol.

4.2. O-mPSI Protocol. In this section, we introduce the proposed O-mPSI protocol. For easier understanding, we divide the protocol into two parts: the data encryption part and the set intersection computation part, which are introduced in the following two subsections, respectively.

4.2.1. ElGamal Threshold Encryption Protocol. In this section, we describe how ElGamal threshold encryption algorithm works in the protocol, and the details are shown in Algorithm 1.

Remark 1. In the key generation phase, parties P_1 and P_2 jointly generate the public key pk so that both parties can use the same public key for encryption in the subsequent encryption phase.

Inputs:

P_1 : The cuckoo hash table T_1

P_2 : The hash table T_2

Output: the encrypted cuckoo hash table $ct_{T_1}^*$ and $ct_{T_2}^*$.

Key generation phase

P_1 and P_2 calculate as below:

1. Determine a hash function $H: \{0, 1\}^* \rightarrow G$.
2. Run the *KeyGen* algorithm in ElGamal threshold encryption to generate the key pairs (pk_i, sk_i) , $i = 1, 2$.
3. Publish the public keys pk_i ($i = 1, 2$) and keep the private key sk_i ($i = 1, 2$).
4. Each party computes $pk = pk_1 \cdot pk_2$.

Encryption phase

1. P_1 encrypts the cuckoo hash table T_1 by bins, for all $k \in [m]$, P_1 computes as follows:
 - (1) choose a random number r_1^k , for all items x_i^k ($i = 1, \dots, b$) in the k^{th} cuckoo hash table T_1^k , using ElGamal threshold encryption algorithm to encrypt them: $ct_{T_1^k} = E(x_i^k) = (g^{r_1^k}, g^{H(x_i^k)} \cdot pk_1^{r_1^k})$
 - (2) sends $ct_{T_1} = (ct_{T_1^1}, \dots, ct_{T_1^m})$ to P_2 in shuffled order.
2. P_2 encrypts the hash table T_2 by bins, for all $k \in [m]$, P_2 computes as follows:
 - (1) choose a random number r_2^k , for all items y_i^k ($i = 1, \dots, b$) in the k^{th} hash table T_2^k , using ElGamal threshold encryption algorithm to encrypt them: $ct_{T_2^k} = E(y_i^k) = (g^{r_2^k}, g^{H(y_i^k)} \cdot pk_2^{r_2^k})$
 - (2) sends $ct_{T_2} = (ct_{T_2^1}, \dots, ct_{T_2^m})$ to P_1 in shuffled order.
3. P_1 re-randomizes the received ct_{T_2} by bins. For all $k \in [m]$ and $\forall i \in [b]$, P_1 computes as follows:

$$ct_{T_2^k}^* = E^*(y_i^k) = (g^{r_1^k + r_2^k}, g^{H(y_i^k)} \cdot pk_1^{r_1^k + r_2^k}).$$
 Then $ct_{T_2}^* = (ct_{T_2^1}^*, \dots, ct_{T_2^m}^*)$.
4. P_2 re-randomizes the received ct_{T_1} by bins. For all $k \in [m]$ and $\forall i \in [b]$, P_2 computes as follows:

$$ct_{T_1^k}^* = E^*(x_i^k) = (g^{r_1^k + r_2^k}, g^{H(x_i^k)} \cdot pk_2^{r_1^k + r_2^k})$$
 Then $ct_{T_1}^* = (ct_{T_1^1}^*, \dots, ct_{T_1^m}^*)$.

ALGORITHM 1: ElGamal threshold encryption protocol.

Remark 2. In the encryption phase, after encrypting the hash table, each party P_i ($i = 1, 2$) needs to send ct_{T_i} to the other party for rerandomization. Because each party privately chose a random number for encryption, two equal elements are encrypted differently by the two parties. In addition, to calculate the set intersection correctly, adding the rerandomization operation performed by the other party to make the encryption results of equal elements the same is necessary.

4.2.2. Outsourced Two-Party Set Intersection Protocol. In this section, we will explain the intersection calculation part of the O-mPSI protocol. Detailed description can be seen in Algorithm 2.

Remark 3. In the data storage phase, P_1 hashes all its elements into one of its two corresponding bins, and each element is stored in only one bin. P_2 hashes all its elements into both of its two corresponding bins, and each element is stored in both bins. In this case, suppose that there is $x_i = y_j$ ($x_i \in X$ and $y_j \in Y$); the two bin positions $h_1(x_i)$ and $h_2(x_i)$ corresponding to x are the same as the two bin positions $h_1(y_j)$ and $h_2(y_j)$ corresponding to y . Then no matter which bin ($h_1(x_i)$ or $h_2(x_i)$) x is stored in, the corresponding element y can be found in the two bins $h_1(y_j)$ and $h_2(y_j)$. Therefore, it can be known that storing all elements in set Y in both bins can avoid missing intersection elements when computing the set intersection.

Remark 4. In the data encryption phase, parties P_1 and P_2 use the ElGamal threshold encryption algorithm in Section 4.2.1 to encrypt the elements in their hash table and at the same time add a permutation sequence, so that both parties cannot know the specific location of the element. After hash tables T_1 and T_2 undergo the same permutation process, their bin numbers still correspond, so the intersection can be calculated correctly by the bins.

Remark 5. In the intersection calculation phase, if the server continues to calculate on the encrypted intersection according to the addition homomorphism of ElGamal homomorphism encryption algorithm, the encrypted intersection sum can be obtained. Then continuing step 2, the plaintext intersection sum will be obtained by both P_1 and P_2 .

5. Security Analysis

The security proof of the O-mPSI protocol based on decisional Diffie-Hellman (DDH) assumption is shown in this section.

Theorem 1. *Let G be a represented group of order p . Let A be a PPT adversary against the IND-CPA security within a time limit t , and A can send q_{send} send-queries, q_{exe} execute-queries, and q_h random oracle queries at most. Then we can get*

Inputs:

P_1 : Set $X = \{x_1, \dots, x_{n_1}\}$
 P_2 : Set $Y = \{y_1, \dots, y_{n_2}\}$

Output: The intersection set I , where $I = X \cap Y$.

Data storage phase:

1. P_1 and P_2 determine the number m of bins, the size b of bins and the two hash functions $h_1, h_2: \{0, 1\}^* \rightarrow \{1, \dots, m\}$.
2. For $x_i \in X, i \in [n_1]$, P_1 computes the two bin positions $h_1(x_i)$ and $h_2(x_i)$ corresponding to x_i , and inserts x_i into one of them. P_1 fills the bin with less than b elements with dummy elements, then generates the cuckoo hash table T_1 .
3. For $y_j \in Y, j \in [n_2]$, P_2 computes the bin positions $h_1(y_j)$ and $h_2(y_j)$ corresponding to y_j , and inserts y_j into both of them. P_2 fills the bin with less than b elements with dummy elements, then generates the hash table T_2 .

Data encryption phase:

1. P_1 calls the ElGamal threshold encryption protocol to compute the encrypted hash table $ct_{T_2}^*$, and then sends $ct_{T_2}^*$ to the server in shuffled order.
2. P_2 calls the ElGamal threshold encryption protocol to compute the encrypted cuckoo hash table $ct_{T_1}^*$, and then sends $ct_{T_1}^*$ to the server in shuffled order.

Intersection calculation phase:

1. After the $ct_{T_1}^*$ and $ct_{T_2}^*$ are received, the server proceeds as follows:
 - (1) computes the encrypted set intersection by bins: for $k \in [m]$, computes $ct_{T_k} = ct_{T_1^k}^* \cap ct_{T_2^k}^*$.
 - (2) then the final encrypted intersection $ct_I = ct_{T_1} \cup ct_{T_2} \cup \dots \cup ct_{T_m}$.
 - (3) publishing $ct_{I=X \cap Y}$.
 2. After receiving ct_I , the users P_1, P_2 and the server proceed as follows:
 - (1) each party $P_i (i = 1, 2)$ half-decrypts ct_I to ct_{I_i}' using its private key, and then sends ct_{I_i}' to the server.
 - (2) then the server sends ct_{I_2}' to P_1 and sends ct_{I_1}' to P_2 .
- Each party fully decrypts the received other party half-decrypts intersection, and gets the plaintext intersection set $I = \{z_1, \dots, z_w\}$.

ALGORITHM 2: Outsourced two-party set intersection protocol.

$$\text{Adv}_{O\text{-}m\text{PSI}}^{\text{IND-CPA}}(\mathcal{A}) \leq q_h \text{Adv}_{\mathcal{A}}^{\text{DDH}} \left(\left(t + \frac{1/3 \cdot q_{\text{send}} + q_{\text{exe}} + 1}{p^2} \right) + \frac{q_h^2 + (1/3 \cdot q_{\text{send}} + q_{\text{exe}})^2}{2p} \right). \quad (2)$$

Proof. Let A be the adversary against the IND-CPA security of O-mPSI protocol. Then construct PPT adversaries to attack the DDH assumption through A . If A can break the IND-CPA security, then at least one PPT adversary has successfully broken the DDH assumption. We utilize hybrid games to prove Theorem 1. The game starts from the real attack and ends when the adversary does not have any advantage. An event Succ_n is defined for each game $G_n (0 \leq n \leq 4)$, which indicates that A guesses the bit b in the test-query correctly.

- (1) Game G_0 : The game G_0 corresponds to a real attack in the random oracle model. According to Definition 2, we can get

$$\text{Adv}_{O\text{-}m\text{PSI}}^{\text{IND-CPA}}(\mathcal{A}) = \Pr[\text{Succ}_0] - \frac{1}{2}. \quad (3)$$

- (2) Game G_1 : We simulate the random oracle $H: \{0, 1\}^* \rightarrow G$ (and there is also a random oracle H' that will appear in G_3) in G_1 by maintaining hash list L (and L') as usual. Besides, Send, Execute, Corrupt, Collude, Reveal, Challenge, and Test oracles will be simulated as in the real attack (see Table 3). It is easy to know that G_1 is completely indistinguishable from the real attack, so that

$$\Pr[\text{Succ}_1] - \Pr[\text{Succ}_0] = 0. \quad (4)$$

- (3) Game G_2 : Like G_1 , we simulate all oracles in G_2 , except for games where some collisions occur: $H(x)$ and $H(y)$. Because x or y is simulated, they are randomly and uniformly selected. Therefore, from the birthday paradox, we know that

$$|\Pr[\text{Succ}_2] - \Pr[\text{Succ}_1]| \leq \frac{q_h^2 + (1/3 \cdot q_{\text{send}} + q_{\text{exe}})^2}{2p}. \quad (5)$$

- (4) Game G_3 : In G_3 , we use the private oracles H' instead of the oracle H to calculate $s(ct_{T_1})$ and $s(ct_{T_2})$, which are completely independent of ct_{T_1} and ct_{T_2} . The games G_3 and G_2 are indistinguishable unless the event $\text{Ask}H_3$ occurs: A queries the hash function H for ct_{T_1} or ct_{T_2} . In addition, no matter what bit b in challenge-query is, the answer is random. Therefore, it can be seen that

$$\Pr[\text{Succ}_3] - \Pr[\text{Succ}_2] \leq \Pr[\text{Ask}H_3] \quad (6)$$

$$\Pr[\text{Succ}_3] = \frac{1}{2}$$

- (5) Game G_4 : In G_4 , we simulate the executions through the random self-reducibility of the Diffie-Hellman problem, and given one DDH instance $(A = g^x, B = g^y)$, we randomly select $\alpha, \beta \in \mathbb{Z}_p$ and

TABLE 3: Simulation of the queries in O-mPSI protocol.

Number	Queries
(1)	Hash $H(m)$ (or $H'(m)$): If there is a record (m, r) existing in list L , return r . Otherwise, select a random number $r \in G$ to return, and add the record (m, r) to the list L (or list L').
(2)	Send (P^i, a, start) : For each of the m hash bins, if $a = 1$, choose a random number r_1^k ($k \in [m]$), compute $E(x^k) = (g^{H(x^k)} \cdot pk^{r_1^k})$, and then return $ct_{T_1} = (ct_{T_1^1}, \dots, ct_{T_1^m})$; if $a = 2$, choose a random number r_2^k , compute $E(y^k) = (g^{H(y^k)} \cdot pk^{r_2^k})$, and then return $ct_{T_2} = (ct_{T_2^1}, \dots, ct_{T_2^m})$.
(3)	Send (P^i, a, ct_{T_a}) : For $k \in [m]$, if $a = 1$, compute $E^*(y^k) = (g^{H(y^k)} \cdot pk^{r_1^k+r_2^k})$, and then return $ct_{T_2}^* = (ct_{T_2^1}^*, \dots, ct_{T_2^m}^*)$; if $a = 2$, compute $E^*(x^k) = (g^{H(x^k)} \cdot pk^{r_1^k+r_2^k})$, and then return $ct_{T_1}^* = (ct_{T_1^1}^*, \dots, ct_{T_1^m}^*)$.
(4)	Send $(S^i, (ct_{T_1}^*, ct_{T_2}^*))$: For each of the m hash bins, compute $ct_{T_k} = ct_{T_k^*} \cap ct_{T_k^*}$ and $ct_I = ct_{T_1} \cup ct_{T_2} \cup \dots \cup ct_{T_m}$. Then return ct_I .
(5)	Execute (P_1^i, P_2^i, S^i) : According to the successive simulations of the send-queries, return $ct_{T_1} \leftarrow \text{send-query}(P_1^i, \text{start})$, $ct_{T_2} \leftarrow \text{send-query}(P_2^i, \text{start})$, $ct_{T_2}^* \leftarrow \text{send-query}(P_1^i, ct_{T_2})$, $ct_{T_1}^* \leftarrow \text{send-query}(P_2^i, ct_{T_1})$, and $ct_I \leftarrow \text{send-query}(S^i, (ct_{T_1}^*, ct_{T_2}^*))$.
(6)	Corrupt (U^i, a) : Return sk_1 and X if $a = 1$; return sk_2 and Y if $a = 2$.
(7)	Collude (S^i) : Return $ct_{T_1}^*$, $ct_{T_2}^*$, and ct_I of S^i .
(8)	Reveal (U^i) : Compute $sk = sk_1 + sk_2$ and return the private key sk of P_1^i and P_2^i .
(9)	Challenge (U^i, m_0, m_1) : The instance U^i randomly selects $b \leftarrow_R \{0, 1\}$, then computes $E(m_b) = (g^r, g^{H(m_b)} \cdot pk^r)$, and returns the $E(m_b)$.
(10)	Test-query (U^i) : According to the $E(m_b)$ obtained from challenge-query, A outputs a guess bit b' . If $b' = b$, return 1; otherwise return 0.

let $E(x) = A^\alpha$, $E(y) = B^\beta$, so we can get a quadruple $\text{DDH}(g, A, B, \text{DDH}(E(x), E(y)))$. Then we have

$$\text{DDH}(E(x), E(y)) = \text{DDH}(A^\alpha, B^\beta) = \text{DDH}(A, B)^{\alpha\beta}. \quad (7)$$

Moreover, $\text{Ask}H_4$ means that the adversary A had queried the random oracle H on $E(x)$ or $E(y)$. Then we get

$$\begin{aligned} \Pr[\text{Ask}H_3] &= \Pr[\text{Ask}H_4] \\ \Pr[\text{Ask}H_4] &\leq q_h \text{Adv}_{\mathcal{A}}^{\text{DDH}} \left(t + \frac{1/3 \cdot q_{\text{send}} + q_{\text{exe}} + 1}{p^2} \right). \end{aligned} \quad (8)$$

According to the above equations, we can conclude that

$$\text{Adv}_{\text{O-mPSI}}^{\text{IND-CPA}}(\mathcal{A}) \leq q_h \text{Adv}_{\mathcal{A}}^{\text{DDH}} \left(\left(t + \frac{1/3 \cdot q_{\text{send}} + q_{\text{exe}} + 1}{p^2} \right) + \frac{q^2 + (1/3 \cdot q_{\text{send}} + q_{\text{exe}})^2}{2p} \right). \quad (9)$$

The simulation queries involved in the protocol are as follows. \square

6. Performance Evaluation

6.1. Theoretical Evaluation. In this section, regarding the complexity of calculation and communication, we compare the O-mPSI protocol with the protocols in [11, 12, 20]. We choose these three protocols because both parties of the protocols in [11, 12] can know the intersection, and the protocol in [20] supports outsourcing. These protocols are very similar to our protocol. The comparison results are shown in Table 4.

Computation complexity. Our O-mPSI protocol uses a cuckoo hash table to store data, and the ElGamal threshold encryption algorithm to encrypt data. We use the number of

modular exponential operations to evaluate the computational overhead of O-mPSI protocol. Party P_1 and party P_2 each performs $2m + mb$ exponential operations when encrypting the hash tables T_1 and T_2 and performs w exponential operations in the decryption phase. The server only performs the intersection calculation which does not involve any exponential operations, where m represents how many bins are in the hash table, b is the size of the bin, and w is the intersection-cardinality. Therefore, the protocol O-mPSI has performed $2(2m + mb + w)$ modular exponential operations in total. We define $n = n_1 \gg n_2$, $b = 4$, $mb = 1.5n$, where n_i ($i = 1, 2$) is the cardinality of the private set of party P_i . The possible values of intersection w are in the range $[0, n]$, where we take its maximum value n . Therefore, the computational complexity is $6.5n$ in the O-mPSI protocol.

TABLE 4: Comparison of the properties of O-mPSI protocol and related protocols.

Protocol	Comp. Cost		Comm. Cost	Fairness
	User	Server		
Debnath's mPSI1 [11]	$48n$	—	$103n$	Yes
Debnath's mPSI2 [12]	$23n + 7$	—	$12n + 4$	Yes
Abadi's PSI [20]	$10n + 15$	$10n + 15$	$16n + 24$	No
Our O-mPSI	$6.5n$	0	$6n$	Yes

The protocol in [11] is based on the two-way oblivious pseudorandom function mOPRF, without the participation of a third-party server, and its computation complexity is $48n$. The protocol in [12] also does not involve the third-party server; its computation complexity is $23n + 7$. In the verifiable delegated PSI protocol of [20], the user performs $10n + 15$ exponential operations and the server performs $10n + 15$ exponential operations, so the overall computation complexity of [20] is $20n + 30$.

From the analysis above, it is clear that our O-mPSI protocol has much lower computational complexity than the other three protocols.

Communication complexity. Our O-mPSI protocol uses the number of transmitted ciphertexts to express the complexity of communication. Party P_1 sends mb encrypted elements $E(x_i^k)$ and mb rerandomized elements $E^*(y_i^k)$, for $1 \leq i \leq b$ and $1 \leq k \leq m$, to P_2 . Party P_2 sends mb encrypted elements $E(y_i^k)$ and mb rerandomized elements $E^*(x_i^k)$, for $1 \leq i \leq b$ and $1 \leq k \leq m$, to P_1 . Thus, the O-mPSI protocol generates a total of $4mb$ ciphertexts transmissions. The communication complexity of the protocol O-mPSI is $6n$, due to $mb = 1.5n$.

In [11], the protocol generates $103n$ ciphertexts interactions, the protocol in [12] generates $12n + 4$ ciphertexts interactions, and the protocol in [20] generates $16n + 24$ ciphertexts interactions.

To sum up, we can see that our O-mPSI protocol is superior to the other three protocols in terms of the communication complexity.

6.2. Experimental Evaluation. To verify the theoretical analysis results in Section 6.1, the computational costs of our O-mPSI protocol and the protocols in [11, 12, 20] are compared through experiments. The experimental platform is Windows 10, AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz, 16 GB RAM, and the compilation environment is MyEclipse 2017. In the experiment, we set $b = 4$ and $mb = 1.5n$, where m represents how many bins are in the hash table, b is the size of the bin, and n is the set cardinality.

Since our O-mPSI protocol and the protocols in [11, 12, 20] all use homomorphic encryption algorithm to encrypt data, we first compare the time it takes for the four protocols to execute with different modulus lengths. In this experiment, we set $n = 1000$; for modulus of 128 bit, 256 bit, 512 bit, and 1024 bit, the different homomorphic encryption times of the 4 protocols are shown in Figure 2.

It can be seen from Figure 2, with the increase of modulus length, the time of homomorphic encryption performed by the four protocols also increases. Among them, the time cost

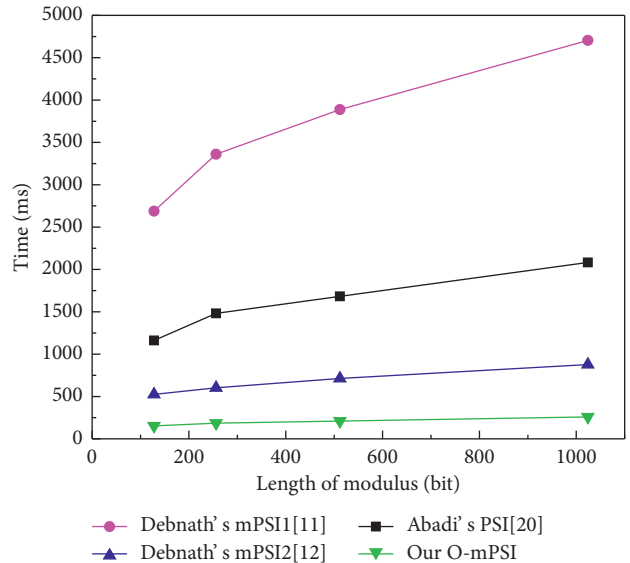


FIGURE 2: Time to perform homomorphic encryption of different modulus length.

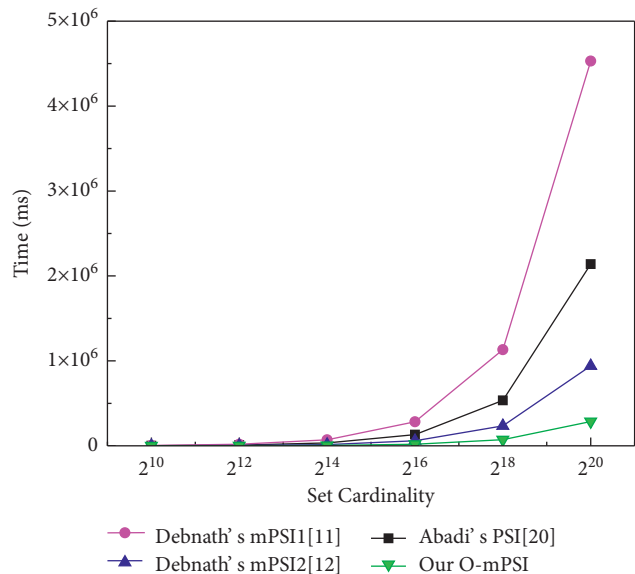


FIGURE 3: Running time at different set cardinality.

of the O-mPSI protocol is lower than that of Debnath's mPSI1, mPSI2 protocol, and Abadi's PSI protocol, and the growth trend is the slowest. The time cost in this experiment depends on how many modular exponential operations are used in the scheme. And according to Table 3, the modular

exponential operation is least used in the calculation of the O-mPSI protocol. Therefore, the increase of modulus length has the least impact on O-mPSI protocol.

We further compare the time it takes for the four protocols to execute at different set cardinalities. In this experiment, we fixed the modulus length to 1024 bit; for the cardinality n of $2^{10}, 2^{12}, 2^{14}, 2^{16}, 2^{18}, 2^{20}$, the running time of the four protocols can be seen in Figure 3.

It can be seen from Figure 3 that the execution time of the four protocols increases with the increasing size of the data set. Among them, the execution time overhead of the O-mPSI increases most slowly compared to that of Debnath's mPSI1, mPSI2 protocol, and Abadi's PSI protocol. This is because the O-mPSI protocol in this paper uses hash algorithm to process the set elements in advance. Therefore, as the size of the data set continues to increase, the time cost curve of O-mPSI protocol has grown slowly, while the time cost curve of the other three protocols has increased significantly.

7. Conclusions

To address the problem of data privacy and security sharing in edge-assisted IoT, we proposed a fair outsourced mPSI protocol with a lower computational cost. The proposed O-mPSI scheme uses the existing hash bin-to-bin method to calculate the intersection and improves it to further reduce the number of element comparisons. The calculation of intersection is outsourced to the edge server, which reduces the computing burden on both sides. The scheme adopts ElGamal threshold encryption algorithm to ensure data security. Only when both parties cooperate can all ciphertexts be completely decrypted. Therefore, this scheme can effectively resist the collusive attack between the server and any one of the two parties. And it proves that this scheme is IND-CPA secure under the DDH assumption. Through theoretical analysis and experimental evaluation, it is shown that the computational cost of our proposed O-mPSI protocol proposed in this paper is lower than that of other mutual PSI protocols.

The combination of edge computing and IoT improves the intelligence of IoT devices and introduces intelligent devices into all aspects of life. The massive use of smart IoT devices has led to a sharp increase in the amount of data generated by users. Privacy protection and secure sharing of big data in the IoT have become the focus of current research. Therefore, research on more secure and efficient PSI technology applied to edge-assisted IoT is our future research direction.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Fund of China (no. 61802117), the PhD Foundation of Henan Polytechnic University (no. B2021-41), the Support Plan of Scientific and Technological Innovation Team in Universities of Henan Province, China (no. 20IRTSTHN013), the youth backbone teacher support program of Henan Polytechnic University (2018XQG-10), and the Research Foundation of Young Core Instructor in Henan Province, China (2018GGJS058).

References

- [1] C. Zhou, J. Hu, and N. Chen, "Remote care assistance in emergency department based on smart medical," *Journal of Healthcare Engineering*, vol. 2021, Article ID 9971960, 10 pages, 2021.
- [2] R. L. Fritz, M. Wilson, G. Dermody, M. E. Schmitter, and D. J. Cook, "Automated smart home assessment to support pain management: multiple methods analysis," *Journal of Medical Internet Research*, vol. 22, no. 11, Article ID e23943, 2020.
- [3] R. S. Krishnan, S. Manikandan, J. R. F. Raj, and Y. H. Robinson, "Android Application Based Smart Bus Transportation System for Pandemic Situations," in *Proceedings of the 2021 Third International Conference On Intelligent Communication Technologies And Virtual Mobile Networks (ICICV)*, pp. 938–942, Tirunelveli, India, February 2021.
- [4] K. Sabri, A. Toufik, and M. Mohamed, "Edge-based Safety Intersection Assistance Architecture for Connected Vehicles," in *Proceedings of the 2021 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 272–277, Harbin City, China, July 2021.
- [5] D. Fadolalkarim, E. Bertino, and A. Sallam, "An Anomaly Detection System for the Protection of Relational Database Systems against Data Leakage by Application Programs," in *Proceedings of the 2020 IEEE 36th International Conference On Data Engineering (ICDE)*, pp. 265–276, Dallas, TX, USA, April 2020.
- [6] S. K. Debnath, K. Sakurai, K. Dey, and N. Kundu, "Secure Outsourced Private Set Intersection with Linear Complexity," in *Proceedings of the 2021 IEEE Conference On Dependable and Secure Computing (DSC)*, pp. 1–8, Aizuwakamatsu, Fukushima, Japan, February 2021.
- [7] Y. Shi and S. Qiu, "Delegated key-policy attribute-based set intersection over outsourced encrypted data sets for CloudIoT," *Security and Communication Networks*, vol. 2021, no. 4, 11 pages, Article ID 5595243, 2021.
- [8] S. Dittmer, Y. Ishai, S. Lu et al., "Function Secret Sharing for PSI-CA: With Applications to Private Contact Tracing," 2020, <https://arxiv.org/abs/2012.13053>.
- [9] Y. Qian, J. Shen, P. Vijayakumar, and P. K. Sharma, "Profile matching for IoMT: a verifiable private set intersection scheme," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 10, pp. 3794–3803, 2021.
- [10] S. K. Debnath and R. Dutta, "Provably secure fair mutual private set intersection cardinality utilizing bloom filter," vol. 10143, pp. 505–525, in *Proceedings of the International Conference on Information Security and Cryptology*, vol. 10143, Springer, Beijing, China, November 2016.

- [11] S. K. Debnath and R. Dutta, "Towards fair mutual private set intersection with linear complexity," *Security and Communication Networks*, vol. 9, no. 11, pp. 1589–1612, 2016.
- [12] S. K. Debnath and R. Dutta, "New realizations of efficient and secure private set intersection protocols preserving fairness," vol. 10157, pp. 254–284, in *Proceedings of the International Conference on Information Security and Cryptology*, vol. 10157, Springer, Beijing, China, November 2016.
- [13] S. K. Debnath and R. Dutta, "Fair mP. S. I. and mPSI-C. A.: Efficient constructions in prime order groups with security in the standard model against malicious adversary," *IACR Cryptol. ePrint Arch*, vol. 216, 2016.
- [14] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on OT extension," *ACM Transactions on Privacy and Security*, vol. 21, no. 2, pp. 1–35, 2018.
- [15] R. Agrawal, A. Evfimievski, and R. Srikant, "Information sharing across private data-bases," in *Proceedings of the 2003 ACM SIGMOD International Conference On Management Of Data*, pp. 86–97, California, CA, USA, June 2003.
- [16] Y. Cai, C. Tang, and Q. Xu, "Two-party privacy-preserving set intersection with FHE," *Entropy*, vol. 22, no. 12, pp. 1339–1353, 2020.
- [17] Y. Huang, D. Evans, and J. Katz, "Private Set Intersection: Are Garbled Circuits Better than Custom Protocols," in *Proceedings of the nineteenth Network and Distributed Security Symposium (NDSS 2012)*, San Diego, CA, USA, February 2012.
- [18] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai, "Efficient circuit-based PSI with linear communication," in *Proceedings of the Advances in Cryptology - EUROCRYPT 2019*, vol. 11478, pp. 122–153, Springer, Darmstadt, Germany, May 2019.
- [19] A. Kavousi, J. Mohajeri, and M. Salmasizadeh, "Efficient scalable multi-party private set intersection using oblivious PRF," in *Proceedings of the International Workshop on Security and Trust Management*, vol. 13075, pp. 81–99, Springer, Darmstadt, Germany, October 2021.
- [20] A. Abadi, S. Terzis, and C. Dong, "Verifiable delegated private set intersection on outsourced private datasets," vol. 9603, pp. 149–168, in *Proceedings of the International Conference on Financial Cryptography & Data Security*, vol. 9603, pp. 149–168, Springer, Christ Church, Barbados, February 2016.
- [21] A. Abadi, S. Terzis, R. Metere, and C. Dong, "Efficient delegated private set intersection on outsourced private datasets," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 4, pp. 608–624, 2019.
- [22] A. Kavousi, J. Mohajeri, and M. Salmasizadeh, "Improved Secure Efficient Delegated Private Set Intersection," in *Proceedings of the 2020 twentyeighth Iranian Conference on Electrical Engineering (ICEE)*, pp. 1–6, Tabriz, Iran, May 2020.
- [23] L. Kissner and D. Song, "Privacy-preserving Set Operations," in *Proceedings of the Annual International Cryptology Conference*, pp. 241–257, Springer, Santa Barbara, California, CA, USA, August 2005.
- [24] J. Camenisch and G. M. Zaverucha, "Private intersection of certified sets," vol. 5628, pp. 108–127, in *Proceedings of the Financial Cryptography and Data Security*, vol. 5628, Springer, Accra Beach, Barbados, February 2009.
- [25] M. Kim, H. T. Lee, and J. H. Cheon, "Mutual Private Set Intersection with Linear Complexity," vol. 7115, pp. 219–231, in *Proceedings of the International Workshop on Information Security Applications*, vol. 7115, pp. 219–231, Springer, Jeju Island, Republic of Korea, August 2011.
- [26] C. Dong, L. Chen, J. Camenisch, and G. Russello, "Fair private set intersection with a semi-trusted arbiter," vol. 7964, pp. 128–144, in *Proceedings of the Computer Science in Data and Applications Security and Privacy XXVII*, vol. 7964, Springer, Newark, NJ, USA, July 2013.
- [27] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang, "Understanding node capture attacks in user authentication schemes for wireless sensor networks," *IEEE Transactions on Dependable and Secure Computing*, 2020.
- [28] D. Wang and P. Wang, "Two birds with one stone: two-factor Authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708–722, 2018.
- [29] R. Pagh and F. F. Rodler, "Cuckoo hashing," *Journal of Algorithms*, vol. 51, no. 2, pp. 122–144, 2004.
- [30] P. Miao, S. Patel, M. Raykova, K. Seth, and M. Yung, "Two-sided malicious security for private intersection-sum with cardinality," vol. 12172, pp. 3–33, in *Proceedings of the Advances in Cryptology - CRYPTO 2020*, vol. 12172, Springer, Santa Barbara, CA, USA, August 2020.

Research Article

A Scalable Security Protocol for Intravehicular Controller Area Network

Zi-An Zhao , Yu Sun , Dawei Li, Jian Cui, Zhenyu Guan, and Jianwei Liu

School of Cyber Science and Technology, Beihang University, Beijing 100191, China

Correspondence should be addressed to Yu Sun; sunyv@buaa.edu.cn

Received 16 September 2021; Revised 2 November 2021; Accepted 30 November 2021; Published 31 December 2021

Academic Editor: Qi Jiang

Copyright © 2021 Zi-An Zhao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Intravehicular communication relies on controller area network (CAN) protocol to deliver messages and instructions among different electronic control units (ECU). Unfortunately, inherent defects in CAN include the absence of confidentiality and integrity mechanism, enabling adversaries to launch attacks from wired or wireless interfaces. Although various CAN cryptographic protocols have been proposed for entity authentication and secure communication, the redundancy in the key establishment phase weakens their availability in large-scale CAN. In this paper, we propose a scalable security protocol suite for intravehicular networks and reduce the communication costs significantly. A new type of attack, suspension attack, is identified for the existing protocols and mitigated in our protocol by leveraging a global counter scheme. We formally verify the security properties of the proposed protocol suite through the AVISPA tool. The simulation results indicate that the communication and computation efficiency are improved in our protocol.

1. Introduction

In a modern vehicle, hundreds of electronic control units (ECUs) are connected through in-vehicle network (IVN) gateway [1] to ensure life-critical operations, such as collision prediction and antilock braking. These ECUs communicate through controller area network (CAN) bus, the de facto standard for intravehicular communication, to guarantee driving security through sensing, actuation, and control. With the increasing number of sensor nodes, the function of the automobile is becoming much more complex, rendering more ECUs to be connected to the bus. In a luxury vehicle, more than 100 ECUs have been installed [2].

Inevitably, with more complex service and sophisticated communication functions incorporated into the automobile, the attack surface is growing rapidly. During the past decades, numerous researches demonstrated the ability to maliciously control a vehicle in road tests, from physical access to remote attack. As the most used automotive interface, onboard diagnostics (OBD) II provides direct and standard access to internal networks, through which CAN

buses are accessible and physically exposed to an adversary [3, 4]. In addition, remote exploitation is feasible via multifarious attack vectors [5], including mechanics tools, Bluetooth, cellular radio, and internet connectivity such as Wi-Fi and 4G [6–8]. These interfaces, which frequently interact with the external world, provide potential entry points for an attacker to insert malicious messages and codes in CAN. Numerous cases [9–11] have been reported to attack the engine, brake, lamp, and fuel gauge on Ford, Toyota, and Tesla automobiles in both parking and driving scenarios to realize steering, braking, and acceleration and display control. Even the firmware and built-in code can be modified by the attackers.

The major security vulnerabilities for CAN bus are the lack of confidentiality and integrity, as well as weak access control. Since there is no destination address in a CAN frame, each node can send and receive messages that are broadcast to the bus based on predefined configuration. An adversary can easily sniff data flow and insert malicious messages, which poses a dangerous situation for both driver and passengers. Consequently, the protection of the CAN bus relies on proper authentication and encryption mechanism.

Several cryptographic protocol suites have been proposed for secure CAN bus communication [12–14]. The recently published cryptographic protocol suite is proposed by Palaniswamy et al. in 2020 [15]. This comprehensive protocol suite consists of seven protocols, covering the integrated process from session key establishment and update to data transmission and connectivity with external devices. We provide an informal analysis of this protocol suite and investigate its potential security weakness against suspension attacks. The analysis also identifies its redundancy during authentication and session key agreement. To address these drawbacks, we propose a scalable cryptographic protocol suite for CAN bus, providing several verified security properties and lower communication overhead against the large-scale intravehicular network.

The following are the main contributions of this paper:

- (1) Through the analysis of the existing protocol suite, we identify the weakness against a new proposed attack, suspension attack. We raise a new protocol suite to mitigate the identified weakness based on the global counter scheme. Security verification is also presented in the AVISPA tool to this protocol suite.
- (2) In order to construct a more scalable CAN protocol suite, we propose a broadcasting scheme in the initial session key distribution protocol (ISDP) by using the Chinese remainder theorem (CRT). Compared with the previous works, the new protocol reduces communication overhead significantly that removes the obstacles for supporting more ECU connections in one CAN bus.
- (3) We propose an external device access protocol based on the certificateless signature scheme. Compared with certificate-based protocols, the proposed method lightens computational overhead and provides preferable efficiency.

The rest of the paper is organized as follows. Section 2 introduces the CAN bus protocol and intravehicular network, including the attack model and security requirements of this protocol suite. Section 3 presents various security solutions for CAN bus in prior works and the analysis of the existing protocol. Section 4 describes our new protocol suite. Section 5 verifies the security properties by using the AVISPA tool. Section 6 compares the simulation performance of our protocol suite with the existing schemes.

2. Background

2.1. CAN Network. The CAN protocol [16], designed by Bosch in 1981 for safety-critical and real-time applications, is a multimaster serial bus that has been widely used in industrial automation. CAN bus communication channel consists of twisted pair wires for differential signaling. The dominant level is represented by a logical 0, and the recessive level by a logical 1. Data frame, remote frame, error frame, and overload frame are four major frame types in CAN bus. Data frame, the most common message type, is used to transmit messages. ECU can also proactively request a

message from others with matching identifiers using a remote frame. CAN protocol entitles all ECUs to send an error frame when an error is detected. Overload frame is used to provide for an extra delay between data or remote frames.

A standard CAN data frame can be separated into several parts as shown in Figure 1.

- (i) *Start of Frame (SOF)*. It is a single dominant bit informing the start of transmission when the bus is idle.
- (ii) *Arbitration Field*. When multiple ECUs attempt to occupy the bus at the same time, an arbitration process is necessary to determine the sending sequence. ECU sending a higher identifier message will detect the lower identifier message and wait until the bus is free. Besides, identifier also implies the content of the message. For example, messages transmitting the temperature of a car engine have their own specific identifier.
- (iii) *Control Field*. It contains one identifier extend (IDE) bit, one reserved bit always set to 0, and four bits as data length code (DLC).
- (iv) *Data Field*. It refers to actual data of length from 0 to a maximum of 8 bytes for transferring information to another node.
- (v) *CRC Field*. It consists of 15-bit cyclic redundancy code (CRC) and 1-bit CRC delimiter. If the CRC checksum of transmitted data is different from the calculated value of the recipient, a CRC error will be triggered.
- (vi) *Acknowledge Field*. It consists of two bits such as ACK slot and ACK delimiter. By default, the sender node sets both of these two bits as recessive. If a receiver node verifies the message successfully, it replaces the ACK slot with dominant 0. Otherwise, a failed transmission will trigger an ACK response error and force the sending node to retransmit the message.
- (vii) *End of Frame*. CAN frame is terminated by a flag consisting of seven recessive bits.

Typically, an intravehicular network is divided into three subnetworks: powertrain, body, and infotainment as shown in Figure 2. The powertrain subnetwork contains life-critical operations such as engine, brakes, and chassis control components, where high bandwidth and stable communication capabilities are available [1]. In the past decade, the infotainment subsystem containing entertainment and information functions is growing rapidly. The body subsystem generically controls the doors, seats, and rear with low-speed CAN. The communication of ECUs among these subnetworks is facilitated through gateway ECU (as well as other kinds of networks), which is assumed to be more powerful than the usual ECUs.

2.2. Attack Scenarios. Though attackers have numerous interfaces to invade the intravehicular CAN bus, we can conclude the existing attack scenarios specifically proposed

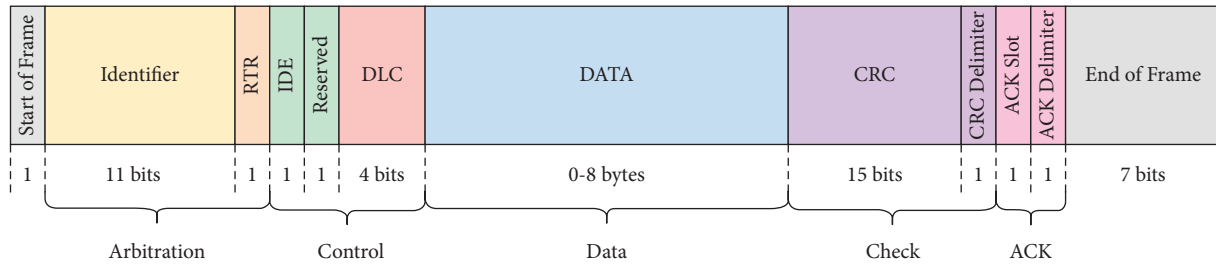


FIGURE 1: The structure of a standard CAN data frame.

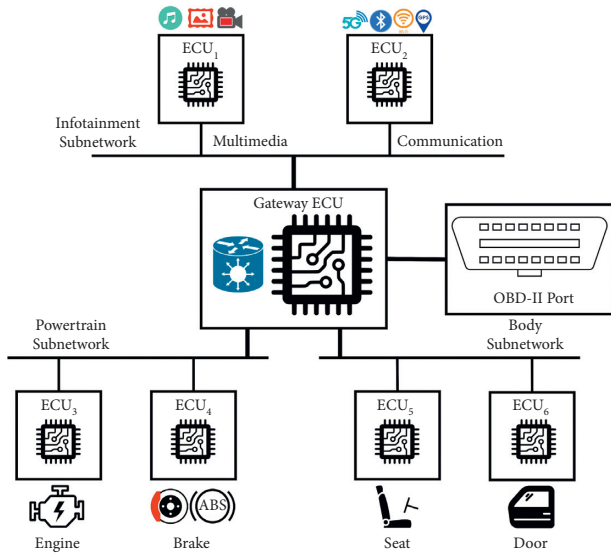


FIGURE 2: The topology of an intravehicle network.

by researchers into five parts: eavesdrop, replay, fuzzing, masquerade, and tampering [1, 17].

Eavesdrop. In this scenario, attackers have no ability to inject any messages into the bus. They can only monitor the communication on the bus and analyze the messages and flow as passive attackers.

Replay Attack. In a replay attack, there is no need for the attacker to understand the content of any message. The attacker just intercepts the historical message flow and records the corresponding relation between the message and the action caused by it in advance. Afterwards, the attacker can retransmit these messages at any time and obtain the same ECU’s action as he expected. In fact, if the freshness of messages transmitted in the network cannot be guaranteed, the receiver of the replayed message will always be deceived.

Fuzzing Attack. The objective of a fuzzing attack is to override any periodic messages by sending fabricated messages with the same arbitration ID at a higher frequency. Thus, other nodes that normally receive messages are forced to receive the fuzzing attack messages more frequently than the legitimate ones.

Masquerade Attack. The objective of a masquerade attack is to inject illegal messages while concealing the fact that an ECU is compromised. To achieve this goal,

the attacker needs to manipulate two ECUs, one original sender ECU and the other substitutional sender ECU. The attacker suppresses the original ECU from sending periodic messages and injects malicious messages at the original frequency with the substitutional sender ECU.

Tampering Attack. The objective of a tampering attack is to distort the messages in real-time, while the frame is transmitting. The attacker first launches a bus-off attack [17] to force the victim ECU to enter “error passive” state in advance. Afterwards, the attacker can change the transmitting data field arbitrarily without triggering any bit errors. The CRC field is also changed accordingly to prevent CRC error.

Suspension Attack. We also identified a new attack scenario, suspension attack, which is effective on existing CAN cryptographic protocol suites. This attack can be launched by leveraging the error handling mechanism and the lack of data flow sequence check. For example, in [13, 15], counters are assigned to synchronize messages between sender ECU and receiver ECU for each ID. However, the cross-ID sequence cannot be protected by using the separated ID-based counter.

Here, we propose two ways for attackers to launch a suspension attack.

First, attackers can send an error frame when the sending node is transmitting the ACK or SOF field of the proposed data frame. Since the data and CRC fields are completed, attacks can sniff the whole message while concealing its availability with an error frame. Once other ECUs receive the error frame and drop the received data frame, their counter will not update as normal, which gives a chance for attackers to replay this message at any time afterwards.

Second, attackers can just suspend one ECU when it is ready to send a new data frame. Before the message is sent, this ECU will be shut down, and this message will be cached in the transmission buffer. Since the message has never been sent out, the counter will not be overdue unless other ECUs send a new message with the same ID.

We used Raspberry Pi and CAN analyzer to build a demonstration system to present a suspension attack as shown in Figure 3. In our system, the attacker is assumed to have the ability to block the messages in the sending buffer of a victim sender ECU. Thus, the attacker can launch a suspension attack as long as he refuses to forward the message

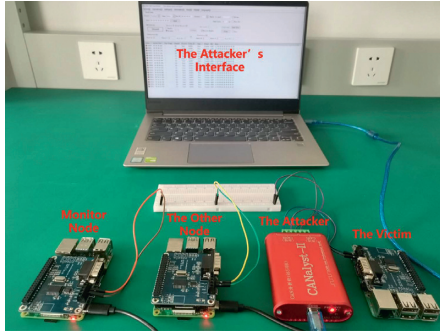


FIGURE 3: The demonstration system for suspension attack.

to the CAN bus and replays it afterwards. The result shows that the delayed message cannot be detected by other ECU nodes, which implies vulnerability in protocols using a local counter scheme.

2.3. Security Requirements. Based on the attack scenarios presented above, we give an attack model to describe the capability of the adversary in a formal way. The attacker is assumed to run the efficient polynomial-time algorithm, which has full control to the intravehicle network. The attacker is under the Dolev–Yao model [18], with the capabilities to act as a legitimate user to send, obtain, or even manipulate messages from any party in the interactive cryptographic protocol. The attacker can intercept and masquerade a legitimate user to interact with other parts in the protocol, attempting to inject malicious messages or steal ephemeral and long-term secrets. In Section 5, a security verification based on AVISPA gives an attacker instance under Dolev–Yao model to this proposed cryptographic protocol suite. In Section 4.7, we strengthen the capabilities of the attackers just like under Canetti–Krawczyk model and analyze the protocol informally.

Under the presence of the adversary defined above, the following security goals are going to be of interest:

- (i) *Confidentiality.* In the original CAN protocol, data frames are transmitted in clear, making the key information such as ID and data be exposed to an attacker apparently. Accordingly, every data frame should be encrypted to avoid eavesdropping. Only a legitimate node with a proper key can decrypt the message.
- (ii) *Mutual Authentication.* Two communicating entities must confirm the legality of the other before transmitting, in case of temporarily illegal access to steal traffic.
- (iii) *Key Freshness.* The session key should be updated in time under the circumstances when a counter overflow happens or an external device connection is released, which guarantees that the current session key is randomly and independently selected.
- (iv) *Forward and Backward Secrecy.* Forward secrecy protects past sessions against future compromises of keys. That is, the disclosure of the current session

key will not lead to the leak of the session key that has been used in the past [19]. Backward secrecy indicates that compromise of past session keys does not affect the security of future session keys.

3. Related Work

Since confidentiality and authenticity of the messages transmitted on the CAN bus are not guaranteed, attackers have numerous opportunities to remotely eavesdrop or tamper with data and cause great danger to the safety of vehicles. In order to solve the potential safety problems of the intravehicular CAN network, researchers have proposed various types of solutions.

In order to reduce computational complexity and bandwidth on the CAN bus, several lightweight cryptographic protocols have been proposed for confidentiality, integrity, and authenticity. Nilsson et al. proposed a delayed data authentication mechanism, which uses 64 bit MAC for four data frames embedded in the following four data frames with no data [20]. Herrewége et al. proposed a backward-compatible data authentication protocol based on HMAC [21]. Groza et al. designed LiBrA-CAN, an efficient protocol based on mixed message authentication codes (M-MACs), which aggregates several MACs into one to increase security [12]. Subsequently, Groza et al. proposed a TESLA-like [22] protocol by the priority of messages [23]. They used a master node to verify messages from the sender ECU and recomputed the tag with the key shared with the receiver ECU. Farag et al. proposed CANTrack, an intuitive scheme based on a dynamic key for encrypting CAN messages to prevent replay attacks [24]. Fassak et al. proposed a secure protocol for ECU authentication and session key establishment based on elliptic curve cryptography (ECC) [25]. Pan et al. proposed a new security scheme combining private key derivation algorithm based on electronic fingerprint and ECC algorithm for shared key distribution, which enhanced the nonreproducibility of messages [26].

Other methods improve the security of CAN bus in physical characteristics or other aspects. Lin et al. presented a new formulation that models the path-based security constraints and minimizes security risk directly to solve the problem that the overhead of security mechanisms might cause violations of design constraints [27]. Jain et al. utilized the physical properties of the CAN bus to construct a tree-based group key exchange protocol, which has logarithmic complexity for node addition and deletion [28]. Nürnberger and Rossow proposed VatiCAN, which was designed to be backward-compatible to allow tried and trusted components to rely on the same CAN messages [29]. Humayed and Luo proposed an ID-hopping scheme aiming to prevent targeted DoS attacks [30]. Siddiqui et al. proposed an authentication framework using the physical unclonable function for enhanced security, which can be integrated with existing resource constraint embedded devices [31].

The recent CAN security scheme based on the cryptographic protocol is proposed by Palaniswamy [15]. They analyzed the existing frame-level authentication protocol proposed by Woo et al. [13] and identified weaknesses and

limitations. The enhanced protocol suite covers entity authentication, secure transmission for remote frames and data frames, session key updates, and secure access with external devices. After evaluation, the new protocol suite is claimed to satisfy known key secrecy (KKS) [32] and withstand the attacks, namely masquerading, replay, and MITM. However, the protocol suite proposed by Palaniswamy et al. still has some drawbacks in efficiency and potential security weaknesses. Initial session key distribution protocol (ISDP) and session key update protocol (SKUP) are derived from a variant of the AKEP2 protocol. Despite security is guaranteed, broadcasting property in CAN bus is not leveraged to these protocols. GECU has to establish redundant challenge-response authentication with each ECU and cause unnecessary communication overheads. As for data transmission protocol (DTP) and RTRP protocol, each counter is arranged to record the data flow of a certain arbitration ID. Unfortunately, this scheme is vulnerable to suspension attacks introduced in Section 2.2. When a suspension attack is launched, messages will be forced to delay without disturbing their consolidated sequence, so this attack will not be defended by detecting the anomalous counter value in HMAC. The other drawback is that no protocols have been designed for cross-subsystem data transmission. Since there are usually two or more CAN bus in modern vehicles, it is necessary to directly exchange cross-subsystem messages encrypted with different session keys. In vehicle connection protocol (VCP), the digital certificate introduced in [13, 15] will lead to extra computational overheads in complicated certificate management and verification procedure of certificate chains. These drawbacks motivate us to improve its performance and reconstruct the whole protocol suite.

4. Scalable Protocol Suite

In the new proposed protocol suite, we reconstruct the design of previous works by enhancing the efficiency during authentication procedure and addressing security weaknesses against local counters. The new protocol suite consists of six phases, among which ISDP and SKUP are constructed for session key derivation; DTP, RTRP, and cross-subsystem data transmission protocol (CSTP) are designed to protect confidentiality and integrity for data transmission; and VCP is for external device authentication and key establishment. The notations used in the proposed protocol suite are presented in Table 1.

4.1. Initial Session Key Distribution Protocol. This protocol allows GECU to establish an initial session key with ECUs in the same subnetwork as shown in Figure 4. The existing ISDP protocols are based on some variants of the AKEP2 protocol, requiring GECU to finish the complete authentication process with each ECUs [13, 15]. We propose a simplified ISDP protocol by fully leveraging the broadcast mechanism of the CAN bus and the property of the Chinese remainder theorem. In this protocol, GECU completes authentication and session key distribution with all the ECUs in the same time.

Step 1. GECU Challenge

The protocol is launched by GECU. GECU generates a random integer Seed and uses long-term preshared key GK to calculate its HMAC value $MAC = H_{GK}(ID_{GECU}||Seed)$, then pack the seed and HMAC together, and send to the bus.

Step 2. ECU Response

Each of ECU_i can verify the Seed and generate another random integer R_i and random prime P_i as challenge values. Then ECU_i generates $MAC_i = H_{GK}(K_i||Seed||R_i||P_i)$ as a response for GECU's challenge. K_i , R_i , and P_i are also contained in MAC_i for identification and message integrity.

Step 3. GECU Response

After receiving ECUs' response, GECU separately verifies MAC_1 to MAC_n and generates $Hash_i = H_{GK}(K_i||Seed||R_i)$ as response value for ECU_i which is slightly different from MAC_i . By using the property of the Chinese remainder theorem, GECU can construct $S = (\sum_{i=1}^n Hash_i y_i Z_i) \bmod Z$ that is congruent to $Hash_i$ modulo P_i for each i from 1 to n . Therefore, each ECU_i can verify their unique response $Hash_i$ using the same S . Finally, session keys EK and AK can be derived from i for both GECU and ECUs.

4.2. Data Frame Transmission Protocol. The security goals of data frame transmission protocol are providing confidentiality and integrity for CAN data frame. To prevent replay attack against data frame, a counter is needed to provide the freshness of ciphertext as shown in Figure 5. ECUs in the same subnetwork share a global counter.

Step 1. Data Frame Generation

When an ECU receives a message from the bus, it always increases its counter to guarantee synchronization. In DTP protocol, the sender ECU uses the CTR mode of AES to encrypt the message. Since the length of the data field is 8 bytes, only the first 64 bits of $Enc_{EK}(CTR)$ are used to generate ciphertext $Enc_{EK}(CTR) \oplus m$. The authentication part consists of HMAC of arbitration ID, ciphertext, and counter.

Step 2. Decryption/Counter Update

After the receiver ECU receives a message, HMAC is verified first before decryption. Other ECUs also increase their counters to guarantee the synchronization except an error frame is received.

4.3. Remote Frame Transmission Protocol. RTRP protocol is similar to DTP as shown in Figure 6. The main difference is that the remote frame does not have a data field; thus, only hash computation is necessary.

Step 1. RTR Frame Generation

When the receiver ECU needs messages from the sender ECU, the former ECU sends a remote frame

TABLE 1: Notations used in the proposed protocols.

Notation	Description
GECU	Gateway ECU
ECU _{<i>i</i>}	ECU using identity <i>i</i>
ID _{<i>i</i>}	Arbitration identifier used by the <i>i</i> th ECU
Seed _{<i>k</i>}	Seed value to derivate session key in <i>k</i> th session
R _{<i>i</i>}	Random value generated by ECU _{<i>i</i>}
GK	Long-term preshared symmetric key between GECU and all ECUs
K _{<i>i</i>}	Long-term preshared symmetric key between GECU and ECU _{<i>i</i>}
P _{<i>i</i>}	Big prime generated by ECU _{<i>i</i>}
S	Aggregated HMAC value generated by GECU
KDF _{<i>k</i>}	Session key derivation function
EK _{<i>k</i>}	Encryption key of <i>k</i> th session
AK _{<i>k</i>}	Authentication key of <i>k</i> th session
CTR _{<i>i</i>}	Frame counter of <i>i</i> th subnetwork
Q	Generator on the elliptic curve group \mathbb{G} with order <i>q</i>
<i>a, b, r_i</i>	Random number generated in Z_q
(<i>y_i, Y_i</i>)	Secret key and public key pair used in external connection for device <i>i</i>
(<i>s, P_{pub}</i>)	Secret key and public key pair of key generation center
Enc _{EK} (·)	Symmetric encryption function
H _{<i>k</i>} (·)	Secure keyed hash function, (<i>y_i, Y_i</i>): $\{0, 1\}^* \times \text{key} \rightarrow \{0, 1\}^{32}$
H ₁ (·, ·)	Secure hash function, H ₁ (·, ·): $\{0, 1\}^{32} \times \mathbb{G} \rightarrow \{0, 1\}^{32}$

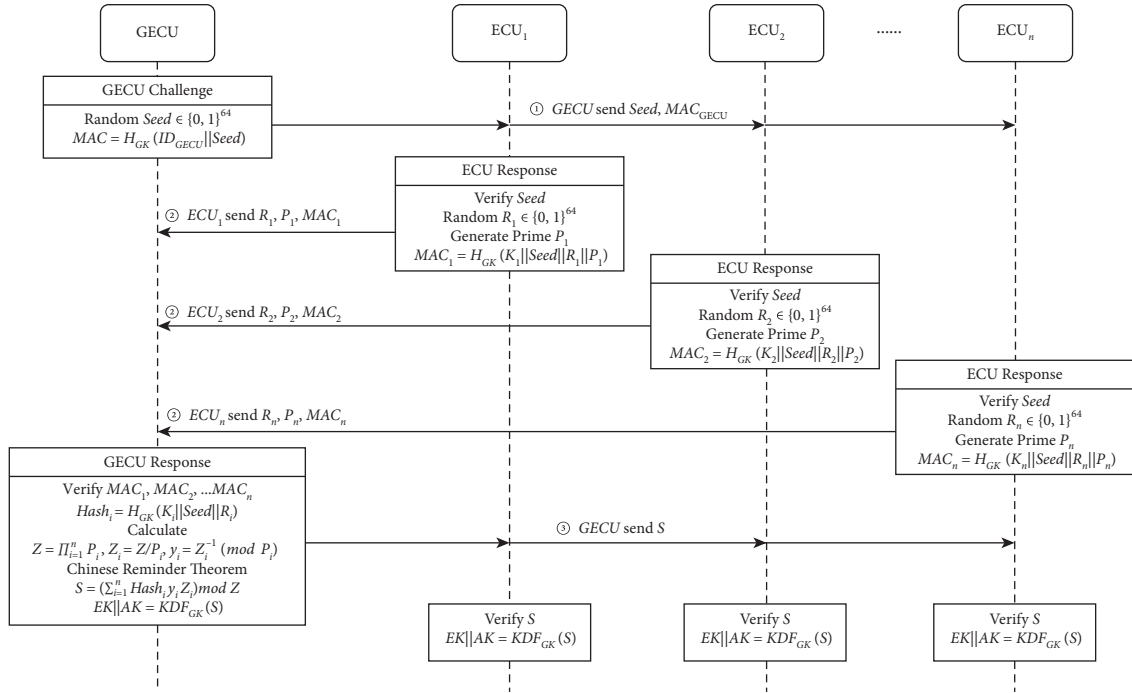


FIGURE 4: The procedure of ISDP.

$MAC_{RTR} = H_{AK}(RTR \text{ Frame} || ID_r || CTR)$ containing arbitration ID and counter value.

Step 2. Verification/Counter Update

After the sender ECU receives the message, it increases its counter if the MAC is verified. Similarly, other ECUs also update their counters if the message is received. Afterwards, the sender ECU can send data frames following the DTP protocol.

4.4. Cross-Subsystem Data Frame Transmission Protocol.

CAN buses in in-vehicle networks are separated into different subnetworks, such as powertrain, chassis, safety, and infotainment parts. All subnetworks are independent of each other but connected with a center gateway. However, the demand for cross-subnetwork data transmission still exists. In ISDP protocol, session keys are established in their own subnetworks. ECUs in different subnetworks cannot communicate directly. Therefore, a cross-data transmission

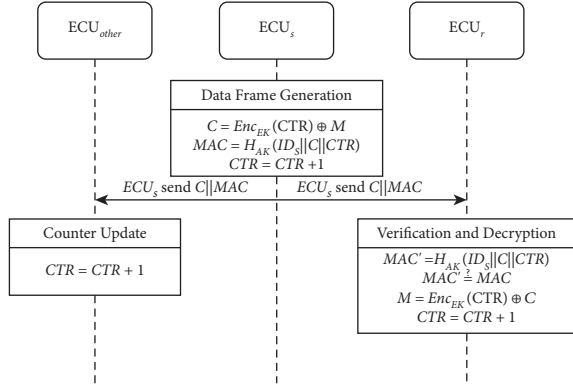


FIGURE 5: The procedure of DTP.

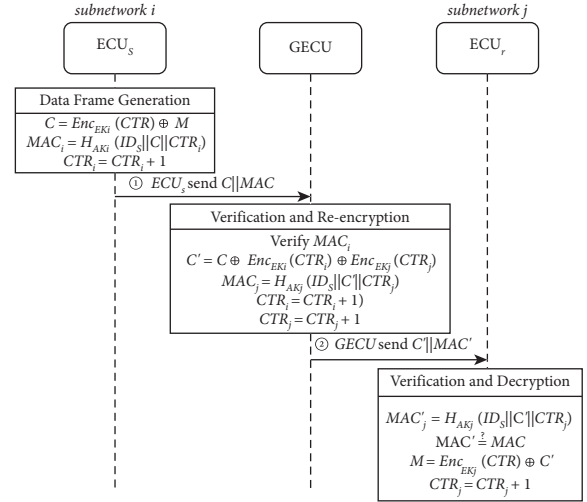


FIGURE 7: The procedure of CSTP.

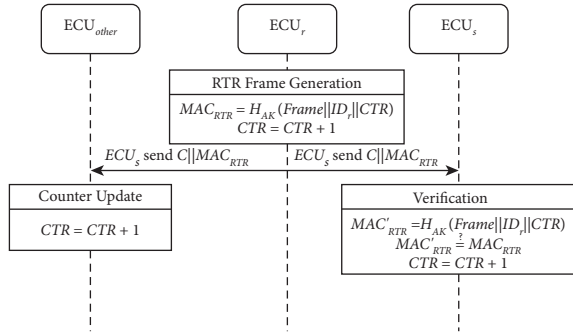


FIGURE 6: The procedure of RTRP.

protocol is needed for encryption communication. In addition, synchronization of counters in different subnetworks is impossible. Thus, message retransmission of GECU is essential for cross-subnetwork messages as shown in Figure 7.

Step 1. Data Frame Generation

At first, the sender ECU from subnetwork i generates a data frame using EK and AK as in DTP protocol.

Step 2. Verification and Re-encryption

When GECU receives a frame with a certain arbitration ID, a retransmission mechanism will be triggered. GECU verifies the messages, re-encrypts with the session key in subnetwork j and re-transmits them to the other subnetwork following a routing table. GECU can convert counter values from different subnetworks and provide synchronization.

Step 3. Verification and Decryption

Finally, the receiver ECU can receive and verify the messages from the other subnetwork.

4.5. Session Key Update Protocol. Session keys need to be updated under two conditions. First, after a predefined time, counters need to be reinitialized in case of overflow. Second, when external devices are released from the vehicle, the session key will be updated in case of

malicious leakage. The procedure of SKUP is shown in Figure 8.

Step 1. New Session Key Derivation

Session key update protocol is launched by GECU. When the global counter reaches a predefined value or releases an external device, GECU will send a data frame. The data field is a new random seed, and the authentication part is the HMAC value $MAC = H_{AK}(ID_{GECU} || Seed_{k+1} || GK)$ that contains arbitration ID, a fresh random seed, and long-term key GK to ensure authenticity.

Step 2. Verification and Update

All other ECUs will verify this message and derive a new session key from this HMAC value. Then, ECUs' responses will be sent to achieve the key confirmation.

4.6. Vehicle Connection Protocol. When an external device is connected to the CAN bus, the authentication and key agreement process must be done with GECU. However, certificate-based authentication is heavy for vehicular access protocol that needs the support of public key infrastructure (PKI). To avoid the usage of certification, we propose a cryptographic scheme that the pair of secret and public keys can only be derived by an authority (e.g., the car manufacturer), which can ensure the creditability of the vehicle's public key as shown in Figure 9.

Step 0. Initial Key Generation

When a vehicle or an external device is manufactured, it should be registered with key generation center (KGC) that holds the master secret key s and publishes its public key $P_{pub} = s \cdot Q$ as the public parameter. KGC derives a pair of asymmetric keys (y_i, Y_i) for the registered device, where $Y_i = r_i \cdot Q$, $y_i = r + s \cdot H_1(ID_i, Y_i)$, and r_i is random generated in Z_q . Subsequently, (y_i, Y_i) is injected to the memory of the device via a secure channel.

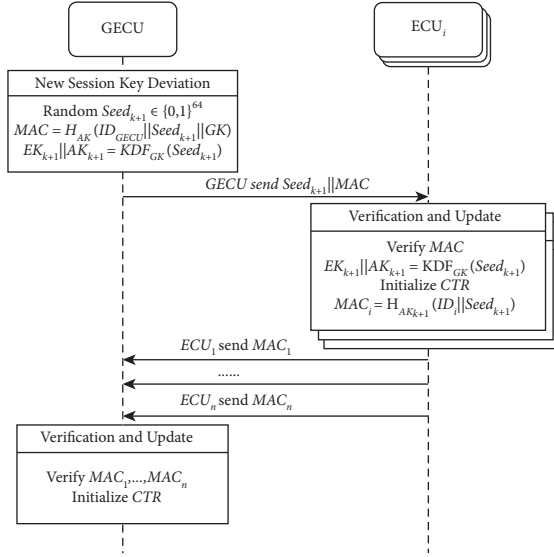


FIGURE 8: The procedure of SKUP.

Step 1. EDEV Challenge

When the external device is intended to connect to a vehicle, it first generates a random challenge value $V_E = a \cdot Q$.

Step 2. GECU Response and Challenge

Then GECU generates a signature $S_G = b + y_G \cdot h_{E2}$ on the identity ID_E and challenge value V_E of the external device. Then the signature, challenge value $V_G = b \cdot Q$, and public key Y_G are transmitted together.

Step 3. EDEV Response

The external device can verify the signature S_G with GECU's public key Y_G . Accordingly, the external device generates a signature $S_E = a + y_E \cdot h_{G2}$ on GECU's identity and challenge value. Afterwards, the signature S_E is transmitted with public key Y_E . Since GECU has been authenticated by the external device, the session key can be derived by the ECDH scheme on the external device side.

Step 4. GECU Verification

GECU can authenticate with the external device if $S_E P = V_E + h'_{G2}(Y_E + h_{E1} P_{pub})$ is satisfied. By using the ECDH scheme, the session key can be derived on the GECU side.

4.7. Informal Security Analysis of New Protocol Suite

Mutual Authentication. In the ISDP protocol, authentication relies on the preshared symmetric key. The responses generated by both ECU and GECU require a group key GK shared among entities in a subnetwork and a unique key K_i shared between GECU and ECU_i, which are unobtainable to the adversary. In VCP protocol, asymmetric key pair held by the external device and GECU guarantees authenticity.

Session Key Agreement. In ISDP protocol and SKUP protocol, the session key is derived from a group broadcasted verification value and a preshared key GK . In VCP protocol, the session key is derived from Diffie–Hellman scheme with signature for entity authentication, which avoids the threat of man-in-the-middle attack.

Protocol Attack Resistance

- (1) **Replay Attack.** If attackers act as GECU to launch ISDP protocol by replaying the initial challenge message, the impersonated GECU cannot generate a valid response without GK . Similarly, attackers cannot generate a valid ECU response message as normal ECU because of the freshness of the seed. In DTP and RTRP protocol, by using a counter for synchronization, an old counter value from a replay message will not be accepted by normal receiver ECU. In SKUP protocol, the new session key derivation message is the last frame using the current session key, which is meaningless to replay. In VCP protocol, both GECU and the external device must generate a signature to a fresh unique value from the other side, which is impossible to replay.
- (2) **Suspension Attack.** The new proposed DTP, CSTP, and RTRP protocol uses a global counter for all messages with a different ID. Suspension to any random ECU will lead to desynchronization of the delayed message and malicious messages will be discarded automatically.
- (3) **Masquerade Attack.** When the attacker tries to send a malicious random data frame, the receiver ECU or GECU will detect the anomalies and send an error with the protection of frame authentication provided by HMAC. Other ECUs on the bus will not update their counter. Therefore, an injection attack will not influence the availability of the protocol through counter synchronization mechanism.

Forward/Backward Secrecy. SKUP provides forward/backward secrecy. Since GK is contained in the calculation of MAC, even if AK is revealed to the attacker, EK and AK used in other sessions will not be exposed. In other words, only for entities with the knowledge of both AK and GK , SKUP can be triggered validly.

5. Security Verification

AVISPA is a push-button tool for formal analysis of the security protocol. It provides a modular and expressive formal language (HLPSL) for specifying protocols and their security properties, integrating different back ends that implement a variety of automatic analysis techniques. To generate the message sequence chart and check that the HLPSL specification is correct, we used the SPAN tool to choose the automatic analysis techniques and run the HLPSL script. AVISPA comprises four back ends: OFMC, CL-AtSe, SATMC, and TA4SP. Because of the calculation

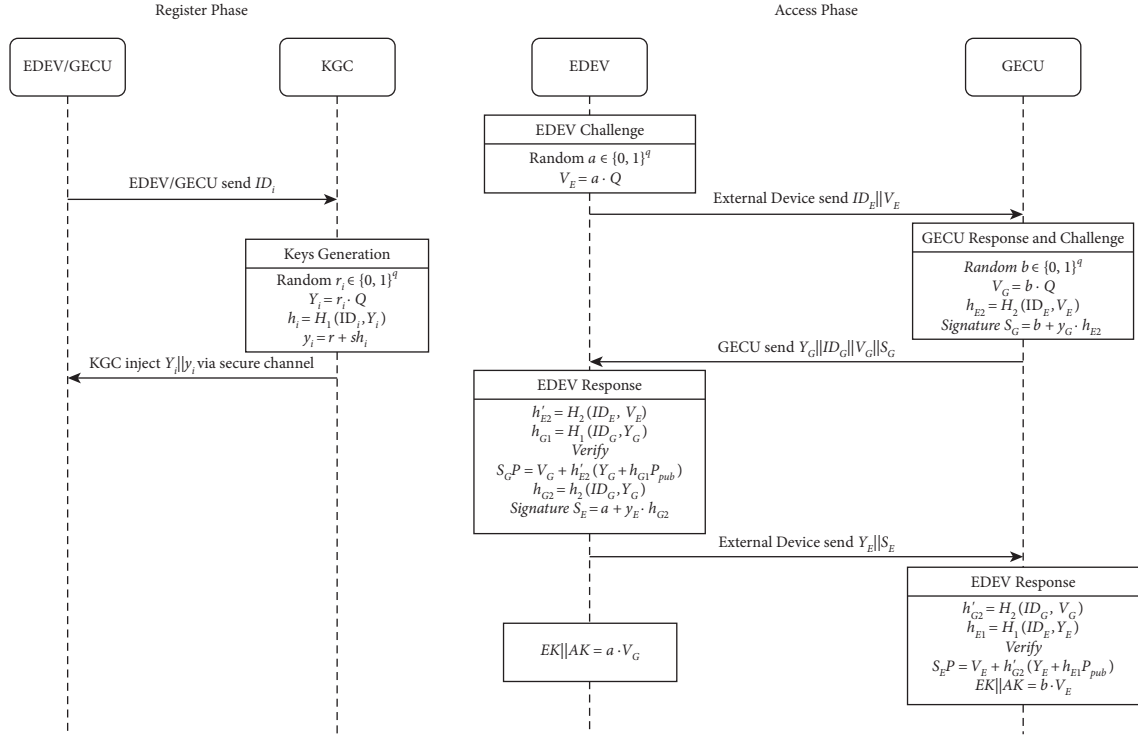


FIGURE 9: The procedure of VCP.

process of the security protocol, we used the OFMC tool to test the security properties.

We begin with the description of ISDP in HPSL. Figure 10 illustrates the information and actions of ECU_i in ISDP protocol. This element just illustrates the basic roles of the protocol, namely ECU and $GECU$. Preknown information is set by the parameters of the role. Variables described in section “local” are used to create or receive information. Section “transition” describes a detailed protocol process of ECU_i . “State” is used to identify the status of the role and facilitate the next action. Security properties are also described in section “transition” that will be analyzed with the implementation of the protocol.

The second element is shown in Figure 11. This element illustrates the composition of those basic roles for representing the protocol to facilitate the reference of the environment. Instead of section “transition,” section “composition” refers to basic roles and transfers the pre-known information to the basic roles.

Figure 12 illustrates the execution environment of the protocol instance to study. To instantiate the protocol, this element describes the preknown information of basic role and role sessions. The set “intruder_knowledge” indicates the information that the intruder can obtain. Section “composition” refers to the role session for the instance of the protocol. If multiple sessions are allowed to occur at the same time, in reality, this section can refer to several sessions to excavate possible attacks.

Figure 13 includes two elements: section “goal” and the execution of role environment. Section “goal” illustrates the declaration of the security properties to analyze. The execution of environment starts the process

of the protocol and analyzes the proposed security properties.

Descriptions for other protocols with their own security properties defined in AVISPA are similar to ISDP protocol. Table 2 shows the total verification results of the proposed protocol suite in AVISPA. The confidentiality and authenticity of critical parameters in each of the six protocols are all verified successfully. Besides, forward secrecy is also guaranteed in ISDP, SKUP, and VCP protocol.

6. Performance Analysis

This section presents a performance comparison of our protocol suite with the protocols of Woo et al. [13] and Palaniswamy et al. [15]. In the comparison, we consider security properties as well as costs in communication and computation. A simulation in MATLAB is also presented for efficiency comparison. The result shows that the proposed scheme achieves the minimum communication complexity in ISDP/SKUP protocol and the best computation cost in the VCP protocol. Furthermore, enhanced security attributes are also provided in this scheme that is not available in the previous works.

Table 3 shows the communication efficiency in the session key establishment stage, covering ISDP and SKUP protocols. As for ISDP protocol, the former designs in [13, 15] request $GECU$ to establish a session key with each ECU independently. In other words, $GECU$ needs to accomplish a mutual challenge-response scheme with each ECU s in sequence, which leads the message complexity up to $3n$, where n represents the number of ECU nodes. In our modified ISDP design, by using the CRT theorem, $GECU$'s


```

role role_ECUi(
  Ei,Ej,G      :agent,
  Ki,GK       :symmetric_key,
  IDi,IDj,IDgecu :text,
  KDF,H       :hash_func,
  SND,RCV     :channel(dy))
played_by Ei def=
local
  State      :nat,
  Rj,Ri,Seed,Pi :text,
  EK,AK      :hash(symmetric_key.message),
  MACg       :hash(symmetric_key.text.text),
  MACi       :hash(symmetric_key.symmetri_key.text.text),
  MACj       :hash(symmetric_key.symmetri_key.text.text),
  MAC        :hash(symmetric_key.hash(symmetric_key.symmetri_key.text.text)
                .hash(symmetric_key.symmetri_key.text.text).message),
  Hashi      :hash(symmetric_key.symmetri_key.text.text),
  Hashj      :hash(symmetric_key.symmetri_key.text.text),
  S          :message
init
  State := 0
transition
1. State = 0  $\wedge$  RCV(Seed'.MACg')  $\wedge$  MACg' = H(GK.IDgecu.Seed')  $\Rightarrow$ 
  State' := 2  $\wedge$  Ri' := new()  $\wedge$  Pi' := new()  $\wedge$  MACi' := H(GK.Ki.Seed'.Ri'.Pi')  $\wedge$ 
  SND(Ri'.Pi'.MACi')  $\wedge$  witness(Ei,G,e_g_MACi,MACi')
2. State = 2  $\wedge$  RCV(Hashi'.Hashj'.S'.MAC')  $\wedge$  MAC' = H(GK.Hashi'.Hashj'.S')  $\wedge$ 
  Hashi' = H(GK.Ki.Seed.Ri)  $\Rightarrow$ 
  State' := 4  $\wedge$  EK' := KDF(GK.S')  $\wedge$  AK' := KDF(GK.S')  $\wedge$ 
  SND(Hashi'.Hashj'.S'.MAC')  $\wedge$ 
  request(Ei,G,e_g_MACgi,MACg)  $\wedge$ 
  request(Ei,G,e_g_S,S')  $\wedge$  witness(Ei,Ej,e_e_S,S')  $\wedge$ 
  secret(EK',ek1,{Ei,Ej,G})  $\wedge$  secret(AK',ak1,{Ei,Ej,G})
end role

```

FIGURE 10: Role role_ECUi of ISDP in HLPSSL.

```

role session(
  Ei,Ej,G      :agent,
  Ki,Kj,GK     :symmetric_key,
  IDi,IDj,IDgecu :text,
  KDF,H,Mutiple,Division,Add,Invert,Mod      :hash_func)
def=
local SND,RCV :channel(dy)
composition
  role_ECUi(Ei,Ej,G,Ki,GK,IDi,IDj,IDgecu,KDF,H,SND,RCV)
   $\wedge$  role_ECUj(Ei,Ej,G,Kj,GK,IDi,IDj,IDgecu,KDF,H,SND,RCV)
   $\wedge$  role_GECU(Ei,Ej,G,Ki,Kj,GK,IDi,IDj,IDgecu,KDF,H,Mutiple,
             Division,Add,Invert,Mod,SND,RCV)
end role

```

FIGURE 11: Role session of ISDP in HLPSSL.

response to each ECUs can be aggregated in a single message and decrease the communication complexity significantly to $n + 1$. Furthermore, in modified ISDP protocol, every ECUs receive GECU's challenge at the same time; therefore, their calculation can be processed simultaneously, which is beneficial to lighten computation delay. As for the SKUP protocol, compared with the design of Woo et al. [13], we simplified the in-sequence update scheme by using a broadcast scheme like the ISDP protocol.

In this section, we compare the computation overhead in VCP protocol with other current schemes. Since certificate verification operation is composed of two point multiplication operations, one point addition operation, and one hash operation, we mainly involve the time of hash operation, point multiplication over the elliptic curve (EC), and so on. The time corresponding to each operation is listed in

```

role environment()
def=
const
  ek1,ek2,ek3,ak1,ak2,ak3      : protocol_id,
  e_g_MACi, e_g_MACj          : protocol_id,
  e_g_MACgi, e_g_MACgj        : protocol_id,
  e_g_S, e_e_S                : protocol_id,
  ki,kj,gk                    : symmetric_key,
  ecui,ecuj,gecu              : agent,
  idi,idj,idgecu,s2           : text,
  kdf,h,mutiple,division,add,invert,mod : hash_func
intruder_knowledge = {idi, idj, idgecu, ecui, ecuj, gecu, kdf, h,
                     mutiple, division, add, invert, mod, kdf(s2.gk)}
composition
  session(ecui,ecuj,gecu,ki,kj,gk,idi,idj,idgecu,kdf,h,mutiple,division,
         add,invert,mod)
end role

```

FIGURE 12: Role environment of ISDP in HLPSSL.

```

goal
  secrecy_of ek1
  secrecy_of ek2
  secrecy_of ek3
  secrecy_of ak1
  secrecy_of ak2
  secrecy_of ak3
  authentication_on e_g_MACi
  authentication_on e_e_S
  authentication_on e_g_S
  authentication_on e_g_MACj
  authentication_on e_g_MACgi
  authentication_on e_g_MACgj
end goal
environment()

```

FIGURE 13: Goal and execution of environment of ISDP in HLPSSL.

Table 4 according to [33]. Table 5 shows the computation comparison of different schemes. The existing schemes like that in [13, 15] use PKI and digital certificates to provide a trusted public key that leads to extra computation for certificate verification. Due to the certificateless authentication signature scheme, our new proposed protocol achieves the minimum computation overhead.

The new proposed protocol suite shows not only efficiency in communication and computation aspects but also rich security features. Table 6 shows evidence of enhanced security properties especially in resistance to suspension attack, which is discussed in Section 2.2. Based on Tables 3, 5, and 6, the proposed scheme is efficient and robust in security than the previous works.

We simulate message communication delay for session key establishment procedure using Windows 10

TABLE 2: Result of verification.

Protocol	Property	Result	Attack
ISDP	EK Confidentiality	Pass	None
	AK Confidentiality	Pass	
	MAC Authentication	Pass	
	MAC_i Authentication	Pass	
	MAC_j Authentication	Pass	
	Forward secrecy	Pass	
DTP	CTR Confidentiality	Pass	None
	M Confidentiality	Pass	
	MAC Authentication	Pass	
RTRP	CTR Confidentiality	Pass	None
	MAC Authentication	Pass	
CSTP	CTR Confidentiality	Pass	None
	M Confidentiality	Pass	
	MAC Authentication	Pass	
	MAC' Authentication	Pass	
SKUP	EK Confidentiality	Pass	None
	AK Confidentiality	Pass	
	MAC_i Authentication	Pass	
	MAC_j Authentication	Pass	
	Forward secrecy	Pass	
VCP	a Confidentiality	Pass	None
	b Confidentiality	Pass	
	EK Confidentiality	Pass	
	AK Confidentiality	Pass	
	V_E Authentication	Pass	
	V_G Authentication	Pass	
	Forward secrecy	Pass	

TABLE 3: Communication costs in the key establishment procedure.

	Woo et al. [13]	Palaniswamy et al. [15]	Our scheme
<i>Communications involved in the ISDP protocol</i>			
ECU	1	1	1
GECU	2	2	2
Message complexity	$3n$	$3n$	$n + 1$
<i>Communications involved in the SKUP protocol</i>			
ECU	1	1	1
GECU	2	2	1
Message complexity	$2n$	$n + 1$	n
Total message complexity	$5n$	$4n + 1$	$2n + 1$

TABLE 4: Computation costs in VCP protocol.

Operation	Notation	Time (μs)
Hash function	T_{hash}	67
Symmetric encryption	T_E	161
Multiplication over EC	T_{ECM}	612
Addition over EC	T_{ECA}	125

TABLE 5: Computation costs in VCP protocol.

Schemes	Overall computation cost
[13]	$9T_{\text{ECM}} + 4T_{\text{ECA}} + 6T_H + T_E \approx 6571 \mu s$
[15]	$9T_{\text{ECM}} + 4T_{\text{ECA}} + 8T_H + T_E \approx 6705 \mu s$
Our scheme	$8T_{\text{ECM}} + 4T_{\text{ECA}} + 8T_H \approx 5932 \mu s$

environment with Intel Core i5-8265U @1.6 GHz and 8 GB RAM in MATLAB 2019b, involving the execution of ISDP and SKUP protocols. We set four scenarios for evaluating the communication delay with the number of ECUs in CAN bus with different CAN bus speeds. In the presented four scenarios, we, respectively, set 50, 75, 100, and 125 ECU devices in the CAN bus. As shown in Figure 14, message delay increases in all of the three schemes, but the message delay in our method is distinctively lower than the other two. Also, our scheme shows more competitive performance in large ECU numbers due to the low message complexity, which implies a broad prospect, especially in large-scale intra-vehicular networks.

TABLE 6: Security attributes comparison.

	[13]	[15]	Our scheme
Mutual entity authentication	✓	✓	✓
Session key freshness	✓	✓	✓
Resistance to DoS attack	×	✓	✓
Formal verification	×	✓	✓
Forward secrecy	×	✓	✓
Certificate free	×	×	✓
Resistance to suspension attack	×	×	✓

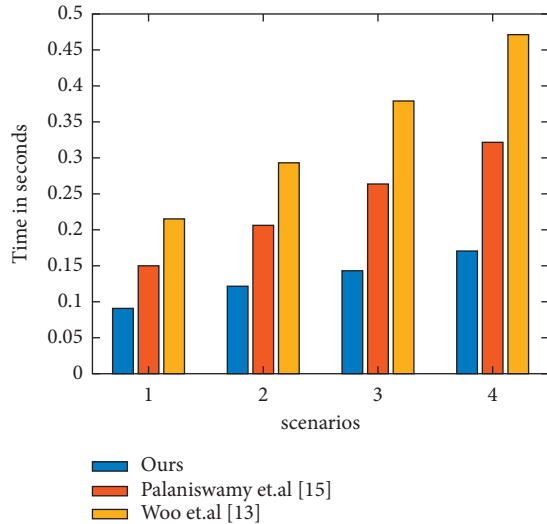


FIGURE 14: MATLAB simulation for communication delay.

7. Conclusion

This paper analyzed the limitation of the state-of-the-art cryptographic protocol suite for intravehicular CAN network by presenting its security weakness against our new identified CAN attack scenario, suspension attack. A new protocol suite is proposed to overcome the vulnerability as well as increase efficiency. The broadcasting scheme in the key distribution phase and certificateless schemes used in the external access phase have shown the ability in reducing communicational and computational overhead in performance analysis. Several security properties of the proposed protocol suite are also convinced under the security verification in AVISPA. In the future, we will enhance the secure CAN protocol suite by associating with intrusion detection systems to resist a broader range of attacks such as denial of service and extending the secure protocol to other intravehicular bus networks such as FlexRay and MOST to give an integrated solution for vehicle bus security system.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that no conflicts of interest exist.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (32071775) and the Opening Project of Shanghai Trusted Industrial Control Platform.







References

- [1] T. Huang, J. Zhou, Y. Wang, and A. Cheng, "On the security of in-vehicle hybrid network: status and challenges," in *Proceedings of the International Conference on Information Security Practice and Experience*, pp. 621–637, Springer, Melbourne, Australia, December 2017.
- [2] A. B. E. S. E. Team, "Future Advances in Body Electronics," Technical Report, NXP Semiconductors, Eindhoven, Netherlands, 2017.
- [3] K. Koscher, S. Savage, F. Roesner et al., "Experimental Security Analysis of a Modern Automobile," in *Proceedings of the 2010 IEEE Symposium On Security And Privacy*, pp. 447–462, IEEE Computer Society, Oakland, California, May 2010.
- [4] I. Rouf, R. D. Miller, H. A. Mustafa et al., "Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study," *USENIX Security Symposium*, vol. 10, 2010.
- [5] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *Black Hat USA*, vol. 201494 pages, 2014.
- [6] S. Checkoway, D. McCoy, B. Kantor et al., "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the USENIX Security Symposium*, vol. 4, pp. 447–462, San Francisco, CA, USA, August 2011.
- [7] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and vulnerable: A story of telematic failures," in *Proceedings of the 9th USENIX Workshop on Offensive Technologies WOOT 15*, Washington, DC, USA, August 2015.
- [8] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.
- [9] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Defense*, vol. 21, no. 260-264, pp. 15–31, 2013.
- [10] S. Nie, L. Liu, and Y. Du, "Free-fall: hacking tesla from wireless to can bus," *Briefing, Black Hat USA*, vol. 25, pp. 1–16, 2017.
- [11] M. Yan, G. Harpak, and J. Li, *Security Research on mercedes-benz: From Hardware to Car Control*, vol. 28, pp. 1–38, Black Hat USA, Black Hat Briefing, USA, 2020.
- [12] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede, "Libra-can: a lightweight broadcast authentication protocol for controller area networks," in *Proceedings of the International Conference on Cryptology and Network Security*, pp. 185–200, Springer, Berlin, Germany, December 2012.
- [13] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle can," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 993–1006, 2014.
- [14] S. Woo, H. J. Jo, I. S. Kim, and D. H. Lee, "A practical security architecture for in-vehicle can-fd," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2248–2261, 2016.
- [15] B. Palaniswamy, S. Camtepe, E. Foo, and J. Pieprzyk, "An efficient authentication scheme for intra-vehicular controller area network," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3107–3122, 2020.

- [16] R. B. GmbH, *Can Specification*, Robert Bosch GmbH, Gerlingen, Germany, 2.0 edition, 1991.
- [17] K. T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1044–1055, Vienna, Austria, October 2016.
- [18] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.
- [19] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pp. 235–244, Athens, Greece, November 2000.
- [20] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Efficient In-Vehicle Delayed Data Authentication Based on Compound Message Authentication Codes," in *Proceedings of the 2008 IEEE 68th Vehicular Technology Conference*, pp. 1–5, Calgary, Canada, September 2008.
- [21] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "Can-auth-a simple, backward compatible broadcast authentication protocol for can bus," in *Proceedings of the ECRYPT Workshop on Lightweight Cryptography*, p. 20, Louvain-la-Neuve, Belgium, November 2011.
- [22] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Proceedings of the Network and Distributed System Security Symposium, NDSS*, vol. 2001, pp. 35–46, San Diego, California, USA, January 2001.
- [23] B. Groza and S. Murvay, "Efficient protocols for secure broadcast in controller area networks," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2034–2042, 2013.
- [24] W. A. Farag, "Cantrack: Enhancing Automotive Can Bus Security Using Intuitive Encryption algorithms," in *Proceedings of the 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, pp. 1–5, Sharjah, United Arab Emirates, UAE, April 2017.
- [25] S. Fassak, Y. E. H. El Idrissi, N. Zahid, and M. Jedra, "A Secure Protocol for Session Keys Establishment between Ecus in the Can bus," in *Proceedings of the 2017 International Conference on Wireless Networks and Mobile Communications*, pp. 1–6, WINCOM), Morocco, Rabat, November 2017.
- [26] Q. Pan and J. Tan, "A Dynamic Key Generation Scheme Based on Can bus," in *Proceedings of the 2019 10th International Conference on Information Technology in Medicine and Education (ITME)*, pp. 564–569, IEEE, Qingdao, China, August 2019.
- [27] C. W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli, "Security-aware modeling and efficient mapping for can-based real-time distributed automotive systems," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 11–14, 2014.
- [28] S. Jain and J. Guajardo, "Physical layer group key agreement for automotive controller area networks," in *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*, pp. 85–105, Springer, Santa Barbara, CA, USA, Feb 2016.
- [29] S. Nürnberger and C. Rossow, "- vatiCAN - vetted, authenticated CAN bus," in *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*, pp. 106–124, Springer, Santa Barbara, CA, USA, August 2016.
- [30] A. Humayed and B. Luo, "Using id-hopping to defend against targeted dos on can," in *Proceedings of the 1st International Workshop on Safe Control of Connected and Autonomous Vehicles*, pp. 19–26, New York NY, USA, April 2017.
- [31] A. S. Siddiqui, Y. Gui, J. Plusquellic, and F. Saqib, "Secure communication over canbus," in *Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1264–1267, IEEE, Boston, MA, USA, August 2017.
- [32] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 453–474, Springer, Berlin, Heidelberg, April 2001.
- [33] M. Ouaisa, M. Houmer, and M. Ouaisa, "An enhanced authentication protocol based group for vehicular communications over 5G networks," in *Proceedings of the 2020 3rd International Conference on Advanced Communication Technologies and Networking (CommNet)*, pp. 1–8, Marrakech, Morocco, September 2020.

Research Article

High-Capacity Robust Behavioral Steganography Method Based on Timestamp Modulation across Social Internet of Things

Mingliang Zhang ^{1,2} Xiangyang Luo ^{1,2,3} Pei Zhang ^{1,2} Hao Li ^{1,2} Yi Zhang ^{1,2}
and Lingling Li ⁴

¹Zhengzhou Institute of Information Science and Technology, Zhengzhou 450001, China

²State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China

³Henan Province Key Laboratory of Cyberspace Situation Awareness, Zhengzhou 450001, China

⁴School of Intelligent Engineering, Zhengzhou University of Aeronautics, Zhengzhou 450046, China

Correspondence should be addressed to Xiangyang Luo; luoxy_jeu@sina.com

Received 17 September 2021; Accepted 2 December 2021; Published 31 December 2021

Academic Editor: Ding Wang

Copyright © 2021 Mingliang Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Social Internet of Things (SIoT) is an emerging field that combines IoT and Internet, which can provide many novel and convenient application scenarios but still faces challenges in data privacy protection. In this paper, we propose a robust behavioral steganography method with high embedding capacity across social networks based on timestamp modulation. Firstly, the IoT devices on the sending end modulate the secret message to be embedded into a timestamp by using the common property on social networks. Secondly, the accounts of multiple social networks are used as the vertices, and the timestamp mapping relationship generated by the interaction behaviors between them is used as the edges to construct a directed secret message graph across social networks. Then, the frequency of interaction behaviors generated by users of mainstream social networks is analyzed; the corresponding timestamps and social networks are used to implement interaction behaviors based on the secret message graph and the frequency of interaction behaviors. Next, we analyze the frequency of interaction behaviors generated by users in mainstream social networks, implement the interaction behaviors according to the secret message graph and the frequency of interaction behaviors in the corresponding timestamps and social networks, and combine the redundant mapping control to complete the embedding of secret message. Finally, the receiver constructs the timestamp mapping relationship through the shared account, key, and other parameters to achieve the extraction of secret message. The algorithm is robust and does not have the problem that existing multimedia-based steganography methods are difficult to extract the embedded messages completely. Compared with existing graph theory-based social network steganography methods, using timestamps and behaviors frequencies to hide message in multiple social networks increases the cost of detecting covert communication and improves concealment of steganography. At the same time, the algorithm uses a directed secret message graph to increase the number of bits carried by each behavior and improves the embedding capacity. A large number of tests have been conducted on mainstream social networks such as Facebook, Twitter, and Weibo. The results show that the proposed method successfully distributes secret message to multiple social networks and achieves complete extraction of embedded message at the receiving end. The embedding capacity is increased by 1.98–4.89 times compared with the existing methods SSN, NGTASS, and SGSIR.

1. Introduction

With the widespread use of IoT devices and social networks, people have combined them to form the SIoT [1], which build many interesting scenarios. For example, it allows users to convert data about plants, pets, and so forth

acquired by sensors in their homes into posts, which are then posted to social networks to share their lives in real time. In this era of connected everything, users' privacy protection is always a hot topic being discussed. For edge devices involving personal safety, more covert means of communication are needed to transmit secret commands that prevent

communication data from being monitored and tampered within the link to avoid significant loss of life and property. Steganography is a privacy protection method that uses redundant methods such as human vision and hearing to conceal the existence of secret communication [2] and has received wide attention from scholars in the field of information security. It allows SIoT to deliver critical data with higher security requirements in an imperceptible manner while delivering ordinary data. Therefore, it is of great importance and practical value to study the covert communication of social IoT.

This paper will focus on the way how edge devices of SIoT can implement steganography on social networks. The traditional steganography method based on multimedia has high embedding capacity and superior performance in resisting statistical detection [3]. However, the generated steganography object is vulnerable to channel attack when it is sent through a lossy channel, which will result in the inability to extract secret message correctly. Meanwhile, once the carrier is modified during the embedding of secret message [4], there is a risk of being identified as abnormal data [5]. Social network behavioral steganography (SNBS) is a method, which uses user's comments, forwardings, and other behaviors on social networks to realize covert communication. It combines the rich interaction behaviors of social networks [6] with steganography to make up for the shortcomings of traditional steganography methods in terms of robustness and detection resistance. On lossy channel, it does not cause the loss of behavioral data but ensures the robustness of steganography methods. In addition, secret message is hidden in behaviors, which effectively resists the analysis and detection technology for multimedia data; the sender and receiver do not directly establish communication to avoid causing special attention of the third party [7] and avoid the possibility of the sender's exposure or betrayal to testify against the receiver. However, the embedding capacity of SNBS is very low. Although it avoids the detection of multimedia data, it also introduces behavioral abnormalities. This limits the practical use of such methods. How to improve the embedding capacity and correct the abnormal behaviors is an urgent problem to be solved in the practical application of SNBS technology. This paper will focus on the improvement of SNBS capacity and the correction of abnormal behaviors.

Recently, researchers have developed a series of studies in the field of SNBS. Existing social network steganography methods can be divided into two categories according to the difference of secret message carriers: social network multimedia steganography (SNMS) and social network behavioral steganography (SNBS).

SNMS refers to embedding secret message into multimedia such as posts and comments published by users. This kind of methods combines multimedia such as text [8–11], image [12–15], and video containing secret message with a social network and uses posts, comments, and other ways to realize covert communication.

SNBS uses user's interaction to convey secret message and can be further divided into non-graph-theory-based steganography and graph-theory-based steganography. For non-

graph-theory-based steganography, in order to prove whether social network behavioral data can provide instructions for botnets, Pantic and Husain [16] use the length of Twitter to realize the transmission of secret data. This method has certain concealment, but it is easy to be detected. Zhang [17] proposes a method to send secret message by marking "love." The behavioral steganography method of marking "love" attempts to send message in the WeChat Moments, which is widely used in China, but there are insufficient embedding capacity and detection resistance. In view of the problem that the relationship between friends is not considered in [17], Hu et al. [18] use the behavioral correlation between sender and friend to calculate the reasonable probability of marking "love" on a social network, which improves the security, but the embedding capacity needs to be improved. Regarding the "prisoner model," the security issue between sender and receiver is not considered; Yang et al. [7] give a constraint framework to ensure content security and behavioral security and provide a reference method for the behavioral security of social network steganography. Li et al. [19] propose a framework for reposting posts and other network activities to hide secret message. This framework hides secret message in interactive activities and provides a theoretical basis for social network steganography. For graph-theory-based steganography, methods of using a graph theory to describe steganography can be traced back to [20]. This paper describes the method of using a dynamic data structure in memory to store secret messages. Nechta [21] constructs an undirected graph of secret message on social networks to achieve covert communication. The method proposed by Wu et al. [22] conveys secret message through an undirected graph and enhances the security of communication through a directed graph; however, there are some redundancies in the nodes of this method. To address the problem that the secret data may be exposed due to the attacker's mastery of the reconstruction graph process, Wu et al. [23] improve the secret data security by remapping the correspondence between the vertices of the graph structure with a key, but the embedding capacity of the method still needs to be improved.

In response to the above problems, this paper proposes a high-capacity and robust behavioral steganography method for cross-platform social networks based on timestamp modulation. This method uses accounts in multiple social networks to construct a directed secret message graph on the sender side and hides the secret message in the timestamps of the interaction behaviors generated by the secret message graph. At the receiving end, the receiver uses the shared account and key to extract the secret message through the public social networks. The main work of this paper is as follows:

- (1) This paper proposes a method to construct a directed secret message graph across multiple social networks and map the secret message into the timestamp generated by the edge set of the secret message graph. Compared with existing methods, this method has a very high embedding capacity, and the effective number of bits transferred and the embedding rate are improved.
- (2) This paper proposes a distribution modulation algorithm that hides secret messages into multiple

social networks by timestamps and fits the frequency characteristics of ordinary user interactions. This algorithm increases the cost of attackers to analyze the covert communication, reduces the probability of anomalous behaviors, and improves the detection resistance.

- (3) This paper proposes an algorithm to overcome the problem of information extraction failure due to interaction delay by increasing the redundancy time. The algorithm increases the number of mappings between participant account indexes and timestamps to avoid the possibility of mismatch between them and solve the robustness problem.

The rest of this paper is organized as follows. Section 2 briefly introduces the work related to SNBS. Section 3 introduces the method proposed in this paper. After that, Section 4 gives the experimental results and evaluation. Finally, this paper is concluded by Section 5.

2. Related Work

Steganography based on graph theory uses user's accounts and interactions on social networks to construct secret message graphs to realize covert communication. Compared with SNMS, steganography based on graph theory has high robustness and there is no statistical anomaly for multimedia steganography, and it is effective against multimedia steganography analysis techniques. Therefore, this section will briefly introduce the previous research based on graph theory steganography, so as to explain the feasibility of designing steganography methods based on graph theory.

Nechta [21] used social network behaviors to construct undirected graphs for covert communication, abbreviated as SSN. It constructs a secret message undirected graph $G(V, E, \varphi)$, where V denotes the set of vertices in the graph, E denotes the set of edges in the graph, and φ denotes the adjacency function. This function is shown in equation (1) and is used to define whether the edges between V_a and V_b exist in graph G . $|V|$ denotes the number of vertices in set V . Moreover, vertices (v_a, v_b) are exhaustively enumerated, where $a < b$. The number of bits that can be transmitted by this method is N_1 shown in equation (2).

$$\varphi = \begin{cases} 0, & E_{a,b} \notin E, \\ 1, & E_{a,b} \in E \end{cases} \quad (1)$$

$$N_1 = |V| * \frac{(|V| - 1)}{2}. \quad (2)$$

To facilitate the comparison of the ability of existing methods to deliver the size of secret messages, we take the product of the number of interactive behaviors h and the average number of bits carried by each behavior as the embedding capacity $c = h * l_s$. Let the average number of bits carried by a single behavior be l_s . In this paper, the numbers of bits sent in [21–23] are denoted as N_1, N_2, N_3 , and the numbers of interaction behaviors are denoted as h_1, h_2, h_3 , and their embedding capacities are denoted as c_1, c_2, c_3 ,

respectively. The embedding capacity in [21] is shown by the following equation:

$$c_1 = \frac{h * N_1}{h_1} = h * |V| * \frac{(|V| - 1)}{2h_1}. \quad (3)$$

The paper in [21] proposed a novel method to enable covert communication on social network, which provides a good reference for steganographic methods to social networks.

Wu et al. [22] used undirected graphs to hide secret message and directed graphs to hide topology and enhanced the security of the proposed method, abbreviated as NGTASS. For a graph $G(V, E)$, where $V = \{v_1, v_2, \dots, v_n\}$ and $E = \{e_1, e_2, \dots, e_n\}$, each bit of the secret message binary string is embedded as an edge of the undirected graph $G_0(V_0, E_0)$ and then the message graph G_0 is embedded into the directed graph $G_1(V_1, E_1)$. The edges in the directed graph G_1 correspond to the edges in the undirected graph G_0 , and the direction is from the high index vertex to the low index vertex. If we ignore the direction pointing, G_0 is a subgraph of G_1 actually. The premise of this method is that the steganography and extracting processes share a set U containing all vertices in G_1 . The interaction information of the social network is public, and the receiver can observe the interaction of the sender. When a receiver obtains G_0 and G_1 , the secret message will be extracted. It is worth noting that the method uses $|V|$ accounts to achieve covert communication, and one of the vertices is used to hide the topological information; in fact, $|V| - 1$ vertices are used to encode the secret message. N_2 is shown in equation (4) and c_2 is shown in equation (5).

$$N_2 = \frac{(|V| - 1) * (|V| - 2)}{2}, \quad (4)$$

$$c_2 = \frac{h * N_2}{h_2} = \frac{h * (|V| - 1) * (|V| - 2)}{2h_2}. \quad (5)$$

This method uses undirected graphs to encode secret message and uses additional vertices to hide the topology, which reduces the embedding capacity.

Wu et al. [23] based their work on [22] to remap the correspondence between the vertices of the graph structure by a key, abbreviated as SGSIR. This method controls $n + 2$ vertices, which are indexed from v_0 to v_{n+1} , and the vertex set consisting of v_1 to v_n is denoted by V . The edges whose starting and ending points belong to the vertex set V are denoted as the edge set E . The edge set E has m edges, where m is an integer power of 2. For the embedding process, the method selects m edges in E based on a random seed R and assigns an index. Then, the secret message is converted into a binary sequence, and every $\log_2 m$ group is converted into a decimal sequence $D = \{d_1, d_2, \dots, d_m\}$ and is allocated $m + 1$ build operations. Finally, four types of build operations are performed to send the secret message to the social network. For the extracting process, a receiver reconstructs the graph structure and extracts the secret message by using R, V , and public social network. The method controls $|V|$ vertices to achieve covert communication, but vertices v_n and v_{n+1} are used to denote 0 and 1, respectively, so the method uses $|V| -$

2 vertices to encode the secret message. N_3 is shown in equation (6) and c_3 is shown in equation (7).

$$N_3 = 2^{\lfloor \log_2((|V|-2) * (|V|-3)/2) \rfloor} * \lfloor \log_2 \frac{(|V|-2) * (|V|-3)}{2} \rfloor, \quad (6)$$

$$c_3 = \frac{h * N_3}{h^3} = \frac{h * 2^{\lfloor \log_2((|V|-2) * (|V|-3)/2) \rfloor} * \lfloor \log_2((|V|-2) * (|V|-3)/2) \rfloor}{h^3}. \quad (7)$$

This method uses an undirected graph to construct a secret message graph and uses interactive remapping to improve security. Even if the attacker successfully reconstructs the graph structure, he cannot obtain the embedded data. At the same time, each behavior can carry more secret messages. However, it consumes a certain amount of vertices to hide information such as key and topology and uses undirected graphs to realize covert communication, which reduces the embedding capacity of the method.

We draw the basic framework of this type of method, as shown in Figure 1. The main steps can be briefly described as follows:

- (i) Encrypting secret message: encrypt the secret message to be transmitted
- (ii) Converting secret message to binary: convert ciphertext information into binary which is easy to send
- (iii) Building a secret message graph: the secret data is modulated into a secret message graph
- (iv) Releasing the secret message graph to the social networks: the edges in the secret message graph will be interactively generated on the social networks in turn
- (v) Extracting secret message: the extracting process of these methods is the reverse process of the embedding process

Based on the principles of existing methods, it is clear that behavioral steganography methods based on graph theory using undirected graphs to hide secret message have shortcomings. They convey less secret message and have lower embedding capacity. Moreover, they use a single social network to achieve steganographic communication, which requires frequent operations on the social networks. Timestamp is one of the public properties of social networks, and if the secret message is transformed into timestamps distributed over multiple social networks and used to construct directed graphs, it can effectively reduce the number of interactions generated in a single social network and improve the detection resistance and embedding capacity. Therefore, in this paper, we focus on a high-capacity, anomaly-detection-resistant behavioral steganography method using social networks timestamps.

The main notations in this paper are shown in Table 1. The notations without subscripts have a general description, and the subscripts s and r denote belonging to the sender and receiver, respectively.

3. Proposed Method

Timestamp is a time representation, also called Unix timestamp, which is defined as the total number of seconds from 00:00:00 GMT on January 01, 1970, to a certain point in time. The proposed method modulates the secret message as timestamps, constructs a directed secret message graph, and releases the edges of the directed secret message graph to social networks in the form of interactive timestamps. In this section, we describe our proposed method through a framework diagram of the method, several key steps, and an example.

The basic idea of this method is as follows: firstly, the sender-controlled accounts are used as the set of vertices of the graph, the possible edges between the vertices are traversed, and index numbers are assigned to construct the secret message graph. Then, the edges in the graph are used to construct the interaction index matrix, denoted by \mathbf{I} , whose elements are called interaction indexes, and the secret message is converted into interaction indexes, and the mapping relationship between interaction indexes and timestamps are constructed. Finally, the secret message is modulated into timestamps by using interaction indexes as an intermediate form, which enables the embedding process of covert communication. The framework of the proposed method is shown in Figure 2; Steps 1–5 belong to the embedding process, and Steps 6–9 belong to the extracting process.

Step 1. Data preprocessing: using the key, the plaintext is encrypted into ciphertext, and the ciphertext is converted into decimal data suitable for subsequent steps.

Step 2. Generating interaction index matrix: the accounts controlled by the sender are used as the set of vertices of graph, and the possible edges between vertices are traversed. Hash the key, take some of the values as random seed to assign index numbers to the edges in the graph, and build a matrix called interaction index matrix.

Step 3. Generating cross-platform interaction sequence and construct the cross-platform directed interaction graph: the preprocessed data is used to generate a sequence for interaction through an interaction index matrix, which can construct an interaction graph among multiple social networks.

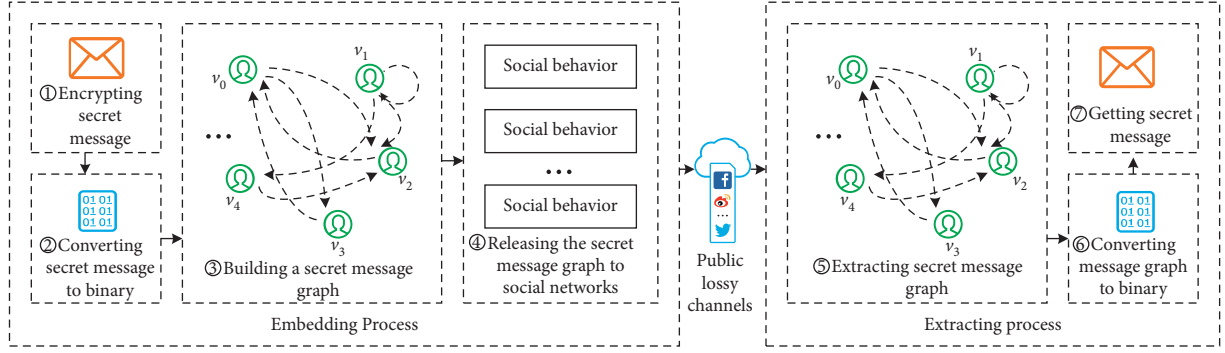


FIGURE 1: Basic framework of behavioral steganography based on graph theory.

TABLE 1: Notation table.

Notations	Explanation
A_s	ASCII code of the sender character
d	Maximum interaction delay
E	The set of edges in G
F	Time interval sequence of behaviors
G	Graph
I	Interaction index matrix
l_b	Number of accounts actually controlled by the sender
l_p	The binary to be sent
l_r	Binary with secret message
l_s	Embedding capacity of a single behavior
m	Number of edges in G
p	Time interval
r_b	Effective bit rate
R	Random seed
S	Timestamp index sequence
S_R	Random sequence
t_c	Current timestamp
V	The set of vertices in G
B_s, B_r	Grouped binary for sender and receiver
D_s, D_r	Decimal sequence of sender and receiver
E_s, E_r	Interaction sequence of sender and receiver
$n, N $	Number of accounts actually controlled
T_s, T_r	Sender and receiver timestamp sequence
V_s, V_r	The set of vertices of the sender and receiver

Step 4. Generating cross-platform robust timestamp sequence: this step firstly obtains data about the user's behaviors on the corresponding social networks, which is used to analyze the interaction patterns. Secondly, the behavior generated by a user at one moment may be recorded at the next moment, which will cause a delay in the timestamp of the behavior, resulting in the receiver not being able to extract the message correctly. This problem can be solved by using a time redundancy control mechanism, for which it is necessary to obtain the maximum time delay over a while. Finally, the mapping relationship between vertices and timestamps is constructed by random seed, and the timestamp sequence of behaviors is generated by combining the interaction sequence, interaction delay, and interaction patterns.

Step 5. Releasing secret message graph: combine directed interaction graphs and timestamp sequence to construct secret message graph and release it to social networks.

Step 6. Regenerating the interaction index matrix: the receiver uses the account set and key to generate the interaction index matrix. The process of generation is the same as Step 2.

Step 7. Extracting timestamp sequence: the receiver uses the accounts to obtain user data from the corresponding social networks. The receiver parses the user interaction data in turn to obtain a sequence of timestamps, the elements of which are of the form $\langle \text{sender index, timestamp} \rangle$.

Step 8. Reconstructing directed secret message graph: the receiver senses and obtains the maximum interaction delay of these networks over a while and tries to reconstruct the index of interaction participants by the maximum interaction delay and timestamp. Then, parse the $\langle \text{sender index, timestamp} \rangle$ element into a $\langle \text{sender index, interaction index} \rangle$ element, and construct a directed secret message graph.

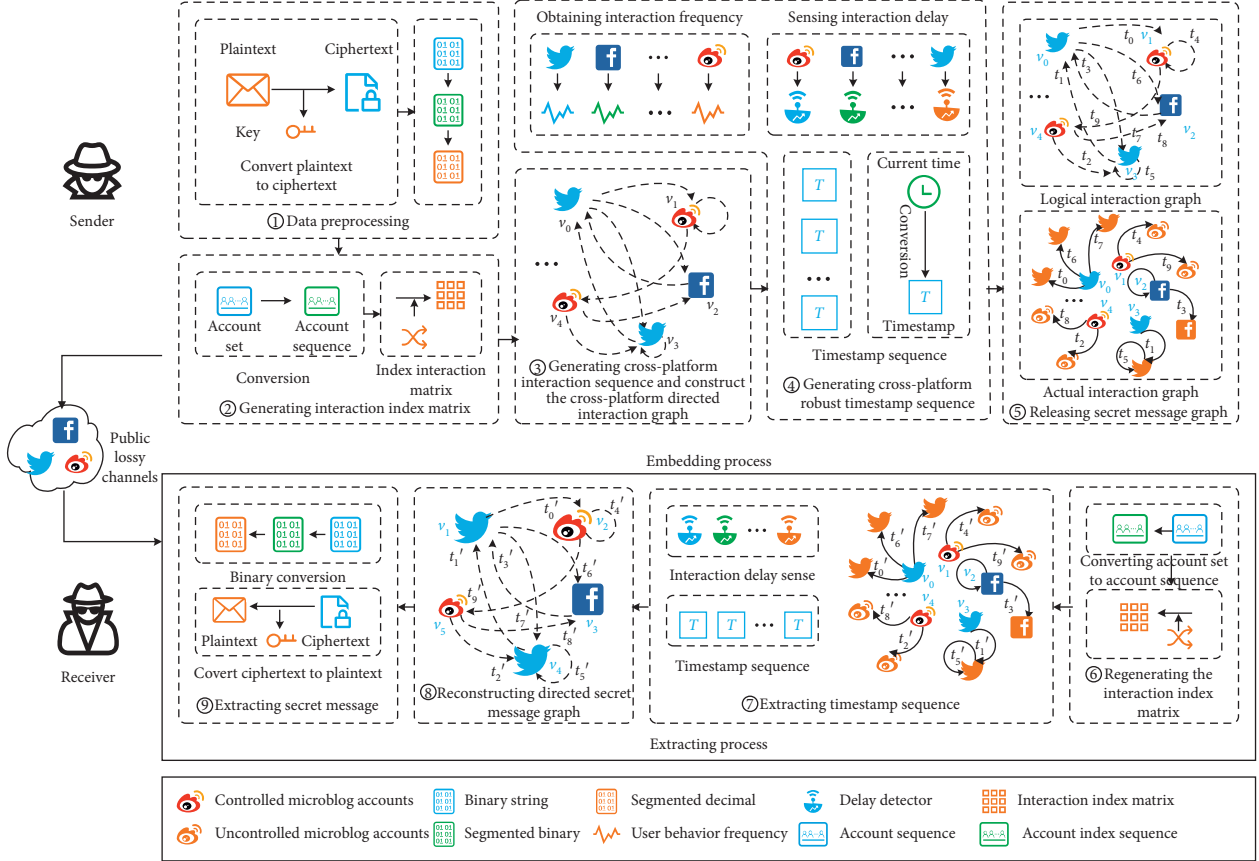


FIGURE 2: Framework of the proposed method.

Step 9. Extracting secret message: decimal secret data can be extracted by interaction indexes. Then, the secret message can be obtained through a share key.

In this paper, we use graph theory to describe the interaction behaviors between accounts. For a directed graph $G = V, E$, $G = \langle V, E \rangle$, $V = \{v_0, v_1, \dots, v_{n-1}\}$ and $E = \{e_0, e_1, \dots, e_{m-1}\}$. $\forall e \in E$, $e = \langle u, v \rangle$, which represents an interaction between accounts u and v . u is the account that generates the behaviors, and v is the account that interacts with u . For a directed graph G , with n vertices in V and m edges in E , n vertices can generate at most n^2 different interaction behaviors. We use l_b to denote the number of binary bits carried by each behavior. For coding purposes, m will be an integer power of 2, and $l_b = \log_2 m$ will be an integer. In fact, the following relationship will exist between l_b , m , and n in this method (note: “[\cdot]” means rounding down):

$$l_b = \lfloor \log_2 n^2 \rfloor. \quad (8)$$

From equation (8), when n is determined, l_b and m are obtained. For example, when $n = 6$, $l_b = 5$ and $m = 32$. The main steps of the method are described in detail in the following subsections.

3.1. Generating Interaction Index Matrix. An n -order matrix I is initialized, whose elements are all -1 , and, after the hash key, take part of the value as the random seed, and generate

m different pseudorandom indexes i in I , where $i \in [0, m - 1]$. At this time, the $n^2 - m$ elements of I are still -1 . The pseudocode to generate matrix I is shown in Algorithm 1.

3.2. Generating Cross-Platform Interaction Sequence and Construct Cross-Platform Directed Interaction Graph. The binary data obtained by encrypting the plaintext is divided into groups of 8 bits; each group is an element of $A_s = \{a_0, a_1, \dots\}$. Then, S_s is divided into l_b groups and converted to decimal, denoted as $D_s = \{d_0, d_1, \dots\}$. If the number of binaries in the remaining S_s is less than l_b , it needs to append 0 until its length is l_b . $\forall i_{j,k} \in I$, j, k are the row index and column index of matrix element i , respectively. By matching the positions of the elements in set D_s in matrix I , the row and column indexes of the elements are obtained, denoted as $\langle v_j, v_k \rangle$, and append them to E_s . The generation process of E_s is shown in Algorithm 2, which is used to implement the cross-platform interactive sequence assignment. A cross-platform directed interaction graph can be constructed based on E_s and V_s .

Accounts on different social networks cannot interact with each other directly; for example, an account on Twitter cannot comment on a Facebook post. To enable the relationship between accounts on social networks, we assign timestamp indexes to accounts, denoted as $S_s = \{s_0, s_1, \dots, s_{n-1}\} = \{0, 1, \dots, n - 1\}$. A timestamp-

```

Input:  $V_s, R$ 
Output:  $I$ 
(1) Generate a random sequence  $S_R$  of length  $m$  from  $R$ 
(2)  $n = \text{len}(V_s)$  //The len function represents the length of the acquired sequence
(3) Initialize an  $n$ -order matrix  $I$ , whose elements are all  $-1$ 
(4) for  $i = 0$  to  $m - 1$  do
(5)    $\text{temp} = S_R[i]$ 
(6)    $\text{row} = \lfloor \text{temp}/n \rfloor$ 
(7)    $\text{column} = (\text{temp} - \text{row} * n) \% n$ 
(8)    $I[\text{row}, \text{column}] = i$ 
(9) end for
(10) return  $I$ 

```

ALGORITHM 1: Interaction index matrix generation algorithm.

```

Input:  $V_s, I, D_s$ 
Output:  $E_s$ 
(1)  $l = \text{len}(D_s)$ 
(2) for  $i = 0$  to  $l - 1$  do
(3)    $\text{temp} = D_s[i]$ 
(4)   Get the row and column of temp in matrix  $I$ , and assign them to  $j$  and  $k$  respectively
(5)    $E_s.\text{append}([v_j, v_k])$ 
(6) end for
(7) return  $E_s$ 

```

ALGORITHM 2: Cross-platform interaction sequence allocation algorithm.

timestamp index mapping is constructed. For a given timestamp t , there is a unique timestamp index corresponding to it. For example, for accounts v_a, v_b , and v_c , v_a and v_b are two accounts on Facebook and v_c is an account on Twitter; v_a commented on a post of v_b at timestamp t . If the mapping value of t is c , it means that when timestamp is t , v_a established a relationship with v_b , which is called logical interaction. v_a interacted with v_b for real, which is called actual interaction. The logical and actual interactions are referred to as interactions. The graph formed by the logical interaction and its vertices is called the logical interaction graph. The graph formed by the actual interaction and its vertices is called the actual interaction graph, which is shown in Figure 2.

3.3. Generating Cross-Platform Robust Timestamp Sequence.

User behavior data can be obtained from the corresponding social networks by accounts, and the time interval p of the interaction behaviors of the regular user is extracted. The frequency of the interaction interval within p is statistically distributed in minutes, and the time interval sequence F to be interacted is obtained by p . F is a two-dimensional array, the dimension represents the corresponding social networks, and the second dimension stores the timestamps of specific social network interaction behaviors. Before sending a secret message, the interaction delay of the currently used social network is constantly sensed and the maximum interaction delay d corresponds to the number of redundancies in the redundancy mapping over time.

Get the current time and convert it to the timestamp form t_c . The mapping between V_s and timestamp can be constructed by R to obtain the timestamp index. The timestamp of the interaction behavior is used to specify the time when the behavior was generated, and its sequence is denoted by T_s . T_s is generated by Algorithm 3, which has an initial value of t_c .

Algorithm 3 uses the method of adding redundant mapping to increase the robustness of this method. $\forall t \in T_s$, the algorithm executes the preset interactive behavior at t ; then the secret timestamp can be posted to the social networks. Combined with the generated timestamp sequence and the posts crawled in Step 4, release the secret message graph to the social network at the corresponding timestamp.

3.4. Reconstruction of the Directed Secret Message Graph.

Initialize an empty interaction sequence, and, from the timestamps that have been arranged in ascending order in Step 9, the timestamp index s can be calculated according to the following equation:

$$s = \lfloor \frac{t}{(d+1)} \rfloor \% n. \quad (9)$$

For any interaction element $\langle v_j, v_k \rangle$, j is the index of the behavioral initiator in V_r , and k is the index of s in S . Get the interaction element $\langle v_j, v_k \rangle$ from j, k and append it to E_r . Similarly, the elements in T_r are executed in sequence to obtain the interaction sequence E_r , which reconstructs the secret message graph. Combining the subscripts of the


```

Input:  $V_s, E_s, t_c, d, S, F, p$ 
Output:  $T_s$ 
(1)  $l = \text{len}(E_s)$ 
(2)  $n = \text{len}(V_s)$ 
(3) for  $j = 0$  to  $p - 1$  do
(4)   for  $i = 0$  to  $l - 1$  do
(5)      $e = E_s[i]$ 
(6)     receiver_index =  $e[v_j]$  //Get the receiving account of the interaction sequence
(7)     tsi =  $S[\text{receiver\_index}]$ 
(8)     repeat
(9)        $t_c = t_c + F[j][i]$ 
(10)      if  $i == 1$  and  $[t_c/d]\%n == \text{tsi}$  then
(11)         $T_s.append(t_c)$ 
(12)      if else  $\{ [t_c/(d + 1)] \%n == \text{tsi}$  and  $T_s[i] > T_s[i + 1] \}$ 
(13)         $T_s.append(t_c)$ 
(14)      else
(15)         $t_c = t_c + 1$ 
(16)      end if
(17)    until  $i == l - 1$ 
(18)     $E_s.append(\langle v_j, v_k \rangle)$  //The append function means append the current element to the sequence  $E_s$ 
(19)  end for
(20) end for
(21) return  $T_s$ 

```

ALGORITHM 3: Cross-platform robust timestamp sequence generation algorithm.

sending and receiving account numbers of the elements of E_r , the decimal information carried by this sequence can be determined from \mathbf{I} in Step 6. For any interaction element $\langle v_j, v_k \rangle$, the decimal data value transmitted can be determined from the following equation:

$$\text{value} = \mathbf{I}[j, k]. \quad (10)$$

Using equation (10), the elements of sequence E_r are executed sequentially, and the obtained values are appended to the initially empty D_r sequence in turn. Then, D_r is converted into binary B_r and spliced into S_r . Finally, the secret message transmitted can be extracted.

$$I = \begin{bmatrix} 20 & 31 & 6 & 11 & 30 & 23 \\ -1 & 16 & 18 & 1 & 29 & -1 \\ 14 & -1 & 21 & -1 & 26 & 0 \\ 24 & 27 & 2 & 7 & 12 & 8 \\ 5 & 28 & 17 & 10 & 15 & 4 \\ 22 & 9 & 13 & 3 & 25 & 19 \end{bmatrix}. \quad (11)$$

3.5. Example. To make the proposed method easier to understand, this section gives an example to briefly describe the process of sending, using plaintext to deliver the secret message. The secret message sent is “hello”; $V_s = \{v_0, v_1, \dots, v_5\}$; the key is “secret”; $d = 2$; $t_c = 1614704185$. Thus, we can get $n = |V_s| = 6$; then $l_b = 5$, $m = 32$, and $S = [s_0, s_1, \dots, s_5] = [0, 1, \dots, 5]$. When sending a secret message, the secret message is firstly converted to binary, divided into a group of every l_b bits, and then

converted to decimal, and the conversion process is shown in Table 2. Then, generate a random sequence S_R of length m , assuming that at this time $S_R = [17, 9, 20, 33, 29, 24, 2, 21, 23, 31, 27, 3, 22, 32, 12, 28, 7, 26, 8, 35, 0, 14, 30, 5, 18, 34, 16, 19, 25, 10, 4, 1]$, through the parameters of Algorithm 1, using Algorithm 1 to obtain \mathbf{I} , as shown in equation (11). For example, the value of index 0, element 17 in S_R , calculated by Algorithm 1, is row = 2 and column = 5, so the element in row 2 and column 5 of \mathbf{I} is 0.

Then, through equation (11) and Algorithm 2, the interaction sequence E_s is generated. For example, the element with index 0 in sequence D_s is 13, and its row = 5 and column = 2 in equation (11). Therefore, $\langle v_5, v_2 \rangle$ can be generated, which is shown in Table 3, and the constructed directed interaction diagram is shown in Figure 3(a).

Next, the timestamp sequence T_s is generated by Algorithm 3. Here, take $e_0 = \langle v_5, v_2 \rangle$ as an example, and briefly describe its generation process. $\langle v_5, v_2 \rangle$ means that v_5 needs to interact with v_2 , and the subscript value 2 of v_2 is obtained by equation (9). The index of the timestamp corresponding to 1614704190 is 2. Therefore, v_5 performs a certain interaction behavior at timestamp 1614704190, which means that $\langle v_5, v_2 \rangle$ has a virtual interaction. The rest of the elements are calculated in the same way as shown in Table 3. The secret message graph is constructed based on E_s and V_s , as shown in Figure 3(b). Finally, by combining T_s and the post message, the secret message graph is released to multiple social networks at the corresponding timestamps.

The actual interaction graph generated by sending “hello” is shown in Figure 4. The red timestamp in the figure indicates that the interactive behavior is delayed, the green timestamp indicates normal sending, and the orange social network icon indicates that it is not under the control of the sender account.

TABLE 2: Process of converting plaintext to decimal.

Secret message	h	e	l	l	o			
A_s	01101000	01100101	01101100	01101100	01101111			
S_s	0110100001100101011011000110110001101111							
B_s	01101	00001	10010	10110	11000	11011	00011	01111
D_s	13	1	18	22	24	27	3	15

TABLE 3: Interaction diagram and timestamp sequence generation process.

D_s	13	1	18	22	24	27	3	15
Er	$\langle v_5, v_2 \rangle$	$\langle v_1, v_3 \rangle$	$\langle v_1, v_2 \rangle$	$\langle v_5, v_0 \rangle$	$\langle v_3, v_0 \rangle$	$\langle v_3, v_1 \rangle$	$\langle v_5, v_3 \rangle$	$\langle v_4, v_4 \rangle$
Ts	1614704190	1614704193	1614704208	1614704220	1614704238	1614704241	1614704247	1614704250
S_N	Facebook	Twitter	Twitter	Facebook	MicroBlog	MicroBlog	Facebook	Twitter

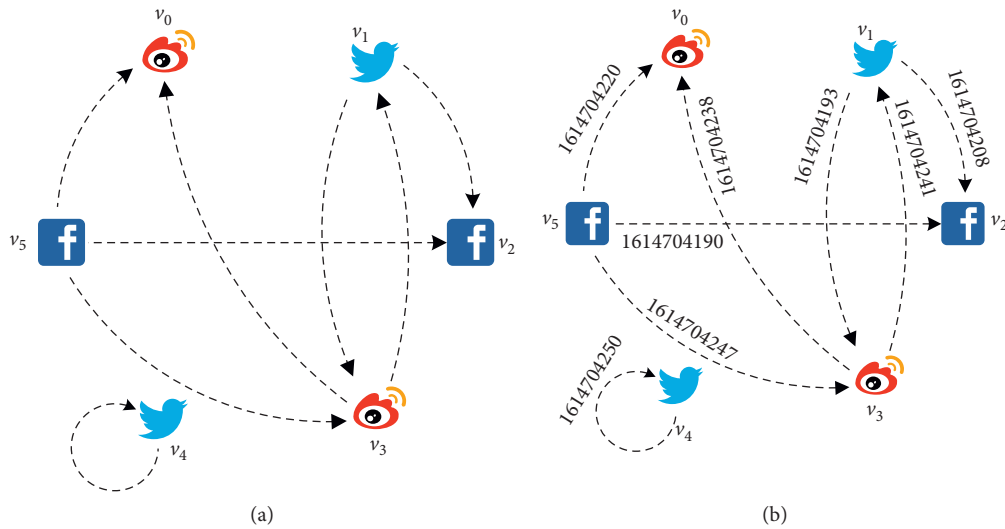


FIGURE 3: Logical interaction graph. (a) Logical interaction graph without timestamps. (b) Logical interaction graph with timestamps.

According to Figure 4, we can extract the timestamp and other messages. The extracting process is the inverse of the embedding process and will not be repeated here.

4. Experiments and Evaluation

To evaluate the performance of the proposed method, we implemented the methods in [21–23] and implemented the method proposed in this paper using Python.

4.1. Experimental Settings. The experiment uses *Gone with the Wind* as the secret message, “world” as the key, and 6 accounts on 3 platforms as the sending account. Therefore, $n = |V| = 6$, and then $l_b = 5$ and $m = 32$. To determine the maximum interaction delay, we measured the time delay of Facebook, Twitter, and Weibo within 2 hours before the experiment. The specific method is as follows: use an automated tool to automatically perform a certain behavior at the expected timestamp. Then the time data that have been extracted on social networks can be converted to a timestamp form. The corresponding results are shown in Figure 5. Through observation, we have conducted more than 200 interactions within 2 hours, and the maximum

interaction delay is $d = 4$. In addition, Figure 6(a) shown by the interaction patterns of regular users on social networks can determine $p = [0, 16]$.

4.2. Comparative Experiments and Evaluation Related to Embedding Capacity. There are 4 groups of experiments in this subsection. The first group of experiments provided data support for the three following groups of experiments.

4.2.1. Comparative Experiments and Evaluation on the Average Number of Bits Carried by a Single Behavior. When n and the secret message are determined, the length of secret message required in [21–23] is determined. Starting from the first character of *Gone with the Wind*, we select the character length of the secret message required for the 4 methods. The number of bits carried by a single behavior obtained by delivering a secret message once has large randomness. To ensure that the average number of bits carried by a single behavior is representative, the average number of bits carried by a single behavior is calculated by sending the secret message 500 and 1000 times, respectively, and is denoted as l_s . According to the definition of embedding capacity in Section 2,

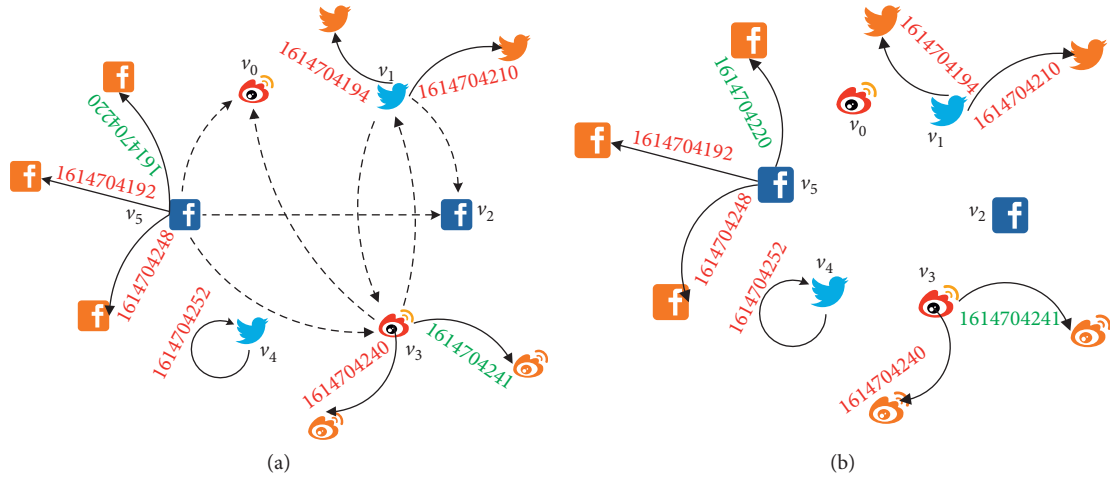


FIGURE 4: Actual interaction graph. (a) Mixed graph of logical and actual interactions. (b) Actual interactions graph.

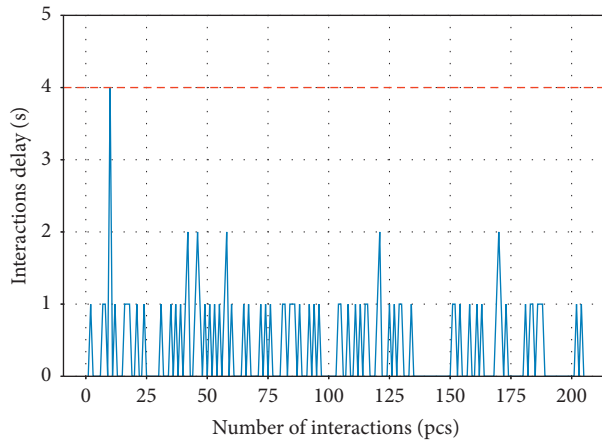


FIGURE 5: Interaction delay over a period of time.

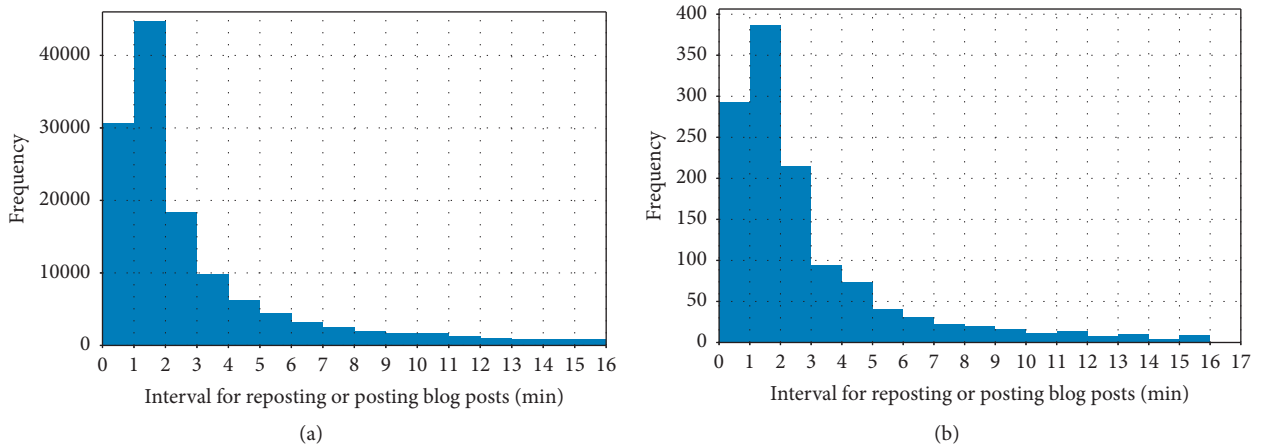


FIGURE 6: Interaction patterns. (a) Interaction patterns of ordinary users. (b) Interaction patterns of the proposed method.

l_s represents the embedding capacity of a single behavior. If the difference in the number of sendings is large and there is no significant difference in l_s , it means that l_s has stabilized and is representative.

Figures 7(a) and 7(b) show l_s calculated when $n \in [7, 16]$ and the secret message is sent 500 times. It shows that l_s of our method is higher in comparison to other methods. Besides, l_s is stable when sending 500 and 1000

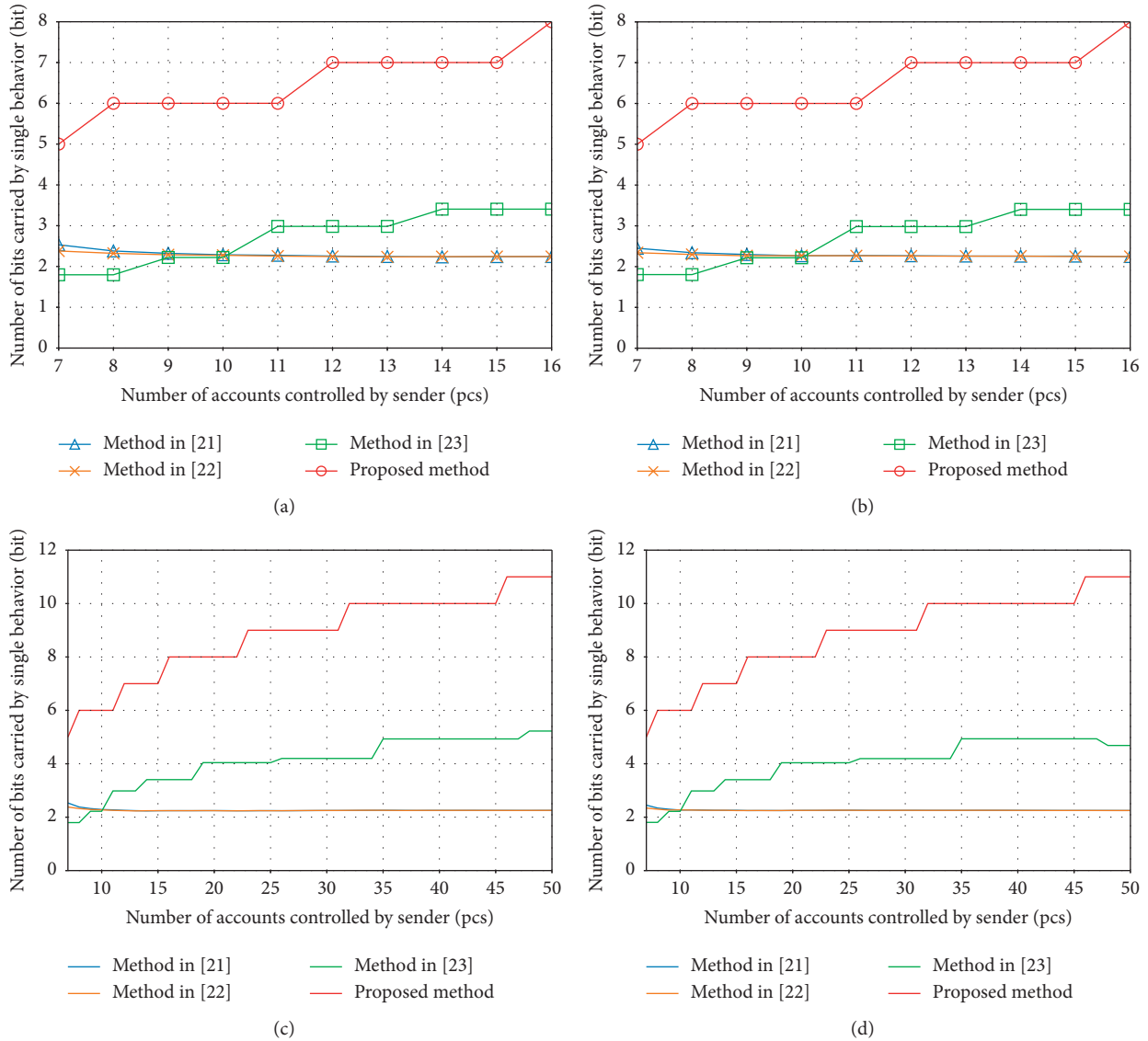


FIGURE 7: Comparison experiment of the average number of bits carried by a single behavior. (a) The average number of bits carried by a single behavior when $n \in [7, 16]$ and 500 messages are sent. (b) The average number of bits carried by a single behavior when $n \in [7, 16]$ and 1000 messages are sent. (c) The average number of bits carried by a single behavior when $n \in [7, 50]$ and 500 messages are sent. (d) The average number of bits carried by a single behavior when $n \in [7, 50]$ and 1000 messages are sent.

secret messages, respectively. Therefore, l_s is representative when sending 1000 secret messages. The following experiments will be based on this data.

To observe the trend of l_s under larger n , we did a group of experiments at $n \in [7, 50]$, which are shown in Figures 7(c) and 7(d). Figures 7(c) and 7(d) show that our method is still higher than other compared methods in the larger n range. As n increases, the superiority of the proposed method becomes more prominent.

4.2.2. Comparative Experiments and Evaluation on Embedding Capacity. Figure 8(a) shows the embedding capacity under different n when the number of behaviors is 10, and Figure 8(b) shows the embedding capacity trend of n in a larger range. Although they have the same

trend as Figure 7, they have different meanings. For example, when $n = 35$, the embedding capacity of our method is 100 bits.

Table 4 shows the embedding capacity of the four methods when the number of behaviors is 10, and n is 10, 15, 20, 25, 30, 35, 40, 45, and 50, respectively. Compared with other methods, our method improves the embedding capacity. Table 4 shows that when n is 10, 15, 20, 25, 30, 35, 40, 45, and 50, as n increases, the embedding capacity shows an upward trend. When n is 20, the rate of the embedding capacity between our method and the compared method has a minimum of 1.98. When n is 50, the rate of embedding capacity between this method and the compared method has a maximum of 4.89. When n is the else value, the rate of the embedding capacity between our method and the compared method is between 1.98 and 4.89. Therefore, the

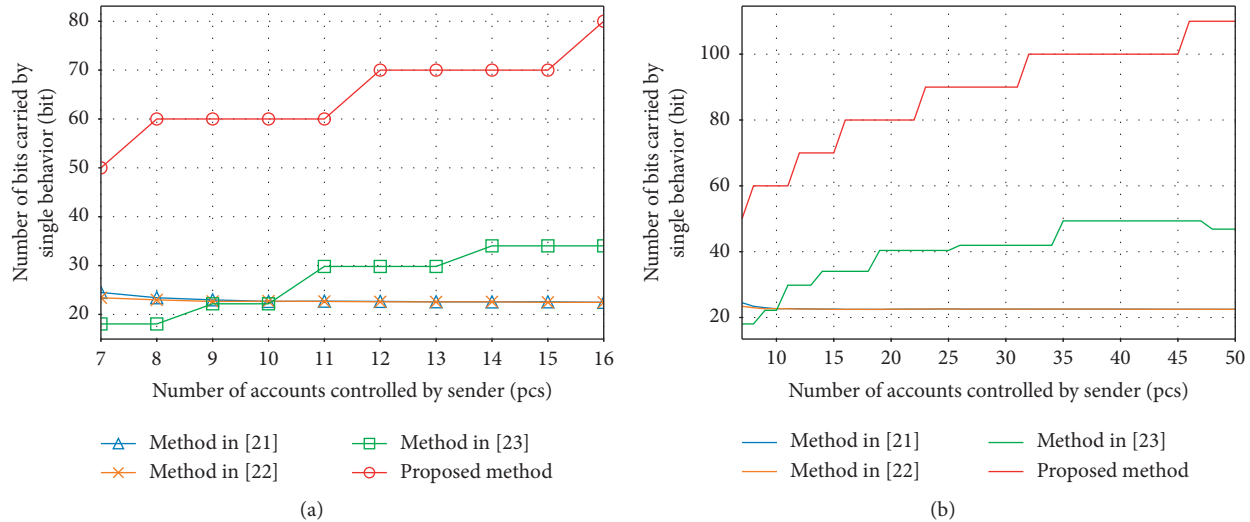


FIGURE 8: The embedded capacity in 10 interaction behaviors. (a) The embedding capacity of $n \in [7, 16]$. (b) The embedding capacity of $n \in [7, 50]$.

TABLE 4: Comparison data of embedding capacity (bit).

n		10	15	20	25	30	35	40	45	50
Method	Method in [21]	22.71	22.47	22.53	22.58	22.56	22.56	22.53	22.5	22.49
	Method in [22]	22.66	22.57	22.49	22.59	22.55	22.55	22.55	22.5	22.49
	Method in [23]	22.17	34.03	40.37	40.37	41.93	49.35	49.35	49.35	46.85
Proposed method		60	70	80	90	90	100	100	100	110
Minimum multiple of boost		2.64	2.06	1.98	2.23	2.14	2.02	2.02	2.03	2.34
Maximum multiple of improvement		2.7	3.12	3.56	3.99	3.99	4.43	4.44	4.44	4.89

experimental results show that the embedding capacity in our paper is significantly higher in comparison to the other methods. When $n \in [10, 50]$ and the step size is 5, it increases by 1.98 to 4.89 times.

4.2.3. Comparative Experiments and Evaluation on the Number of Behaviors. Suppose that the sender intends to send l_p binary bits, and the steganography method can send l_r binary bits each time. If $l_p < l_r$, the l_p binary sent is called the valid sent bits, and the remaining $l_r - l_p$ binary bits are invalid sent bits.

For the methods proposed in [21–23] and our method, when l_p bits are sent and l_p is not an integer multiple of l_b , $l_r = l_b - l_p \% l_b$ bits. The number of bits l_p sent by the sender may not be equal to l_r . In these four methods, 7 social accounts are controlled and 8-bit information needs to be sent. Our method [21–23] needs to be appended with 2, 7, 13, and 17 invalid bits, respectively, to convey the secret message. From this, the effective bit rate $r_b = l_p / l_r$ can be calculated as 0.8, 0.53, 0.38, and 0.33 for sending effective bits, respectively.

Figure 9 shows the number of behaviors generated by sending the same effective number of bits when n is 7. It shows that our method sends the same effective number of

bits, and the number of behaviors generated is significantly lower compared to the other three methods. When using a suitable time interval to perform interactive behaviors, our method will consume less time. Figure 9(b) shows that as the number of effective bits sent increases, the number of behaviors of our method is still gradually increasing, which is smaller than the compared method. Compared with the other three methods, our method generates fewer behaviors when sending the same effective number of bits, and the effective bit rate is higher. At the same time, Figure 9 shows that our method is more flexible in sending message.

4.2.4. Comparative Experiments and Evaluation on Embedding Rate. The embedding rate in this paper refers to the number of bits that each account can carry; that is, embedding rate = c/n . The embedding capacity in this subsection is calculated when the number of behaviors is 10, and the embedding rate is shown in Figure 10. In Figure 10(a), the number of n ranges from 7 to 16. The embedding rate of our method is much higher compared to the other methods. In Figure 10(b), as n increases, the embedding rate generally shows a downward trend, but the embedding rate of our method is higher compared to the other methods.

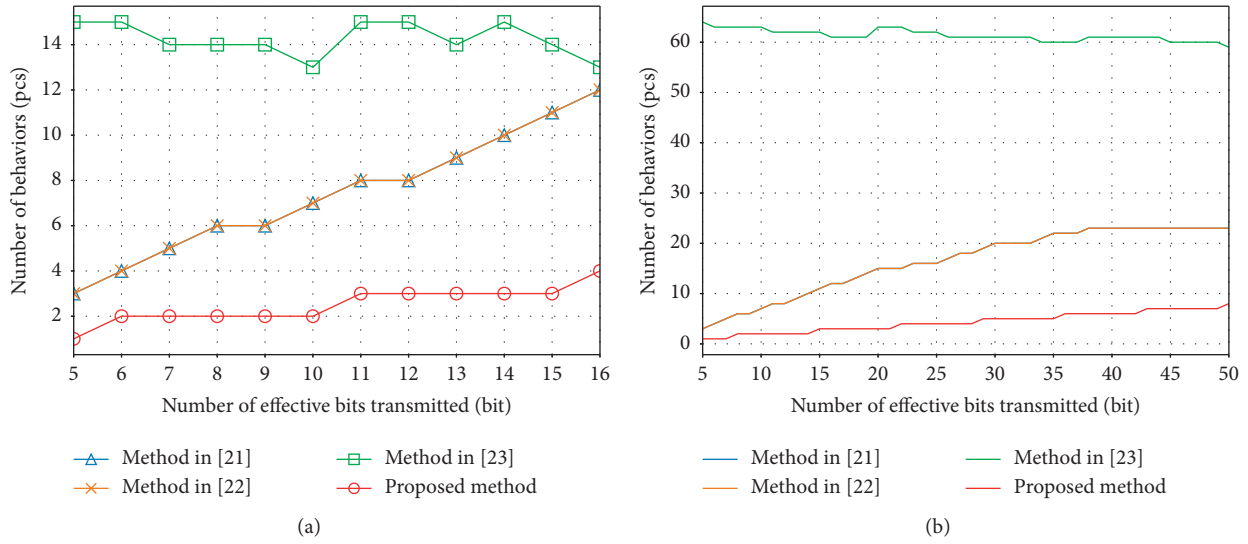


FIGURE 9: The number of behaviors generated under transmission of different effective bits. (a) The number of behaviors generated by the number of effective bits of [5, 16] when $n=7$. (b) The number of behaviors generated by the number of effective bits of [5, 50] when $n=7$.

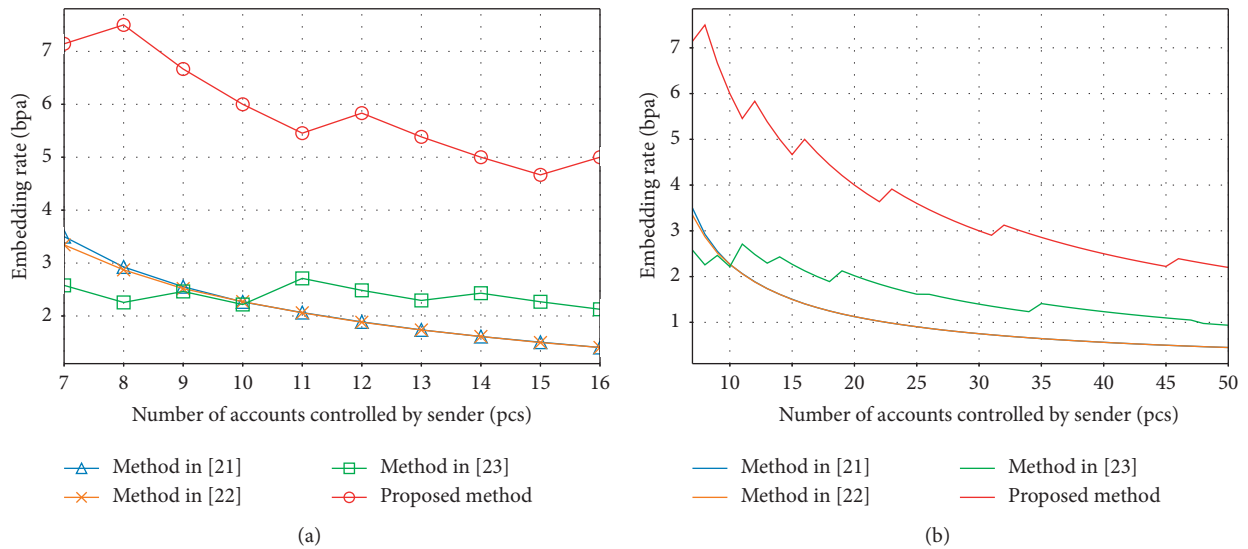


FIGURE 10: Embedding rate when the number of behaviors is 10. (a) Embedding rate when the number of behaviors is 10, $n \in [7, 16]$. (b) Embedding rate when the number of behaviors is 10, $n \in [7, 50]$.

This subsection conducts a comparative experiment of the average number of bits carried by a single behavior and conducts an experiment of embedding capacity on the basis of the average number of bits carried by a single behavior. The experimental results show that our method carries out an average number of bits and a large number of bits carried by a single behavior. The embedding capacity of each behavior has better performance. In addition, this subsection also conducts a comparative experiment on the number of behaviors and the embedding rate generated by the transmission of the effective number of bits. The experimental results show that when sending a certain length of effective bit, this method will generate fewer behaviors. Meanwhile, this method has a higher embedding rate for each account that sends secret message; that is, each account can carry more information.

4.3. Experiments and Evaluation on Detection Resistance. References [7, 16, 18] collect and fit the user’s behavior frequency to avoid anomalies caused by the sender’s behaviors. For this reason, we take the same method to ensure resistance to detection. Firstly, we crawled more than 160,000 posts from 94 bloggers in the “entertainment” section of public social networks and calculated the time interval between two adjacent posts or retweeted by the same blogger. Then, we merged the time interval data of all bloggers and sorted them and removed the top 10% and bottom 10% data to plot Figure 6(a). The time interval between retweets or posts on social networks is consistent with the frequency of interactions, and anomalous behavior can be avoided if the time interval between interactions on social networks is consistent with this pattern.

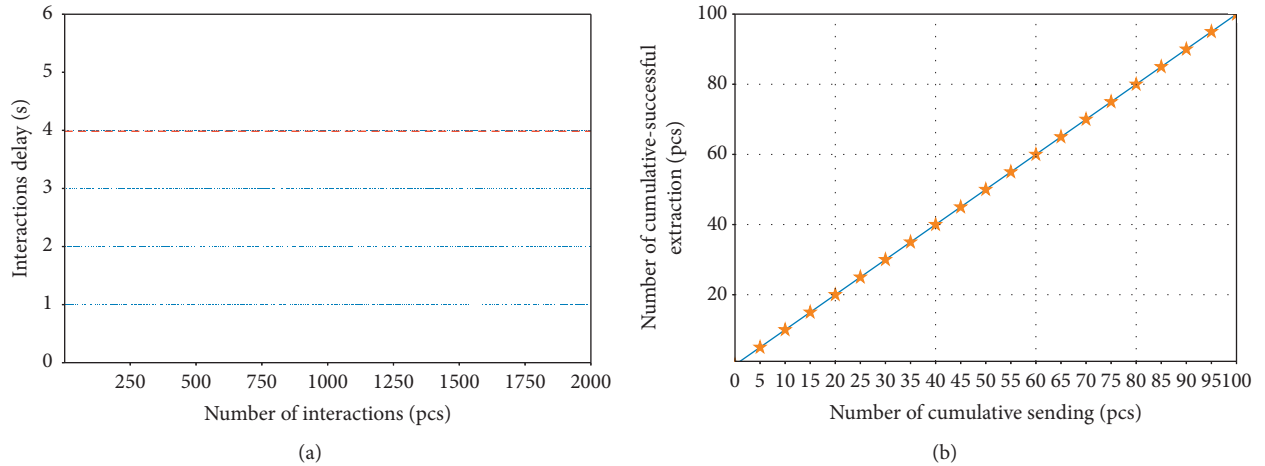


FIGURE 11: Robustness performance of the proposed method. (a) Interaction delay of the proposed method. (b) Extraction effect of the proposed method.

TABLE 5: Average time consumed to send 100 bits of data with different behavioral delay(s).

	$n = 7$			$n = 11$			$n = 15$		
Delay	1st group	2nd group	3rd group	1st group	2nd group	3rd group	1st group	2nd group	3rd group
$d = 2$	547.08	561.07	584.10	323.15	341.53	359.12	242.97	262.33	286.41
$d = 3$	554.20	569.94	587.16	331.38	348.69	364.47	247.34	265.33	284.78
$d = 4$	562.41	578.54	597.31	340.98	360.38	377.56	255.19	274.77	290.72

This experiment selects the first paragraph of text from *Gone with the Wind* as a secret message to send. Figure 6(a) shows the patterns of the interaction behaviors of ordinary users on social networks, and Figure 6(b) shows the patterns of the interactive behavior generated by our method. Based on Figure 5 of Section 4.1, $d = 4$ is the interaction time delay. As shown in Figure 6(b), the difference of the timestamps of the interaction behaviors during the secret message transmission is plotted as a frequency, and the time interval for generating the interaction behaviors is $[0, 16]$. The trend in Figure 6(b) is generally consistent with Figure 6(a) and the interval is also consistent, which indicates that the interaction behaviors generated by our method are consistent with the interaction behaviors of social network users. Therefore, our method can resist the analysis and detection of attackers in terms of behavioral anomalies.

4.4. Experiments and Evaluation on Robustness. Robustness is a measure of the stability of a method. When the carrier data passes through a lossy channel, the channel or the attacker may destroy the carrier data. The execution time of the behavior may be influenced by various factors, for which we need to verify the robustness of the method. In this paper, we ensure the robustness of the proposed method by sensing the interaction delay and using redundancy control mechanism. This method allows the receiver to extract the secret message correctly even if there is an interaction delay when the sender delivers the secret message. To verify the robustness of our method, we have done the following experiments.

From *Gone with the Wind*, part of the data is selected and sent 100 times as secret messages, and the maximum interaction delay $d = 4$ can be seen from Figure 5 in Section 4.1. A total of 24,800 behaviors are generated during the sending process, and the interaction delays of the first 2,000 behaviors are selected and plotted in Figure 11(a). Figure 11(b) shows the relationship between the cumulative number of secret messages sent and the cumulative number of successful extractions.

Figure 11(a) shows that the interaction delay during the sending process does not exceed 4 seconds. From Figure 11(a), it can be observed that when $d = 4$, the secret message sent is always extracted correctly. This implies that the interaction delay is less than 4 seconds during the period when the secret message is sent. Otherwise, it is difficult for all secret messages to be extracted correctly. It also indicates that the redundancy number of redundant mappings can resist the effect of interaction delay on the correct extraction of secret messages. Figures 11(a) and 11(b) show that our method can resist the effect of interaction delay, which shows our method is robust.

4.5. Experiments and Evaluation on Time Consumption. To avoid behavioral anomalies, the behavioral frequency of some users needs to be fitted when sending secret messages, which requires a certain time sacrifice. In addition, the interaction delay can also have an impact on the consumed time. To examine the impact of different interaction delays on time consumption when sending secret messages of a certain length, we designed a set of experiments. The experiments consider the effect of interaction delay and the

number of accounts on the time consumed. Firstly, 100 sentences from *Gone with the Wind* were selected as secret messages and the time consumed was counted. Then, it was divided into 3 groups and the average time was calculated. Finally, the consumed time is converted uniformly to the time consumed when sending 100 bits, and the statistics are shown in Table 5.

When $d = 2$ and $n = 7$, it consumed 547.08 seconds on average after the first group of messages. When $d = 2$ and $n = 11$, it consumed 323.15 seconds on average after the first group of messages. When $d = 3$ and $n = 7$, it consumed 554.20 seconds on average after the first group of messages. Table 5 shows that the time consumed increases as the delay increases and decreases as the number of accounts increases. The reason for the above situation is that as the delay increases, the number of redundant mappings goes up and takes up more time. When the number of accounts increases, the number of simultaneously generated behaviors also increases, which will consume less time.

5. Conclusion

To enhance the communication security of edge IoT devices, this paper proposes a behavioral steganography method based on timestamp modulation. It utilizes timestamps under social networks to achieve cross-platform covert communication and uses directed graphs to encode secret messages to improve the embedding capacity. At the same time, the method reduces the number of behaviors generated on a single platform by spreading the secret message across multiple social networks, which reduces the probability of an attacker discovering the covert communication. The experimental results show that the proposed method is highly robust and resistant to detection, and the embedding capacity, the number of effective bits transmitted, and the embedding rate are better than those of the compared methods. However, there is still a gap in embedding capacity between social network-based behavioral steganography and traditional methods. In future research, we will continue to focus on and follow up the privacy protection in SIoT and further improve the embedding capacity of behavioral steganography.

Data Availability

Some or all data, models, or codes that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Mingliang Zhang mainly proposed the research ideas, completed the coding of the experiments, visualized the results of the experiments, and completed the original manuscript. Xiangyang Luo mainly developed the proposed method, directed the experiments, reviewed and corrected

the manuscript, and funded this work. Pei Zhang mainly directed the color matching of the graphs, collected the experimental data, and reviewed and corrected the manuscript. Hao Li mainly provided the data needed for the experiments, investigated the progress of research in related works, and reviewed and corrected the manuscript. Yi Zhang mainly verified the experimental results, investigated the research progress of related directions, and reviewed and corrected the manuscript. Lingling Li mainly directed the experiments, corrected the constant form in the algorithm to the variable form to ensure the correctness of the algorithm, and reviewed and corrected the manuscript.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. U1804263, 62172435 and 62002387) and the Zhongyuan Science and Technology Innovation Leading Talent Project of China (Grant no. 214200510019).

References

- [1] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The Social Internet of Things (SIoT)—when social networks meet the internet of things: concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.
- [2] A. A. Abd El-Latif, B. Abd-El-Atty, and S. E. Venegas-Andraca, "A novel image steganography technique based on quantum substitution boxes," *Optics & Laser Technology*, vol. 116, pp. 92–102, 2019.
- [3] S. K. Ghosal, J. K. Mandal, and R. Sarkar, "High payload image steganography based on laplacian of Gaussian (LoG) edge detector," *Multimedia Tools and Applications*, vol. 77, no. 23, Article ID 30403, 2018.
- [4] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, 2017.
- [5] J. Fridrich, M. Goljan, and R. Du, "Reliable detection of LSB steganography in color and grayscale images," in *Proceedings of the 2001 workshop on Multimedia and Security: New Challenges*, pp. 27–30, Ottawa, Canada, October 2001.
- [6] B. Guo, Y. Ding, L. Yao, Y. Liang, and Z. Yu, "The future of false information detection on social media," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–36, 2020.
- [7] Z. Yang, Y. Hu, Y. Huang, and Y. Zhang, "Behavioral security in covert communication systems," in *Proceedings of the 18th International Workshop on Digital Watermarking*, pp. 377–392, Chengdu, China, November 2019.
- [8] A. Wilson, P. Blunsom, and A. Ker, "Linguistic steganography on twitter: hierarchical language modeling with manual interaction," in *Proceedings of the IS&T Electronic Imaging on Media Watermarking, Security, and Forensics*, San Francisco, CA, USA, February 2014.
- [9] F. Bertini, S. G. Rizzo, and D. Montesi, "Can information hiding in social media posts represent a threat?" *Computer*, vol. 52, no. 10, pp. 52–60, 2019.
- [10] S. Mahato, D. K. Yadav, and D. A. Khan, "A novel information hiding scheme based on social networking site viewers' public comments," *Journal of Information Security and Applications*, vol. 47, pp. 275–283, 2019.

- [11] H. Kang, H. Wu, and X. Zhang, "Generative text steganography based on LSTM network and attention mechanism with keywords, Electronic Imaging," in *Proceedings of the IS&T Electronic Imaging on Media Watermarking, Security, and Forensics*, no. 4, Burlingame, CA, USA, January 2020.
- [12] S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, and N. Borisov, "Stegobot: a covert social network botnet, Information Hiding," in *Proceedings of the 13th International Workshop on Information Hiding*, pp. 299–313, Prague, Czech Republic, May 2011.
- [13] K. Muhammad, J. Ahmad, S. Rho, and S. W. Baik, "Image steganography for authenticity of visual contents in social networks," *Multimedia Tools and Applications*, vol. 76, no. 18, pp. 18985–19004, 2017.
- [14] K. Muhammad, M. Sajjad, I. Mehmood, S. Rho, and S. W. Baik, "Image steganography using uncorrelated color space and its application for security of visual contents in online social networks," *Future Generation Computer Systems*, vol. 86, pp. 951–960, 2018.
- [15] J. Tao, S. Li, X. Zhang, and Z. Wang, "Towards robust image steganography," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 594–600, 2018.
- [16] N. Pantic and M. Husain, "Covert botnet command and control using twitter," in *Proceedings of the 31st Annual Computer Security Applications Conference*, pp. 171–180, Los Angeles, CA, USA, December 2015.
- [17] X. Zhang, "Behavior steganography in social network," in *Proceedings of the 12th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 21–23, Kaohsiung, China, November 2016.
- [18] Y. Hu, Z. Wang, and X. Zhang, "Steganography in social networks based on behavioral correlation," *IETE Technical Review*, vol. 38, no. 1, pp. 1–7, 2021.
- [19] S. Li, A. T. S. Ho, Z. Wang, and X. Zhang, "Lost in the digital wild: hiding information in digital activities," in *Proceedings of the 2nd International Workshop on Multimedia Privacy and Security*, pp. 27–37, Toronto, Canada, October 2018.
- [20] C. S. Collberg and C. Thomborson, "Watermarking, tamper-proofing, and obfuscation - tools for software protection," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 735–746, 2002.
- [21] I. Nechta, "Steganography in social networks," in *Proceedings of the 2017 Siberian Symposium on Data Science and Engineering*, pp. 33–35, Novosibirsk, Russia, April 2017.
- [22] H. Wu, W. Wang, J. Dong, and H. Wang, "New graph-theoretic approach to social steganography," in *Proceedings of the IS&T Electronic Imaging on Media Watermarking, Security, and Forensics*, Burlingame, CA, USA, January 2019.
- [23] H. Wu, L. Zhou, J. Li, and X. Zhang, "Securing graph steganography over social networks via interaction remapping, Communications in Computer and Information Science," in *Proceedings of the 6th International Conference on Artificial Intelligence and Security*, pp. 303–312, Hohhot, China, July 2020.

Research Article

Database Padding for Dynamic Symmetric Searchable Encryption

Ruizhong Du,^{1,2} Yuqing Zhang ,¹ and Mingyue Li³

¹School of Cyber Security and Computer Science, HeBei University, BaoDing, China

³School of Cyber Security, NanKai University, TianJin, China

²Key Laboratory of High-Confidence Information System of Hebei Province, BaoDing, China

Correspondence should be addressed to Yuqing Zhang; yqzhang@hbu.edu.cn

Received 16 September 2021; Accepted 6 December 2021; Published 31 December 2021

Academic Editor: Qi Jiang

Copyright © 2021 Ruizhong Du et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Dynamic symmetric searchable encryption (DSSE) that enables the search and update of encrypted databases outsourced to cloud servers has recently received widespread attention for leakage-abuse attacks against DSSE. In this paper, we propose a dynamic database padding method to mitigate the threat of data leakage during the update operation of outsourcing data. First, we introduce an outlier detection technology where bogus files are generated for padding according to the outlier factors, hiding the document information currently matching search keywords. Furthermore, we design a new index structure suitable for the padded database using the bitmap index to simplify the update operation of the encrypted index. Finally, we present an application scenario of the padding method and realize a forward and backward privacy DSSE scheme (named PDB-DSSE). The security analysis and simulation results show that our dynamic padding algorithm is suitable for DSSE scheme and PDB-DSSE scheme maintains the security and efficiency of the retrieval and update of the DSSE scheme.

1. Introduction

With the rapid development of Internet technology, the concept of the Internet of Things (IoT) has attracted widespread attention once it was proposed. Due to the computing power bottleneck of IoT terminals, IoT terminal users must rely on cloud platforms to store and process private data. Therefore, the client loses control of private data, which will lead to the risk of leakage of private data (such as medical records and identity information) and endanger the security of the data. The most intuitive way to protect private data is to encrypt it and store it on the cloud platform before outsourcing it. However, it is difficult to search ciphertext data, which impairs data availability. To solve this problem, Song et al. [1] first proposed searchable encryption (SE) technology.

Previous SSE solutions have been static and could not support dynamic database updates, such as addition or deletion of outsourced files, limiting the application of SSE. Therefore, dynamic symmetric searchable encryption (DSSE) technologies were proposed [2]. However, the process of dynamic update will cause data leakage, and some

existing attacks will pose risks [3]. For instance, file injection attacks can obtain the user's query content by injecting a small part of the updated files [4].

To solve the above problems, Stefanov et al. [5] informally introduced the security concepts of forward privacy and backward privacy. Forward privacy is the security involved when adding a document/keyword pair. It ensures that the newly added text will not be searched by the previous search token. Backward privacy is the security involved in the deletion operation. It guarantees that the deleted documents will not be searched. In other words, if, after searching w , adding a document/keyword pair (w, f) and deleting it before the next query, then searching for the keyword w again in the future will not display f . Bost et al. [5] give the formal definition of backward privacy according to the different degrees of leakage. In recent years, many excellent forward and backward privacy DSSE schemes have been designed [6–17]. However, most of the schemes have low search efficiency and involve a complex index update process. Furthermore, leaks caused by access patterns can easily recover the underlying keywords of the search token, and while many schemes use oblivious RAM (ORAM) to solve this problem, ORAM is less efficient.

Database padding technology is a simple and efficient measure to solve the above problems that is suitable for real large data sets. Nevertheless, studies of padding schemes mainly focus on how to design efficient padding methods [3,18,19] that are not suitable for dynamic databases.

1.1. Our Contributions. In this paper, we design a dynamic database padding algorithm that can be applied to DSSE schemes and give the application scenarios. We have implemented a DSSE scheme with forward privacy and backward privacy (named PDB-DSSE) to prove that the dynamic database padding algorithm proposed in this paper is suitable for the DSSE scheme. In summary, our contributions are as follows:

We introduce outlier detection technology to design a dynamic database padding algorithm that is suitable for DSSE schemes. Specifically, each bogus file is sampled and verified by the outlier detection algorithm that pads the database, so that the generated bogus file and the real file are indistinguishable and finally realize result hiding.

We design a new index structure suitable for the padding database using the bitmap index that consists of two parts that represent real files and bogus files, respectively. In the updating process, the index can be modified by homomorphic addition operations, simplifying the update operation of the index.

We instantiated the application scenario of the dynamic padding algorithm, implemented PDB-DSSE, and tested its performance. Experimental results and security analysis show that the dynamic database padding algorithm proposed in this paper is suitable for the DSSE scheme, and PDB-DSSE maintains the security and search and update efficiency of the DSSE scheme.

1.2. Related Works. Song et al. [1] first proposed SSE. After encrypting the keyword, when searching for files matching the keyword w , the algorithm sends the keyword w and its corresponding key k_i of each position i where the keyword may appear to the server. The server compares it with the encrypted keywords one by one to obtain the result. For a file of length N , the encryption and search algorithm of this scheme requires a stream cipher and block cipher. The time complexity is $O(N)$. Subsequently, research on SSE has been extended in many aspects, such as conjunctive search [15], rank search [11], range query [14], small client storage SSE [16], and verifiable schemes [11,12].

To support dynamic updates, DSSE was proposed in [2]. The early DSSE schemes were vulnerable to potential attacks [3], such as file injection attacks [4]. To reduce the threat posed by this attack, the concepts of forward privacy and backward privacy were proposed in [5]. In 2016, Bost [6] gave the formal definition of forward privacy and a forward privacy DSSE scheme $\sum o\phi$ os. This scheme achieves certain improvements in efficiency and security by using simple cryptographic tools (i.e., pseudorandom functions and trapdoor primitives) without relying on the ORAM structure. In the next year, Bost et al. [7] gave a formal definition of backward security according to different levels of leakage and then gave four

forward and backward privacy schemes. Among them, the Fides scheme they designed achieves the second level of backward privacy. The second scheme, Diana, is very efficient but only implements Type-III backward privacy. Diana is modified to a backward privacy scheme $Diana_{del}$ that only requires two roundtrips. The third scheme Janus is the first proposed backward privacy scheme with only a single roundtrip, but this scheme also only implements Type-III backward privacy. The last scheme, Moneta, implements Type-I backward privacy. However, the computational overhead and communication overhead of this scheme are very large due to the use of the TWORAM structure. In 2018, Sun et al. [8] constructed a new primitive called symmetric puncturable encryption (SPE). They further use the common primitive to propose a noninteractive backward privacy DSSE scheme Janus++. This scheme is more efficient than Janus. However, to achieve backward privacy, it hands over much work to clients, greatly increasing their storage and computing overhead.

In 2018, Chamani et al. [9] proposed several dynamic symmetric searchable encryption schemes, namely *Mitra*, *Orion*, and *Horus*. Among them, *Orion* achieves the highest level of backward privacy: Type-I backward. It requires $O(\log N)$ rounds of interaction, and the search process requires $O(n_w \log^2 N)$ steps. *Horus* improves the efficiency of *Orion*. However, it only reaches the third level of backward privacy. In 2019, Zuo et al. [13] designed a forward privacy and backward privacy DSSE scheme that implements Type-I Γ backward privacy that is somewhat stronger than Type-I. To support a larger database, they further extend it to a multiblock setting. Moreover, they extended the first scheme to support range query in [14].

The early studies on database padding focused on the design of an efficient padding method. Cash et al. [3] proposes a heuristic padding method, which pads the number of files corresponding to keywords by multiples. However, this padding method introduces unnecessary padding. In response to the problem of padding overhead, Bost and Fouque [18] divide the clusters according to frequency and pad the clusters with similar frequencies as one cluster. The algorithm achieves the smallest padding cost while preventing counting attacks. Xu et al. [19] proposed the concept of relative entropy to measure the distance between the original keyword distribution and padded keyword distribution and proposed a bogus file generation algorithm to strengthen database padding. As a result, it is almost impossible for us to distinguish a bogus document from a real document.

1.3. Organization. The content structure in the rest of this article is as follows: We give the background knowledge, including the bitmap index structure that we used, the definition of the DSSE and its security model, and the notation used in the work in Section 2. We describe the dynamic padding algorithm in Section 3. We describe the PDB-DSSE scheme in Section 4. The security analysis is given in Section 5. The performance analysis and simulation experiment results are described in Section 6. Finally, Section 7 summarizes the paper.

2. Preliminaries

2.1. Bitmap Index. The index structure of PDB-DSSE is based on the bitmap index [14]. Assuming that the database can store up to N files, each keyword corresponds to an N -bit string S . If the keyword exists in file f_i , the i -th bit of the S is set to 1; otherwise, it is set to 0. As shown in Figure 1(a), the maximum number of files that the database can store is 6, that is, $N=6$. At this time, the bit string $(000100)_2$ indicates that file f_2 exists. As shown in Figure 1(b), if file f_3 is added, a bit string $(001000)_2$ needs to be generated and added to the initial S . If we perform a delete operation to delete the file f_2 , as shown in Figure 1(c), we need to generate $-(000100)_2 = (111100)_2 \bmod 2^6$ and add it to the initial bit string. In our PDB-DSSE scheme, we use the database padding algorithm to pad the database with bogus files and generate the corresponding bitmap index. Assuming that the index length is N , the first k bits represent real files (k is used to indicate the maximum number of files that the real database can store). The $n - k$ bits represent the padded bogus files (as shown in Figure 1(a)). The real database can store up to 4 files, that is, $k = 4$, and the calculation rules are the same as the bitmap index. The detailed algorithm and calculation process are described in Section 3.

2.2. DSSE Definition. In this section, we define the DSSE scheme. We also give its security model and formally define forward privacy and backward privacy.

The database $DB = (f_i, w_i)_{i=1}^k$, where f_i is the file identifier, w_i is the keyword set, and k represents how many real files are stored in DB. $|DB|$ represents the total number of keyword/file identifier pairs. $W = \cup_{i=1}^k w_i$ is used to represent the collection of all distinct keywords in DB.

A DSSE scheme $DSSE=(Setup, Search, Update)$ consists of the following.

Setup (1^λ): this algorithm enters a security parameter λ . It outputs a client's state σ and an encrypted database EDB which is uploaded to the cloud server.

Search ($q, \sigma; EDB$): this protocol requires interaction between the client and the server. The client has stored the state σ and enters the query q . The server retrieves EDB through the search token and returns the matching result to the client.

Update ($\sigma, op, ind; EDB$): this protocol requires interaction between the client and the server. The update operations it supports include add and delete. If the client wants to perform an update operation and add (or delete) a bunch of (w, f) , the server will update EDB and add (or delete) the corresponding ciphertext. At the same time, the client updates the local state σ . Finally, the server updates EDB.

2.3. Security Model. Given a DSSE scheme defined in Section 2.2, we will define the real game REAL and the ideal game IDEAL to give its security model. REAL reflects the behavior of the original DSSE scheme, and IDEAL reflects what the simulator \mathcal{S} does. \mathcal{S} takes the leaked function of the scheme as input. For adversary \mathcal{A} , the leak function is defined as

$\mathcal{L} = (\mathcal{L}^{stup}, \mathcal{L}^{srch}, \mathcal{L}^{updt})$, \mathcal{L}^{stup} being the information that \mathcal{A} can learn when the setup algorithm is executed, \mathcal{L}^{srch} being the information that \mathcal{A} can learn when executing the search protocol, and \mathcal{L}^{updt} being the information that \mathcal{A} can learn when executing the update protocol. Games REAL and IDEAL are defined as follows.

REAL $_{\mathcal{A}}^{DSSE}(\lambda)$: upon input to a database DB that is chosen by the adversary \mathcal{A} , it runs Setup ($1^\lambda, DB$) to obtain the EDB. \mathcal{A} initiates a series of search queries s (or update query (op, ind)). Finally, \mathcal{A} returns the experimental result $b, b \in (0, 1)$.

IDEAL $_{\mathcal{A}, \mathcal{S}}^{DSSE}(\lambda)$: the simulator \mathcal{S} executes the leaked function $\mathcal{L}^{stup}(\lambda, DB)$ for a series of search queries or update queries for the difference (op, ind) of \mathcal{A} , and the simulator \mathcal{S} inputs \mathcal{L}^{srch} and \mathcal{L}^{updt} , respectively, and returns the output result to \mathcal{A} . Finally, \mathcal{A} returns the experimental result $b, b \in (0, 1)$.

Definition 1 (see [14] (adaptive security of DSSE schemes)). A DSSE scheme is \mathcal{L} -adaptively secure with respect to leakage function \mathcal{L} , iff for any probabilistic polynomial time (PPT) adversary \mathcal{A} issuing a polynomial number of queries q , there exists a stateful PPT simulator \mathcal{S} , such that

$$\Pr[\text{REAL}_{\mathcal{A}}^{DSSE}(\lambda) = 1] - \Pr[\text{IDEAL}_{\mathcal{A}, \mathcal{S}}^{DSSE}(\lambda) = 1] \leq \text{negl}(\lambda). \quad (1)$$

2.4. Forward Privacy. Forward security guarantees that the updates that occur cannot be associated with the operations that occurred before.

Definition 2 (see [6] (forward privacy)). If a dynamic searchable encryption scheme D 's update leak function \mathcal{L}^{updt} can be written as

$$\mathcal{L}^{updt}(op, w, S) = \mathcal{L}'(op, S), \quad (2)$$

D is adaptively forward privacy, where w is a set of keywords of update operation and S is the update file index.

2.5. Backward Privacy. Zuo et al. [13] defined Type-I⁻ backward privacy. Type-I⁻ will reveal the file information containing the keyword w , the total number of updates of w , and the update time of each update on w . Our PDB-DSSE scheme uses "0" and "1" bit string to indicate whether the file exists or not, and its operations of addition and deletion use the same module, so it will not reveal when the file was inserted. In summary, our scheme achieves Type-I⁻ backward privacy.

For example, consider the following series of update operations in a single-keyword query system, which occur in sequence: search for the files corresponding to the keyword w at time 1, add file f_1 for the keyword w at time 2, add file f_2 for keyword w at time 3, add file f_3 for keyword w at time 4, delete file f_1 at time 5, and finally at time 6, search again

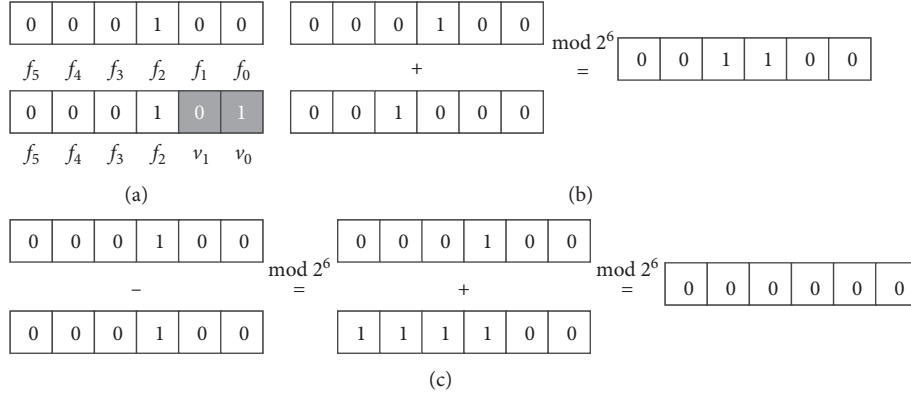


FIGURE 1: Bitmap index. (a) Bitmap index. (b) Add. (c) Delete.

for the document corresponding to the keyword w . Finally, perform the above operations. Type-I⁻ backward privacy revealed that searching for w will return files f_2 and f_3 . And, a total of 4 updates occurred at the above time, which occurred at times 2, 3, 4, and 5.

To formally define Type-I⁻, for the search query list Q and the timestamp t , the search pattern is defined as $\text{sp}(w) = t: (t, w) \in Q'$. The search mode reveals whether the search keyword w is repeated. It is also necessary to define a new leak function TimePDB. For the keyword w , TimePDB(w) lists all the updated timestamps t of w . For update query Q' ,

$$\text{TimePDB}(w) = \{t: (t, \text{op}, (w, S)), Q'\}. \quad (3)$$

Definition 3 (see [13] (backward privacy)). If a dynamic searchable encryption scheme D 's search and update leak functions $\mathcal{L}^{\text{updt}}$ and $\mathcal{L}^{\text{srch}}$ can be written as

$$\begin{aligned} \mathcal{L}^{\text{updt}}(\text{op}, w, S) &= \mathcal{L}'(\text{op}), \\ \mathcal{L}^{\text{srch}}(w) &= \mathcal{L}''(\text{sp}(w), \text{TimePDB}(w)). \end{aligned} \quad (4)$$

D is Type-I⁻ adaptively backward privacy.

2.6. Notation. The notation used in the work is given in Table 1.

3. Dynamic Database Padding Algorithm

To solve the problem of data leakage during the update of DSSE schemes, we introduced outlier detection to design a dynamic database padding algorithm that can be used in the DSSE scheme. The padding algorithm can hide the information of the files currently matching the query keyword. In this scheme, real files and bogus files are represented by bitmap indexes, simplifying the process of update operations.

The dynamic database padding algorithm includes the padding database generation algorithm PDB-Gen and the padding database update algorithm PDB-Update.

3.1. $PDB \leftarrow PDB - \text{Gen}(DB)$. Upon input to a database DB, it outputs a padding database PDB, where $PDB = \{V_1, \dots, V_l\}, 0 \leq l \leq n - k$. The algorithm is described in Algorithm 1. It initializes two empty sets PDB and G that store the number of bogus files that need to be padded for each keyword. For a database DB, we use the same clustering method as [19]. For cluster $\mathcal{C} = \{w_1, \dots, w_{|\mathcal{C}|}\}$, the keywords are sorted by their frequency in ascending order. We pad the keyword counts in a cluster to the same size according to the largest one, calculate the number of bogus files that need to be padded for each keyword, and put the result into set G . Subsequently, we randomly generate bogus files, select those that meet the conditions, and add them to PDB. The bogus file generation algorithm is as follows: we use the m dimension vector to represent bogus file, where m represents the size of the W . If we fill the keyword w_i , then set the i -th bit to 1; otherwise, we set it to 0. Then, we randomly generate a bogus file and judge whether it meets the requirements. Select the τ -bit keyword to fill in the bogus file, where τ is selected from the maximum file size and the maximum file size that the dataset can hold. After generating the bogus file, we use the local outlier factor (LOF) algorithm to detect the outliers of the bogus file [19]. As shown in [19], we should choose a bogus file whose LOF value is approximately equal to 1 for padding. Therefore, we use this threshold to filter samples. If $\text{LOF}(V) < 1$, we add V to the padding database PDB. Otherwise, we roll back the padding counts and reselect the bogus files for detection until all keywords are padded.

3.2. $\widehat{PDB} \leftarrow PDB - \text{Update}(w, \text{index}, k, PDB)$. Upon inputting the keyword w that needs to be updated, the index of the update operation, the number of bits of the real database in the index k , and the padding database PDB, it outputs the updated PDB and the adjusted index. The algorithm is described in Algorithm 2. For the update operation of the padded database, our scheme uses an N -bit bitmap index (Section 2.1) to represent the real files and bogus files, where the first k bits indicate whether the real file exists or not. If file f_i exists, then the i -th bit is set to 1; otherwise, it is set to

TABLE 1: Notation.

Parameter	Notation
λ	The security parameter
F	A secure PRF
n	The length of S
k	The number of files that the database can hold
W	Keyword space $W = \cup_{i=1}^k w_i$
f	The i -th real file ($1 \leq i \leq k$)
V_i	The i -th bogus file ($1 \leq i \leq n - k$)
DB	A database $DB = (f_i, w_i)_{i=1}^k$
PDB	A padding database $PDB = \cup_{i=1}^{n-k} V_i$
EDB	Encrypted database (including DB and PDB)
ST	The current search token for a keyword w
STM	A map that stores ST and the times tm of update operations on keywords in the keyword space W
S	The bit string which used to indicate whether the real file and the bogus file exist
ES	The encrypted S
Sum _{ES}	Sum of the ES
sk	The onetime secret key
Sum _{sk}	Sum of sk

```

(1) PDB, G ← emptymap
(2) for all cluster C:
(3) for  $1 \leq i \leq |w_{|C|}|$  do
(4)   put  $(w_i, |DB(|w_{|C|})| - |DB(w_i)|)$  to G
(5) while G do
(6)    $v \leftarrow 0, \dots, 0_m, \tau \leftarrow [l, u]$ 
(7)   select  $\tau$  different keywords  $w_1, \dots, w_\tau$  randomly
(8)   for  $i = 1: \tau$  do
(9)     if  $G.get(w_i) > 0$  then
(10)       $V[j] = 1$ 
(11)       $G.update(w_i, G.get(w_i) - 1)$ 
(12)     else if  $G.get(w_i) = 0$  then
(13)       select a new  $w_i$  from  $W$ 
(14)     else
(15)       $V[j] = 0$ 
(16)   LOF(V) //validation via LOF detection
(17)   if  $LOF(V) < 1$  then
(18)      $DB \leftarrow PDB \cup \{V\}$  //put bogus files to PDB
(19)   else
(20)     rollback
(21)   return PDB

```

ALGORITHM 1: PDB-Gen(DB).

0. For a bogus file, if the file V_i is padded, the $k + i$ bit of the index is set to 1; otherwise, it is set to 0.

Subsequently, we convert the index into an array and initialize a counter to record the number of files that need to be added or deleted. If the operation is addition, the padding files $PDB(V_w)$ corresponding to the keyword w in PDB need to delete some bogus files corresponding to w . Specifically, we try to change the corresponding position of V_i from 1 to 0. Prior to modifying this position, we perform outlier detection on the changed bogus files V_i . If $LOF(V_i) < 1$, then we modify it and modify the index at the same time. Otherwise, we roll back to the state before the modification and continue to try to modify the next bogus file.

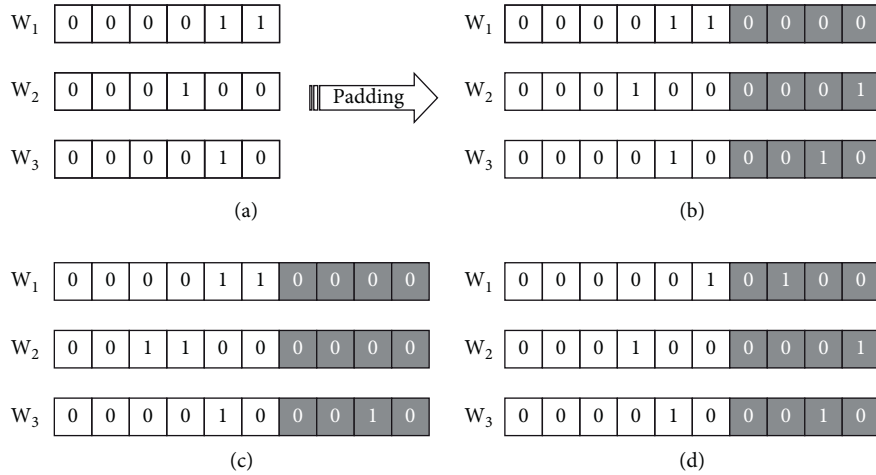
If the operation is deletion, we try to pad files corresponding to keyword w . First, we count the files that need to be deleted, find the bogus file V_i that is not padded with the keyword w in PDB, and modify them one by one. It is worth noting that we try to modify the file corresponding to the position of 0 to 1 and perform outlier detection. If $LOF(V_i) < 1$, we modify the bogus file and index. Otherwise, we roll back to the state before the modification and modify the next file. Finally, the modified index is converted into a sequence and returned.

The corresponding index during the execution of the dynamic padding algorithm is shown in Figure 2 that is explained in plain text. Assuming that the maximum capacity

```

(1) UA  $\leftarrow$  to Array(index)
(2) count  $\leftarrow$  0
(3) (1) op = add
(4)   for  $1 \leq i \leq \text{index.length}$  do
(5)     if UA[i] = 1 then
(6)       count ++
(7)   for  $1 \leq i \leq |\text{PDB}|$  and count > 0 do
(8)     if  $V_i[w] = 1$  then
(9)        $V_i[w] = 0$ //try to modify bogus file
(10)    if  $\text{LOF}(V_i) < 1$  then
(11)      count --
(12)      Add corresponding bit string to UA
(13)    else
(14)       $V_i[w] = 1$ //roll back the modify
(15) (2) op = delete
(16)   for  $1 \leq i \leq \text{index.length}$  do
(17)     if UA[i] = 1 then
(18)       count ++
(19)   for  $1 \leq i \leq |\text{PDB}|$  and count > 0 do
(20)     if  $V_i[w] = 1$  then
(21)        $V_i[w] = 1$ //try to modify bogus file
(22)     if  $\text{LOF}(V_i) < 1$  then
(23)       count --
(24)       Add corresponding bit string to UA
(25)     else
(26)        $V_i[w] = 0$ //roll back the modify
(27)   index  $\leftarrow$  to Array(UA)
(28)   return index

```

ALGORITHM 2: PDB-Update (w ; index; k ; PDB).FIGURE 2: Corresponding index during the execution of the dynamic padding algorithm. (a) Bitmap index. (b) Pad. (c) Add (W_2, f_4). (d) Delete (W_1, f_2).

of the database is 10, the maximum capacity of the real database is 6, that is, $k = 6$, so that the maximum capacity of the PDB is 4. The initial real data are inserted in order $(w_1, f_1), (w_1, f_2), (w_2, f_3)$, and (w_3, f_2) . At this time, the database index is as shown in Figure 2(a). After the PDB-Gen(DB) algorithm is executed, the index is shown in Figure 2(b), where (w_2, V_1) and (w_3, V_2) are padded. If we want to insert (w_2, f_4) , we need to add the bit string $(001000000)_2$ according to the adding rules of the

bitmap index. We search the PDB for the bogus file PDB (V_{w_2}) corresponding to w_2 , and the result is V_1 . Then, deleting V_1 , we need to subtract the bit string $(0000000001)_2$. According to the deletion rule of the bitmap index, this is equivalent to adding the bit string $(111111111)_2$ to obtain the final w_2 new index $(0011000000)_2$, as shown in Figure 2(c). If we perform a delete operation, delete (w_1, f_2) , we need to subtract the bit string $(0000100000)_2$ from the index of w_1 . According to the deletion

rule of the bitmap index, this is equivalent to adding the bit string $(1111100000)_2$ to retrieve all of the files corresponding to the position value of 0 and attempting to modify the bogus files one by one to change the corresponding position 0 to 1. After trying to modify, if outlier detection is performed on V_i and $\text{LOF}(V_i) < 1$, then proceed Modify; otherwise, try to modify the next one. In this example, a bogus file V_3 is added to w_1 and the bit string $(000000100)_2$ needs to be added to obtain the final new index $(0000010100)_2$, as shown in Figure 2(d). The above calculations require modulo 10.

4. PDB-DSSE Scheme

In this section, we construct the PDB-DSSE scheme using the dynamic database padding algorithm proposed in Section 3, where the real files and bogus files are represented by bitmap indexes and symmetric encryption with additive homomorphism [20] is used to encrypt the indexes. We use the framework of [13] combined with our proposed dynamic padding algorithm to prove the feasibility of the dynamic padding algorithm. It specifically consists of the algorithms Setup, Update, and Search.

4.1. $(EDB, \sigma) \leftarrow \text{Setup}(1^\lambda, DB)$. The detailed design of this algorithm is given in Algorithm 3. Input the security parameter λ . The security key K is randomly selected. Generate a security certificate N , where $N = 2^n$. In addition, set the maximum file number k of the database DB, initialize an empty map STM, call the padding database generation algorithm PDB-Gen to generate the padding database PDB, and then initialize a map EDB, where EDB stores encrypted databases (including database DB and padding database PDB) and STM stores ST and the times tm of update operations on keywords in the keyword space W . Finally, send EDB to the server, and the client stores the state $\sigma = (N, k, K, \text{STM})$ secretly.

4.2. $(\sigma', EDB') \leftarrow \text{Update}(w, S, \sigma; EDB)$. This protocol requires interactive execution between the client and the server. Algorithm 4 gives a detailed algorithm design. First, the client enters the update keyword w , status σ , and bit string S . Then, obtain search token ST and the update times tm on keywords in the keyword space W according to STM and use F_k to obtain the key. If it is the first update, initialize the search token ST and the times of updates tm. The hash function H_1 generates the update token UT, the hash function H_2 is used to mask the previous ST, and the hash function H_3 is used to generate a one-time key. According to the bit string that needs to be updated, the algorithm PDB-Update $(w, S, \sigma, \text{PDB})$ is called and the bit string S' used to update the index is returned. Then, S' is encrypted with a simple SE algorithm with additive homomorphism to obtain the ES. Then, the client sends UT, ES, and M_{ST} to the server. Finally, the server updates the ES to ES' .

4.3. $(\sigma', EDB') \leftarrow \text{Search}(s, \sigma; EDB)$. This protocol requires interaction between the client and the server.

Algorithm 5 gives a detailed algorithm design. First, the client enters the keyword w that it wants to search, generates a search token ST and a key k_w , and sends it to the server. The server uses the hash function H_1 to obtain the corresponding update token and accesses the database to obtain the encrypted bit string ES. Then, the server uses simple homomorphic addition to add them to the final result Sum_{ES} and sends it to the client. The client uses the hash function H_3 to obtain the key, calculates the key sum Sum_{sk} , and uses the key to decrypt Sum_{ES} to obtain the bit string S' . After removing the $n - k$ stuffing bits, it outputs the final result bit string S .

5. Security Analysis

In this section, we give the security analysis of our schemes.

PDB-DSSE is forward and backward privacy. Because the hash function inversion operation is almost impossible to complete, the adversary cannot associate the future update with the update before the search. This guarantees forward privacy. For backward privacy, due to the bitmap index, the number of bit strings returned is consistent and the adversary cannot obtain the number of documents currently matching the keyword. Besides, since addition and deletion are done through an algorithm, our solution will not leak the update type. This guarantees backward privacy. In summary, PDB-DSSE is forward and backward privacy.

Definition 4 (adaptive forward and Type-I⁻ backward privacy of PDB-DSSE). Let F be a secure PRF, $\Pi = (\text{Setup}, \text{Enc}, \text{Dec}, \text{Add})$ be a perfectly secure symmetric encryption with homomorphism addition, and RO_1, RO_2 , and RO_3 be random oracles. We define $\mathcal{L}_{\text{PDB-DSSE}} = (\mathcal{L}_{\text{PDB-DSSE}}^{\text{srch}}, \mathcal{L}_{\text{PDB-DSSE}}^{\text{updt}})$, where $\mathcal{L}_{\text{PDB-DSSE}}^{\text{srch}} = (\text{sp}(w), \text{Time PDB}(w))$ and $\mathcal{L}_{\text{PDB-DSSE}}^{\text{updt}} = (\text{op}, w, S)$. PDB-DSSE is $\mathcal{L}_{\text{PDB-DSSE}}$ -adaptive forward and Type-I⁻ backward privacy.

Proof. Similar to [9], we formulate a sequence of games between REAL and IDEAL. Despite the subtle differences between two consecutive games, they are indistinguishable, leading to the conclusion that REAL and IDEAL are indistinguishable. Finally, we use the leakage function defined in Theorem 1 as the input of the simulator to simulate IDEAL:

Game G_0 : G_0 is exactly the same as $\text{REAL}_{\text{DSSE}}^{\text{DSSE}}(\lambda)$ defined in Section 2.3.

Game G_1 : G_1 is identical to G_0 except we do not use a pseudo-random function F to generate the key but randomly select the key with the uniform probability. If w has never been searched, the key is generated and stored in the table key; otherwise, input the keyword w , and return the corresponding key in the table key. G_1 is indistinguishable from G_0 due to the security of the PRF.

Game G_2 : G_2 is the same as G_1 , except that we use a random oracle RO_1 instead of hash function H_1 . During the update process, the function H_1 is used to generate the update token. Instead of using H_1 , we

```

(1)  $K \xleftarrow{\$} \{0, 1\}^\lambda$ ,  $n \leftarrow \text{Setup}(1^\lambda)$ 
(2)  $\text{STM} \leftarrow \text{emptymap}$ 
(3)  $\text{PDB} \leftarrow \text{PDB-Gen}(\text{DB})$ 
(4)  $\text{EDB} \leftarrow \text{Encrypt}(\text{DB}, \text{PDB})$ 
(5) set the max size of DB to  $k$ 
(6) return  $(\text{EDB}, \sigma = (n, k, K, \text{STM}))$ 

```

ALGORITHM 3: PDB-DSSE setup (1^λ , DB).

```

Client:
(1)  $K_w \parallel K'_w \leftarrow F_k(w)$ 
(2)  $(\text{ST}_{\text{tm}}, \text{tm}) \leftarrow \text{STM}[w]$ 
(3) if  $(\text{ST}_{\text{tm}}, \text{tm}) = \perp$  then
(4)  $\text{tm} \leftarrow -1$ ,  $\text{ST}_{\text{tm}} \leftarrow (0, 1)^\lambda$ 
(5)  $\text{ST}_{\text{tm}+1} \leftarrow (0, 1)^\lambda$ 
(6)  $\text{STM}[w] \leftarrow (\text{ST}_{\text{tm}+1}, \text{tm} + 1)$ 
(7)  $\text{UT}_{\text{tm}+1} \leftarrow H_1(K_w, \text{ST}_{\text{tm}+1})$ 
(8)  $M_{\text{ST}_{\text{tm}}} \leftarrow H_2((K_w, \text{ST}_{\text{tm}}) \oplus M_{\text{ST}_{\text{tm}-1}})$ 
(9)  $\text{sk}_{\text{tm}+1} \leftarrow H_3(K'_w, \text{tm} + 1)$ 
(10)  $S' \leftarrow \text{PDB-Update}(w, S, k, \text{PDB})$ 
(11)  $\text{ES}_{\text{tm}+1} \leftarrow \text{Enc}(\text{sk}_{\text{tm}+1}, S', n)$ 
(12) send  $(\text{UT}_{\text{tm}+1}, (\text{ES}_{\text{tm}+1}, M_{\text{ST}_{\text{tm}}}))$  to server
Server:
(13)  $\text{EDB}[\text{UT}_{\text{tm}+1}] \leftarrow (\text{ES}, M_{\text{ST}_{\text{tm}}})$ 

```

ALGORITHM 4: $(\sigma', \text{EDB}') \leftarrow \text{Update}(w, S, \sigma; \text{EDB})$.

```

Client:
(1)  $K_w \parallel K'_w \leftarrow F_k(w)$ ,  $(\text{ST}_{\text{tm}}, \text{tm}) \leftarrow \text{STM}[w]$ 
(2) if  $(\text{ST}_{\text{tm}}, \text{tm}) = \perp$  then
(3) return  $\phi$ 
(4) send  $(K_w, \text{ST}_{\text{tm}}, \text{tm})$  to server
Server:
(5)  $\text{Sum}_{\text{ES}} \leftarrow 0$ 
(6) for  $i = \text{tm}$  to 0 do
(7)  $\text{UT}_i \leftarrow H_1(K_w, \text{ST}_i)$ 
(8)  $(\text{ES}, M_{\text{ST}_{i-1}}) \leftarrow \text{EDB}[\text{UT}_i]$ 
(9)  $\text{Sum}_{\text{ES}} \leftarrow \text{Add}(\text{Sum}_{\text{ES}}, \text{ES}_i, n)$ 
(10) if  $M_{\text{ST}_{i-1}} = \perp$  then
(11) Break
(12)  $\text{ST}_{i-1} \leftarrow H_2((K_w, \text{ST}_i) \oplus M_{\text{ST}_{i-1}})$ 
(13)  $\text{EDB}[\text{UT}_{\text{tm}}] \leftarrow (\text{Sum}_{\text{ES}}, \perp)$ 
(14) send  $\text{Sum}_{\text{ES}}$  to client
Client:
(15)  $\text{Sum}_{\text{sk}} \leftarrow 0$ 
(16) for  $i = \text{tm}$  to 0 do
(17)  $\text{sk}_i \leftarrow H_3(K'_w, i)$ 
(18)  $\text{Sum}_{\text{sk}} \leftarrow \text{Sum}_{\text{sk}} + \text{sk}_i \bmod n$ 
(19)  $S' \leftarrow \text{Dec}(\text{Sum}_{\text{sk}}, \text{Sum}_{\text{ES}}, n)$ 
(20)  $S \leftarrow \text{select first } k \text{ bits of } S'$ 
(21) return  $S$ 

```

ALGORITHM 5: $(\sigma', \text{EDB}') \leftarrow \text{Search}(s, \sigma; \text{EDB})$.

randomly select a string as the update token. In the search process, use the random oracle RO_1 to generate the update token. We also use the random oracle RO_2 and RO_3 similar to RO_1 instead of hash functions H_2 and H_3 . Since a search token is chosen randomly, G_2 is indistinguishable from G_1 .

Game G_3 : G_3 is the same as G_2 , except that we replace the number in each position of the bit string S with 0 and the length of bit string remains the same. Due to the perfect security of simple SE with homomorphic addition, G_3 is indistinguishable from G_2 .

The simulator is the same as G_3 , except that we use the search mode $sp(w)$ instead of the keyword w to simulate the ideal world. During the update process, we chose a new random string for each update in the game G_3 . During the search process, the simulator starts from ST_c and generates new random strings for the previous ST one by one. For the keyword w , \mathcal{S} uses the first timestamp $\hat{w} \leftarrow \text{minsp}(w)$. Then, it uses the random oracle RO_2 to calculate it with the ciphertext c . At the same time, \mathcal{S} calculates S and ST_c through RO_3 and embeds all 0s in the remaining search tokens. Hence, G_3 and simulator are indistinguishable. In addition, what is described in simulator is essentially the game $\text{IDEAL}_{\mathcal{A}, \mathcal{S}}^{\text{DSSE}}(\lambda)$ defined in Section 2.3. Hence, G_3 and $\text{IDEAL}_{\mathcal{A}, \mathcal{S}}^{\text{DSSE}}(\lambda)$ are indistinguishable.

In summary, game $\text{IDEAL}_{\mathcal{A}, \mathcal{S}}^{\text{DSSE}}(\lambda)$ is indistinguishable from G_0 , that is, $\text{REAL}_{\mathcal{A}}^{\text{DSSE}}(\lambda)$, which completes the proof. \square

6. Experimental Analysis

In this section, we present the experimental analysis of our PDB-DSSE scheme, including the comparison with other research results in communication overhead, storage overhead, and the time complexity of query and update.

Table 2 presents the performance comparison between PDB-DSSE and existing research solutions, including security, interaction rounds, and client storage overhead. Here, N is the number of (w, f) in the database, d_w is the number of times the file containing the keyword w is deleted, $|W|$ is the number of all different keywords, and $|D|$ is the number of files stored in the database. Regarding security, as shown in Table 2, PDB-DSSE implements forward privacy and Type-I⁻ backward privacy. Regarding communication overhead, PDB-DSSE only requires one interaction, greatly reducing the communication overhead of the solution. Furthermore, the client storage overhead is $O(1)$ since the client only needs to store the secret key sk and the state σ . Therefore, it is clear that PDB-DSSE only requires one interaction and less client overhead to achieve stronger backward privacy.

Table 3 shows the comparison of the time complexity of the PDB-DSSE scheme with other schemes with stronger backward privacy, Moneta [7], Orion [9], and FB-DSSE [13], where N is the number of (w, f) in the database, f_w is the number of files containing the keyword w , u_w is the number of update operations performed on the keyword w , and t_{ED} is the time it takes for symmetric encryption to perform

encryption and decryption operations. t_{AM} is the calculation time for modular addition.

6.1. Implementation. The simulation experiment of our scheme uses a machine with an AMD Ryzen 7 4800U with a Radeon Graphics 1.8 GHz processor configured with a Windows 10 (64-bit) system and 16 GB RAM. We use the Java language programming to implement our scheme. During the experiments, the number of bits of the bitmap index was adjusted many times to perform experiments to simulate the performance of this scheme under the situation of different maximum file numbers of the database. In our experiments, the update time and search time of the scheme PDB-DSSE were tested when the number of bitmap index bits was 10^2 to 10^9 . We compare our scheme with the Orion scheme [9] that uses the ORAM structure and a higher level of backward security and the FB-DSSE scheme [13] that uses bitmap indexing efficiency. The comparison results are as follows.

Figure 3 shows the search time comparison of PDB-DSSE scheme, Orion scheme [9], and FB-DSSE scheme [13] under different bit lengths that indicate the maximum number of files supported in the scheme. In the Orion scheme, this is represented by the size of $|DB|$. The tested bit length ranges from 10^2 to 10^9 (Orion scheme is $|DB| = 10^2$ to 10^5). Figure 3 shows that the search time of these three schemes increases with increasing bit length ($|DB|$). Since the Orion uses the ORAM structure, its search time complexity is $O(n_w \log^2 N) \cdot t_{SKE}$. When $|DB| = 10^5$, its search time took 96888.2 ms. We can observe from the experimental results that the search efficiency of the ORAM structure is extremely low for large datasets. Through test comparison, it can be found that PDB-DSSE maintains the same search efficiency as the FB-DSSE scheme to a certain extent. In summary, PDB-DSSE has high search efficiency while realizing forward privacy and stronger backward privacy.

Figure 4 shows the comparison of the update time of the PDB-DSSE with Orion [7] and FB-DSSE [13] under different bit lengths (or $|DB|$), where the Orion scheme includes the insertion and deletion times. Since Orion needs to modify the ORAM structure when updating, the insertion time of $|DB| = 10^2$ requires 51.823 ms and the deletion time requires 29.454 ms. In particular, when $|DB| = 10^5$, it takes 3.472 days to build a database with a magnitude of 10^5 and it takes at least 35 days to build a database with a magnitude of 10^6 . Therefore, it is extremely difficult to test the update time when $|DB|$ is greater than 10^6 ; this problem does not exist in PDB-DSSE. We tested the update time of the PDB-DSSE and FB-DSSE schemes with a bit length of 10^2 to 10^9 . We find that the update time of FB-DSSE is stable at 0.2–0.3 ms and the update time of PDB-DSSE is approximately 0.4 ms when the bit length is less than 10^6 . For bit lengths greater than 10^6 , the update time of PDB-DSSE and FB-DSSE increases with increasing bit length. This is because when the bit length is less than 10^6 , the influence of homomorphic addition and modular arithmetic is not significant. The update time complexity of PDB-DSSE is longer than that of the FB-

TABLE 2: Comparison of research results.

Scheme	FP	BP	Roundtrips	Client storage
FIDES [7]	✓	Type-II	2	$O(W \log D)$
DIANA _{del} [6]	✓	Type-III	2	$O(W \log D)$
Janus [6]	✓	Type-III	1	$O(W \log D)$
Janus++ [9]	✓	Type-III	1	$O(W \log D)$
MONETA [6]	✓	Type-I	3	$O(1)$
HORUS [9]	✓	Type-III	$O(\log d_w)$	$O(W \log D)$
ORION [9]	✓	Type-I	$O(\log N)$	$O(1)$
FB-DSSE [13]	✓	Type-I ⁻	1	$O(W \log D)$
PDB-DSSE	✓	Type-I ⁻	1	$O(W \log D)$

TABLE 3: Comparison of search and update time complexity.

Scheme	Search	Update
MONETA [7]	$\widehat{O}(u_w \log N + \log^3 N) \cdot t_{ED}$	$\widehat{O}(\log^2 N) \cdot t_{ED}$
ORION [9]	$O(f_w \log^2 N) \cdot t_{ED}$	$O(\log^2 N) \cdot t_{ED}$
FB-DSSE [13]	$O(u_w) \cdot t_{AM}$	$O(1) \cdot t_{AM}$
PDB-DSSE	$O(u_w) \cdot t_{AM}$	$O(f_w + t_{AM})$

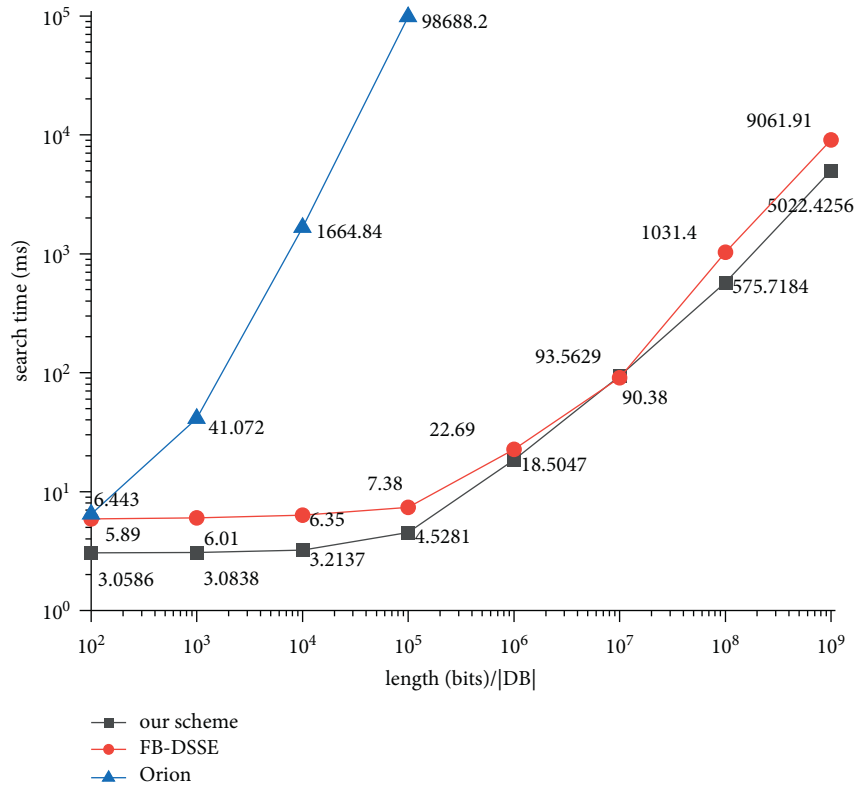


FIGURE 3: The comparison of search time.

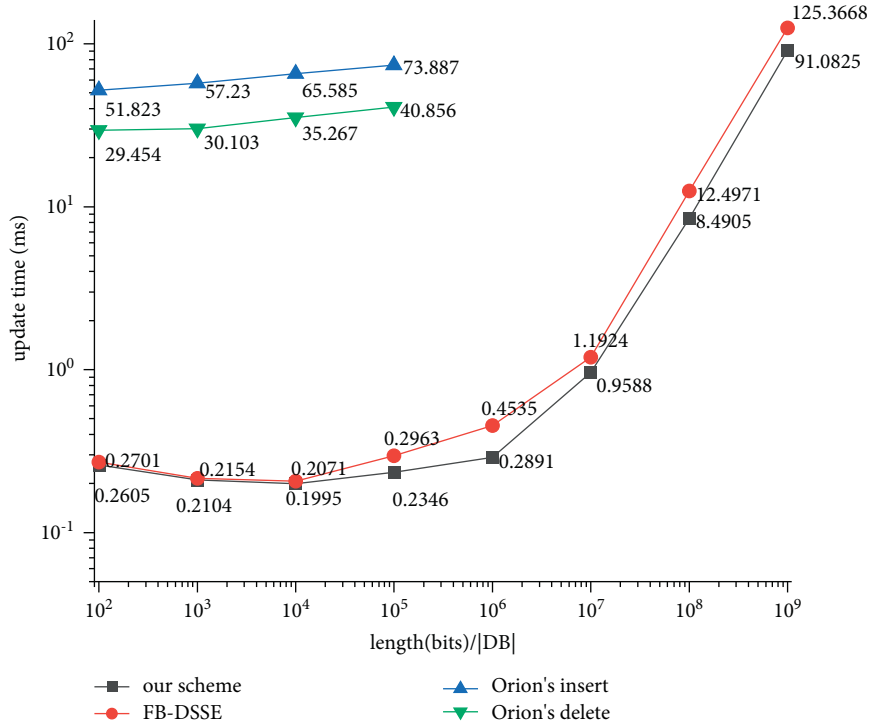


FIGURE 4: The comparison of update time.

DSSE scheme since it needs to update the PDB, but the update efficiency can still be maintained. Through comparison, it can be found that PDB-DSSE has high update efficiency.

7. Conclusion

In this paper, we construct a dynamic padding method that can be used for DSSE schemes and give an application scenario to realize a DSSE scheme PDB-DSSE with forward privacy and Type-I⁻ backward privacy. First, we introduce outlier detection technology to design a dynamic database padding algorithm that can be used in DSSE schemes. We use the padded bogus file to confuse the real file to prevent leakage of information about the files that currently matches the search keyword. Furthermore, we design a new index structure based on the bitmap index that is suitable for the padding database that simplifies the process of modifying the index when updating. Finally, we propose an application scenario of the padding method and implement the $\mathcal{L}_{\text{PDB-DSSE}}$ -adaptive forward and Type-I⁻ backward privacy DSSE scheme. Simulation results and comparative experiments show that the dynamic padding scheme proposed in this paper is suitable for DSSE schemes. In addition, the PDB-DSSE scheme, which incorporates the dynamic padding scheme proposed in this paper, still maintains efficient search and update efficiency.

As the size of the database increases, bitmap indexing and padding algorithms face certain limitations. To support a larger number of files (such as billions), in future work, we will consider dividing the index into blocks, padding each

block separately to improve retrieval and update efficiency and reduce computational overhead. In addition, we will consider the use of bitmap indexes to implement a search for conjunctive keyword queries, thereby improving the retrieval efficiency of existing solutions.

Data Availability

Previously reported Enron Email Datasets were used to support this study and are available at <https://www.cs.cmu.edu/~.enron/>.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] DX Song, D Wagner, and A Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pp. 44–55, IEEE, Berkeley, CA, USA, May 2000.
- [2] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.
- [3] D Cash, P Grubbs, J Perry, and T Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 668–679, ACM, Denver, CO, USA, October 2015.
- [4] Y Zhang, J Katz, and C Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proceedings of the 25th {USENIX} Security*

- Symposium (USENIX Security 16)*, pp. 707–720, USENIX Association, Austin, TX, USA, August 2016.
- [5] E Stefanov, C Papamanthou, and E Shi, “Practical dynamic searchable encryption with small leakage,” *NDSS*, vol. 71, pp. 72–75, 2014.
- [6] R. Bost, “Σοφ οζ: forward secure searchable encryption,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1143–1154, ACM, Vienna Austria, October 2016.
- [7] R Bost, B Minaud, and O Ohrimenko, “Forward and backward private searchable encryption from constrained cryptographic primitives,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1465–1482, ACM, Dallas, TX, USA, October 2017.
- [8] SF Sun, X Yuan, JK Liu et al., “Practical backward-secure searchable encryption from symmetric puncturable encryption,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 763–780, ACM, Toronto, Canada, October 2018.
- [9] J Ghareh Chamani, D Papadopoulos, C Papamanthou, and R Jalili, “New constructions for forward and backward private symmetric searchable encryption,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1038–1055, ACM, Toronto, Canada, October 2018.
- [10] P Rizomiliotis and S Gritzalis, “Simple forward and backward private searchable symmetric encryption schemes with constant number of roundtrips,” in *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pp. 141–152, ACM, London, UK, November 2019.
- [11] A. Najafi, H. H. S. Javadi, and M. Bayat, “Verifiable ranked search over encrypted data with forward and backward privacy,” *Future Generation Computer Systems*, vol. 101, pp. 410–419, 2019.
- [12] L Sardar and S Ruj, “Fspvdsse: A forward secure publicly verifiable dynamic sse scheme,” in *Proceedings of the International Conference on Provable Security*, pp. 355–371, Springer, Cairns, Australia, October, 2019.
- [13] C Zuo, SF Sun, JK Liu, J Shao, and J Pieprzyk, “Dynamic searchable symmetric encryption with forward and stronger backward privacy,” in *Proceedings of the European Symposium on Research in Computer Security*, pp. 283–303, Springer, Luxembourg, UK, September 2019.
- [14] C. Zuo, S. Sun, J. K. Liu, J. Shao, J. Pieprzyk, and L. Xu, “Forward and backward private dsse for range queries,” *IEEE Transactions on Dependable and Secure Computing*, vol. 99, p. 1, 2020.
- [15] X Li, T Xiang, and P Wang, “Achieving forward unforgeability in keyword-field-free conjunctive search,” *Journal of Network and Computer Applications*, vol. 166, Article ID 102755, 2020.
- [16] I Demertzis, JG Chamani, D Papadopoulos, and C Papamanthou, “Dynamic searchable encryption with small client storage,” in *Proceedings of the Network and Distributed System Security Symposium*, DBLP, San Diego, CA, USA, February 2020.
- [17] S. Chatterjee, S. K. Parshuram Puria, and A. Shah, “Efficient backward private searchable encryption,” *Journal of Computer Security*, vol. 28, no. 2, pp. 229–267, 2020.
- [18] R Bost and PA Fouque, “Thwarting leakage abuse attacks against searchable encryption—a formal approach and applications to database padding,” *IACR Opens the Cryptology ePrint Archive*, vol. 2017, p. 1060, 2017.
- [19] L Xu, X Yuan, C Wang, Q Wang, and C Xu, “Hardening database padding for searchable encryption,” in *Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 2503–2511, IEEE, Paris, France, April 2019.
- [20] C Castelluccia, E Mykletun, and G Tsudik, “Efficient aggregation of encrypted data in wireless sensor networks,” in *Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pp. 109–117, IEEE, San Diego, CA, USA, July 2005.

Research Article

Gene Sequence Clustering Based on the Profile Hidden Markov Model with Differential Identifiability

Xujie Ren , Tao Shang , Yatong Jiang , and Jianwei Liu 

School of Cyber Science and Technology, Beihang University, Beijing 100083, China

Correspondence should be addressed to Tao Shang; shangtao@buaa.edu.cn

Received 17 September 2021; Revised 29 October 2021; Accepted 29 November 2021; Published 24 December 2021

Academic Editor: Ding Wang

Copyright © 2021 Xujie Ren et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the era of big data, next-generation sequencing produces a large amount of genomic data. With these genetic sequence data, research in biology fields will be further advanced. However, the growth of data scale often leads to privacy issues. Even if the data is not open, it is still possible for an attacker to steal private information by a member inference attack. In this paper, we proposed a private profile hidden Markov model (PHMM) with differential identifiability for gene sequence clustering. By adding random noise into the model, the probability of identifying individuals in the database is limited. The gene sequences could be unsupervised clustered without labels according to the output scores of private PHMM. The variation of the divergence distance in the experimental results shows that the addition of noise makes the profile hidden Markov model distort to a certain extent, and the maximum divergence distance can reach 15.47 when the amount of data is small. Also, the cosine similarity comparison of the clustering model before and after adding noise shows that as the privacy parameters changes, the clustering model distorts at a low or high level, which makes it defend the member inference attack.

1. Introduction

In recent years, with the development of IoT-based gene sequencing technology, the amount of biological gene sequence data has increased rapidly [1]. The biological sequence data contains information about species evolution, genetic traits, and potential diseases in genes. With the help of biological big data and modern computing methods [2], such as machine learning technology, researchers' studies in genomics, transcriptomics, and proteomics can be further developed, providing help for disease diagnosis and treatment, drug research and development, reproductive health, and other fields.

Clustering is an unsupervised machine learning technique. By dividing the data with high similarity into one cluster and the data with low similarity into different clusters, the unlabeled data can be automatically categorized. The cluster analysis technology can be applied to the genomic data to realize the analysis of gene homology, genetic diseases, and traceability.

The concern about privacy has come at the time of the surge in data. The personal identifiable private information contained in the gene sequences is easy to be used by

attackers, and the individual unique information extracted from the gene fragments will lead to the disclosure of private information [3]. Therefore, it is necessary to develop privacy protection technology for genomic data to ensure that private information will not be stolen. At present, the most popular approach adopted by some service providers is not to provide genomic data in public and open the query authority of the black-box model only to the users. However, Shokri et al. [4] proposed the construction method of a shadow model. By building a shadow data set and a shadow model, the training set of the black-box model was deduced. This attack is generic and not specific to a particular model.

At the same time, Shokri thought that differential privacy [5] (DP) is an effective defense against this kind of attack. Differentially private models are secure against membership inference attacks. By adding Laplace noise to the model, the change of a single individual in the database will not have a significant impact on the output result, which increases the difficulty of the attacker to carry out the inference attacks. Differential privacy is a definition of privacy based on strict mathematical proof. However, it defines privacy as the difference between the output of two neighbor databases,

which is inconsistent with the relevant privacy regulations, such as the U.S. HIPAA safe harbor rule [6]. To solve this problem, Lee and Clifton [7] put forward the concept of differential identifiability (DI), which defined privacy as the probability of an individual being identified by an attacker in the database, which is more consistent with people’s cognition of privacy. In addition, the privacy parameters defined by differential identifiability are easier to understand by practitioners who are not major in privacy protection.

In this paper, a privacy clustering algorithm based on the profile hidden Markov model (PHMM) is proposed. In theory, some algorithms based on dynamic programming, such as Needleman–Wunsch algorithm, can also solve the exact solutions of a multiple sequence alignment. However, as the number of sequences increases, the complexity of the algorithm increases exponentially. The alignment algorithm based on the hidden Markov model can train a probability matrix for long sequences and is more suitable for the data mining analysis of large data sets. The sequences of high similarity are divided into the same cluster. To protect the private data, we add random Laplace noise based on differential identifiability into PHMM. We only use DNA sequences as the example of algorithm illustration and assume that the occurrence probability of each nucleotide is equal. The clustering process is iterative. We show the iterative allocation method of differential identifiability privacy parameters that can be used to add noise to PHMM. The experimental results show that if the privacy parameters are properly set, the proposed model is still usable after adding noise.

2. Related Work

Clustering analysis using HMM was proposed as early as 1985 by Juang and Rabiner [8], who put forward the concept of divergence distance that could effectively measure the similarity between the two HMMs and applied it to the sequence clustering analysis. Krogh et al. [9] proposed a variant of the HMM called the profile hidden Markov model that defined the hidden states as matching states, insertion states, and deletion states for multiple sequence alignment of genes and proteins. At present, such sequencing platforms as PFAM [10], HMMER [11], and SMART [12] use the hidden Markov model for multisequence alignment. Kater et al. [13] proposed a clustering robustness score that solved the problem that most clustering methods were not robust to noise. By artificially adding noise, they obtained the clusters of biologically significant cells in a single cell expression dataset. Jia et al. [14] proposed a double elite genetic clustering algorithm after an in-depth analysis of traditional clustering algorithms. This algorithm guarantees the global convergence of the population by elite strategies.

The model-based approach constructs the clustering model by the integration of a finite number of mixed submodels. Each submodel represents a class of data and is built by calculating the statistical properties of these data. Clustering based on the hidden Markov model was first proposed by Juang and Rabiner [8] and applied to protein sequence clustering by Krogh [9]. For the problem of model

initialization and cluster number selection of HMM clustering, Smyth [15] provided a reference scheme. In model-based clustering, the logarithmic likelihood of the sequence and the model can be considered as the basis to measure the sequence similarity. Clustering based on HMM can be formally expressed as

$$f_k(O) = \sum_{j=1}^k f_j(O|\lambda_j), \quad (1)$$

where the parameter λ_j is obtained by the maximum likelihood estimation method in Section 4. Given the sequence O and the j^{th} model parameter λ_j , $f_j(O|\lambda_j)$ is the probability density of the sequence under this model. Logarithmic probability is usually used to represent the probability of the output sequence of the model to facilitate calculation and prevent underflow. The commonly used methods of calculating the sequence output probability include the forward-backward algorithm and the Viterbi algorithm [16]. Logarithmic probability can be used to describe the degree of fit between the sequence and model, i.e., the degree of homology between the sequence and gene family, and it can be used as the basis of clustering.

The concept of privacy protection can be traced back to the 1970s [17]. Traditional privacy protection methods can be divided into anonymization [18–20] and data perturbation [5, 7]. Anonymization technology is vulnerable to consistency attacks and background knowledge attacks [21], and it lacks strict mathematical proof. Differential privacy [5] is a privacy definition with a strict mathematical basis that protects privacy by limiting the influence of individuals on the output of the database. Differential privacy mechanisms mainly include the Laplace mechanism and exponential mechanism [22], and the noise size is determined by the privacy budget ϵ . In 2012, Lee and Clifton [7] argued that differential privacy did not pay attention to the risk of individuals being identified by their adversaries in the database, which was inconsistent with the definition of relevant laws and regulations. In addition, the parameter setting of differential privacy was a tricky issue, thus putting forward differential identifiability. Compared with differential privacy, the privacy parameter setting of differential differentiability is more intuitive and easier to be understood by relevant practitioners. Shang et al. [23] proposed the composition theorem of differential identifiability and applied it to the k -means clustering in the MapReduce framework.

3. Preliminary

3.1. Profile Hidden Markov Model. The hidden Markov model [24] is a linear serial statistical model composed of multiple observable states and corresponding hidden state nodes. Its hidden states are the unobservable underlying sequences. The transition between the hidden states is carried out according to a certain probability, and each observable state also appears according to a certain probability. In other words, HMM is a set of finite states that are transferred by a series of hidden states, which can be described indirectly by an observable sequence.

The profile hidden Markov model (PHMM) was first proposed by Krogh et al. [9] and applied to a multisequence alignment of proteins. PHMM defines the hidden state as match state, insert state, and delete state on the basis of HMM and adds a start state and an end state. Compared with the traditional HMM, PHMM is more sensitive to the specific position of the state in the observation sequence. Figure 1 shows a simple PHMM with squares representing the match state, diamonds representing the insert state, and circles representing the delete state.

Given an input gene sequence to PHMM, PHMM will match the symbols in the sequence and the insertion and deletion states and output a score. The higher the score, the more the sequence matches with the sequence family of training PHMM and the higher the gene sequence homology from the biological point of view.

The hidden Markov model has three basic problems: evaluation problem, learning problem, and decoding problem.

The decoding problem is the focus of this paper. Given that model $\lambda = (\pi, A, B)$ and observation sequence O . π , A , and B represent the initial state distribution, state transition probabilities, and observation symbol probabilities, respectively. The model outputs the most likely corresponding hidden state path and its probability value of O . The Viterbi algorithm [16] is a dynamic programming algorithm. Let $V_j^M(i)$ be the log odds of matching the sequence x_1, \dots, x_i to the optimal path of the submodel ending with state j . Similarly, $V_j^I(i)$ and $V_j^D(i)$ represent the log odds of the optimal path ending with the states I_j and D_j , respectively. The general formula of the Viterbi algorithm is as follows:

$$\begin{aligned}
 V_j^M(i) &= \log \frac{e_{M_j}(x_i)}{q_{x_i}} \\
 &+ \max \begin{cases} V_{j-1}^M(i-1) + \log a_{M_{j-1}M_j}, \\ V_{j-1}^I(i-1) + \log a_{I_{j-1}M_j}, \\ V_{j-1}^D(i-1) + \log a_{D_{j-1}M_j}, \end{cases} \\
 V_j^I(i) &= \log \frac{e_{I_j}(x_i)}{q_{x_i}} \\
 &+ \max \begin{cases} V_j^M(i-1) + \log a_{M_jI_j}, \\ V_j^I(i-1) + \log a_{I_jI_j}, \\ V_j^D(i-1) + \log a_{D_jI_j}, \end{cases} \\
 V_j^D(i) &= \max \begin{cases} V_{j-1}^M(i) + \log a_{M_{j-1}D_j}, \\ V_{j-1}^I(i) + \log a_{I_{j-1}D_j}, \\ V_{j-1}^D(i) + \log a_{D_{j-1}D_j}. \end{cases}
 \end{aligned} \tag{2}$$

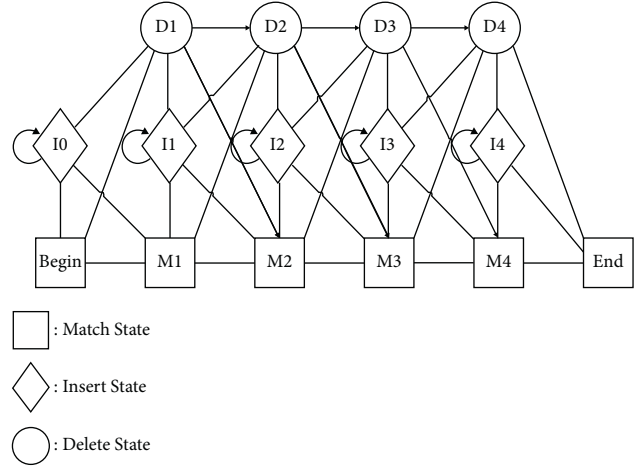


FIGURE 1: A simple PHMM structure.

The maximum value of the three is the log odds of the sequence corresponding to the optimal path, and the optimal path can be found after recalling.

3.2. Differential Identifiability. Differential identifiability aims to solve the problem that ϵ -differential privacy [5] has no clear guidelines on parameter setting. Strictly speaking, differential identifiability is another form of the implementation of differential privacy, which can provide the same privacy protection capability as differential privacy. It takes the probability ρ of individuals being identified in the database as the privacy parameter, which can be easily set by decision makers.

Definition 1. (ρ -differential identifiability).

Given a query function f , for any data set D to be queried, $\forall D' = D - t^*$ and any individual $t \in U - D'$, if

$$\Pr[I(t) \in \mathcal{F}_D \mid M_f(D) = R, D'] \leq \rho, \tag{3}$$

then the mechanism M satisfies ρ -differential identifiability. U represents the set composed of all possible individuals, $I(t)$ represents the identifier of individual t in the universal set U , and \mathcal{F}_D represents the set of all individuals in database D , i.e., $\mathcal{F}_D = \{I(t) \mid t \in D\}$.

Definition 1 states that if an adversary possesses a neighbor data set and has the ability to query the database, the mechanism limits the probability of an attacker identifying an individual in the data set to less than or equal after he obtains the output of the database. Differential identifiability is similar to differential privacy in that they assume equal capabilities of the adversary. The difference between them is that differential privacy defines privacy as the indistinguishable output of two neighbor databases, while differential identifiability focuses on the probability of an individual being identified by an adversary in the database. In contrast, the privacy parameters of differential identifiability are more intuitive and convenient for nonprofessionals to set.

Differential identifiability can also realize privacy protection by adding random noise to the query result, i.e., $R = f(D) + Y$, Y is a random variable that obeys a certain Laplace distribution. Lee and Clifton gave a Laplace mechanism satisfying differential identifiability, i.e., $Y \sim \text{Lap}(\lambda)$. Similar to differential privacy, the model is distorted after adding noise, which interferes with the attacker's inference attack and reduces the probability of a successful attack as is described in Theorem 1.

Theorem 1. *Suppose there is a random mechanism M , and for any adversary, if $\lambda = \Delta f / \ln(m-1)\rho/1 - \rho$, the mechanism M satisfies ρ -differential identifiability, where $m = |\Psi| = |U| - |D'|$ and Δf is the sensitivity of the query function f .*

For data publishing in some complex situations, privacy protection mechanisms may need to be applied multiple times. On the basis of Lee and Clifton's work, Shang et al. [23] studied the composition theorem of differential identifiability. Based on the composition theorem, the differential identifiability privacy protection with mathematical proof can be provided when the adversary has the ability of multiple combinatorial queries.

Theorem 2 (Sequential Composition [23]). *Given a set of n mechanisms M_1, \dots, M_n , each $M_i (i \in [1, n])$ provides ρ_i -differential identifiability. For all databases D , the sequential composition $M(D) = (M_1(D), \dots, M_n(D))$ provides $(m^{n-1} \sum_{i=1}^n \rho_i)$ -differential identifiability.*

Theorem 3 (Parallel Composition [23]). *Given a set of n mechanisms M_1, \dots, M_n , each $M_i (i \in [1, n])$ provides ρ_i -differential identifiability. D_i is the arbitrary disjoint subset of the input database D . The sequence of mechanisms $M_i(D_i)$ provides $(\min \rho_i)$ -differential identifiability.*

In some iterative data mining mechanisms, such as unsupervised clustering, the realization of privacy protection may require noise-adding with multiple rounds or disjoint subdata sets. Sequential composition and parallel composition, respectively, indicate that when the same data set is noised in multiple rounds and the disjoint data set is noised in separate rounds, the model obtained still meets differential identifiability.

4. Profile Hidden Markov Model with Differential Identifiability

4.1. Modeling. Adding noise to the black-box model is an effective privacy protection method to resist an inference attack [30]. The model of differential privacy makes the probability of a dataset producing an output close to its neighbor dataset [4]. Differential identifiability limits the probability that an attacker can identify a particular record in the database by perturbing the model randomly. Algorithm 1 shows the steps of constructing the profile hidden

Markov model with DI. In a clustering algorithm for gene sequence in Section 5, the DI-PHMM constructed by Algorithm 1 will be used to measure the similarity between the sequences.

In PHMM, the states are defined as match, insert, and delete. We adopt the maximum likelihood estimation to estimate the parameters (transition probability and emission probability). A_{kl} represents the number of transitions from the state k to state l in the training data, and $E_k(a)$ represents the number of the emission symbol a of state k . a_{kl} represents the probability of transition from state k to state l , and $e_k(a)$ represents the probability of transmitting observation symbol a under state k . Pseudocount (for simplicity, usually set to 1) is used to add to the count result of each state or observed symbol. It is done to prevent overfitting and avoid the problem of calculating the fractions wrong when a symbol count is 0.

4.2. Privacy Parameter Allocation. The traditional differential privacy follows the mechanism that the privacy budget in each round is half of that of the last round in the iterative process. In previous work [23], according to the mapping relationship between differential identifiability and differential privacy, the privacy parameters of differential identifiability were also set to decrease by half. In essence, this method is still differential private noise and does not satisfy the differential identifiability's composition theorem.

Different from the composition theorem of differential privacy, for the sequence composition acting on the same data set, the parameter ρ of the final model is the product of the privacy parameters of each round, namely $m^{n-1} \sum_{i=1}^n \rho_i$, rather than the summation in differential privacy [26]. Set the expected privacy parameter ρ in advance for a finite number of iterations. For each round, the differential identifiability privacy parameter can be set to

$$\rho_i = \left(\frac{\rho}{m^{N-1}} \right)^{1/N}, \quad (4)$$

where ρ_i is the privacy parameter of iteration in round i , and N is the total time of iteration rounds. According to the sequential composition of differential identifiability, the final model meets $\rho_{\text{final}} = m^{N-1} \sum_{i=1}^N \rho_i = \rho$.

When the number of iteration rounds is infinite or unknown, according to sequence combination and power series convergence, it can be deduced that the privacy parameter of each round should be set as

$$\rho_i = \frac{(\rho m)^{(1/2)^i}}{m}. \quad (5)$$

4.3. Noise Addition. Counting data is generally considered a risk of being subjected to differential cryptanalysis. Accordingly, when the maximum likelihood estimation of observation symbols is carried out here, random noise Y_k is added to the technical results of each observation symbol, and k represents the corresponding hidden state. Y_k obeys the Laplace distribution with a position parameter of 0, i.e.,

Input: Training sequence data $O = (O_1, O_2, \dots, O_n)$ and differential identifiability parameter ρ
Output: A PHMM Calculate probability of transition from state k to state l . $a_{kl} = \text{pseudocount}(A_{kl}/\sum_{l'} A_{kl}')$;
(1) //Generate privacy noise for matching state and insert state emission probabilities.
(2) for a in ('A', 'G', 'T', 'C')
(3) $Y_{ma} \sim \text{Lap}(\Delta f/\ln(m-1)\rho/1-\rho)$,
(4) $Y_{ia} \sim \text{Lap}(\Delta f/\ln(m-1)\rho/1-\rho)$;
(5) //Calculate the emission probability of state k to state a .
(6) for k in ('match', 'insert')
(7) $e_k(a) = \text{pseudocount}(E_k(a) + Y_k/\sum_{a'} E_k(a') + \sum_{k'} Y_{k'})$
(8) //Validity test. Check whether the emission probability is valid. If not, go
(9) back to step 2.
(10) if $e_m(a), e_i(a) \notin [0, 1]$
(11) Regenerate privacy noise
(12) Return $a_{kl}, e_k(a)$

ALGORITHM 1: Profile hidden Markov model with differential identifiability (DI-PHMM).

$Y_k \sim \text{Lap}(\Delta f/\ln(m-1)\rho/1-\rho)$ (position parameter omitted). The sensitivity is usually determined by the query function. When a single individual in the database is modified, added, or deleted, the maximum value of its count changes by 1, and hence, the sensitivity $\Delta f = 1$ here. Differential identifiability defines the set U of all possible data individuals in the full set. For gene sequences, there is $|U| = 5^L$ (the space size of the bases "A, C, G, T," and placeholder "-"). Hence, $m = |U| - |D'| = 5^L - n + 1$.

The validity test of the algorithm aims to prevent the emission probability value less than 0 because of the influence of noise. When it happens, the algorithm regenerates the noise and adds noise until the new probability value is between 0 and 1.

5. Gene Sequence Clustering Algorithm Based on DI-PHMM

5.1. Clustering Algorithm. Assume there are two PHMMs λ_1 and λ_2 and one DNA sequence. Use the Viterbi algorithm on them and output the two scores s_1 and s_2 . If $s_1 > s_2$, we can say that the sequence is more homologous with the sequences that constructed λ_1 . Clustering for the gene sequences is an algorithm that can automatically divide the highly homologous sequences into one class. The homology analysis of the sequences is helpful for the researchers to trace the origin of genes and analyze the point of genetic diseases (such as comparing mouse genetic diseases with human genes). In Algorithm 2, a sequence clustering method based on the hidden Markov model is proposed, and the clustering model is protected by differential identifiability.

The clustering model needs to train for several rounds of iteration, and finally, it converges. The k -means is one of the most widely used clustering algorithms. The key of k -means is the measurement of the distance between the data and the cluster centers, and the data that is the closest to one center is grouped into one group. k cluster centers will update in each round of iteration, as well as data clustered according to distance metric. In general, clustering can be divided into distance-based clustering and model-based clustering

according to the measurement method [27]. The classical k -means uses the Euclidean distance $d_{ij}(x_i, y_j) = \sqrt{(x_i - y_j)^T (x_i - y_j)}$ to measure the similarity between data. The smaller the distance, the greater the similarity. Hence, such data will be grouped together.

Similar to the classical clustering algorithm, the initial central submodel is selected randomly. Each sequence outputs a score for each PHMM and is divided into the corresponding cluster according to the highest score. The central submodel was updated after each round of iteration. According to whether the number of iteration rounds is preset, calculate the size of the privacy parameter of each round. Use the algorithm DI-PHMM, and the sequences within each cluster is used as input. The output model is used as the new central submodel of this cluster. Finally, the distance between the center model of this round and the previous round is calculated according to the divergence distance. If the distance is small enough, the clustering is considered to be converged. Otherwise, the iteration continues until the number of iteration rounds reaches a preset threshold or the central submodel no longer changes. Table 1 compares some gene sequence clustering models' performance.

5.2. Divergence Distance of DI-PHMM. The termination condition of clustering iteration is that the division of the clusters will not change, which is also reflected in the fact that the output score of each sequence for each submodel will not change. Juang and Rabiner [8] studied the divergence distance of the hidden Markov model, which reflects the similarity between the two HMMs. The definition of divergence distance is as follows:

$$D(\lambda_1, \lambda_2) = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{i=1}^T \log \frac{f(O_i | \lambda_1)}{f(O_i | \lambda_2)}, \quad (6)$$

λ_1 and λ_2 represent the two hidden Markov models, respectively. The more similar the two models are, the smaller the divergence distance will be.

The above steps are iterated until the central submodel of the iteration no longer changes or the number of iteration rounds reaches a threshold.

```

Input: Number of clusters  $K$ , training sequence data  $O = \{O_1, O_2, \dots, O_n\}$ , DI parameter  $\rho$  and round number of iteration  $N$ 
(optional)
Output: Index of the cluster to which the sequence belongs  $C = \{C_1, C_2, \dots, C_n\}$  where  $C_n = 1, 2, \dots, K$ 
(1)  $r \leftarrow +$ 
(2) for  $i$  in  $n$ 
(3) for  $j$  in  $k$ 
(4) //Calculate the score of the sequence for each PHMM
(5)  $d_j^i = \text{viterbi}(O_i, \lambda_j^r)$ 
(6) //Divide the sequence into the corresponding cluster according to the highest score
(7)  $C_i = \arg \max_j (d_j^i)$ 
(8) for  $k$  in  $K$ 
(9) if  $(N! = \text{NULL})$ :
(10)  $\rho_r = (\rho/m^{N-1})^{1/N}$ 
(11) else
(12)  $\rho_r = (\rho m)^{1/2^r} / m$  //The privacy parameter is assigned according to whether the number of iteration rounds is fixed.
(13) //Construct a new cluster center sub-model
(14)  $\lambda_k^r = \text{DI-PHMM}(O_k, L, \rho_r)$ 
(15) //The degree of change of the model from the last iteration (divergence distance)
(16)  $D_k = D(\lambda_k^r, \lambda_k^{r-1})$ 

```

ALGORITHM 2: Gene sequence clustering algorithm based on DI-PHMM (DI-GSCA).

TABLE 1: Gene sequence clustering models' performance.

Model	Availability ¹ (%)	Efficiency	Security
DKHC [28]	98.26	Slow	×
DEGCA [14]	96.7	Slow	×
CD-HIT [29]	95	Fast	×
GeneRage [30]	95	Fast	×
DI-GSCA	CS ² : 0.77	Slow	✓

¹The availability data were the highest value in their respective references.

²CS: cosine similarity.

6. Experimental Results

In this section, we will demonstrate the influence of privacy protection on clustering performance by comparing it with the output results of the unnoised model. The dataset is *Mus musculus* Immunoglobulin Lambda Chain Complex (IGL) on chromosome 16 and *Homo sapiens* chromosome 1 genomic from the National Center for Biotechnology Information. A total of 10,000 sequences with a length of 70 bp were intercepted from them. The experimental results were the average values of 10 experiments.

6.1. Distortion of DI-PHMM. Based on the description of the algorithm of DI-PHMM, the divergence distance between the native and noised PHMM models is given to show the effect of noising on the PHMM model. The privacy parameter of differential identifiability is set to nine values of 0.1 to 0.9. Generally speaking, a smaller privacy parameter indicates that the adversary has a lower probability of identifying the original data set. The noise that needs to be added is higher, and the distortion of the model is higher. In this section, the training data sets are randomly divided into 10,000, 5000, 2000, and 1000 sequences to build the PHMM model, respectively. Generally, the larger the data set, the more robust it is to noise. Therefore, when the data set size is

large, the influence of noise processing on the PHMM model is small. The experimental results are shown in Figure 2.

Because of the randomness of the adding noise, the divergence distance of the PHMM model after adding the noise may fluctuate to a certain extent. As a whole, PHMM constructed with the size of 10,000 sequences has the smallest divergence distance after adding the noise, ranging from 0.01 to 8.56. PHMM constructed from a dataset of only 1000 sequences showed a large degree of distortion after noising. When the privacy parameter was 0.1, the divergence distance reached a maximum of 15.47.

Obviously, with the increase of the privacy parameter, the difference between the noise-added model and the normal model decreases, indicating that the noise is indeed reduced. At the same time, larger data sets show stronger robustness to noise. It seems that the fluctuation range of the divergence distance of $n = 5000$ is higher than that of $n = 2000$. It is because of the randomness of noise.

6.2. Performance of Clustering Model. The output result of the clustering algorithm described in the gene sequence clustering algorithm based on DI-PHMM is a vector of the class corresponding to the sequence data set. Cosine similarity is adopted in this section to measure the similarity between the clustering results. The cosine similarity is evaluated by calculating the cosine of the angle between the two vectors. Given two vectors A and B , the cosine similarity between them can be expressed as $\text{similarity} = A \cdot B / \|A\| \|B\| = \sum_{i=1}^n A_i \times B_i / \sqrt{\sum_{i=1}^n (A_i)^2} \sqrt{\sum_{i=1}^n (B_i)^2}$. The value of cosine similarity ranges from 0 to 1. When the cosine similarity is 0, it means that the two vectors are perpendicular and the similarity is 0. When the cosine similarity is 1, the two vectors are equal.

By comparing the cosine similarity between the clustering results constructed by the PHMM model without the noise and privacy clustering results, the influence of adding

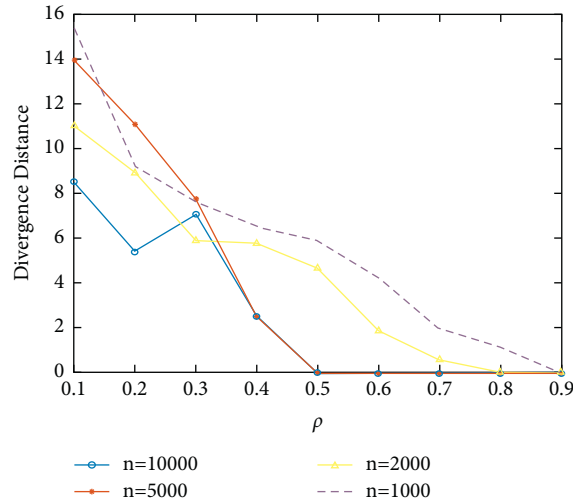


FIGURE 2: Divergence distance of PHMM before and after noise addition.

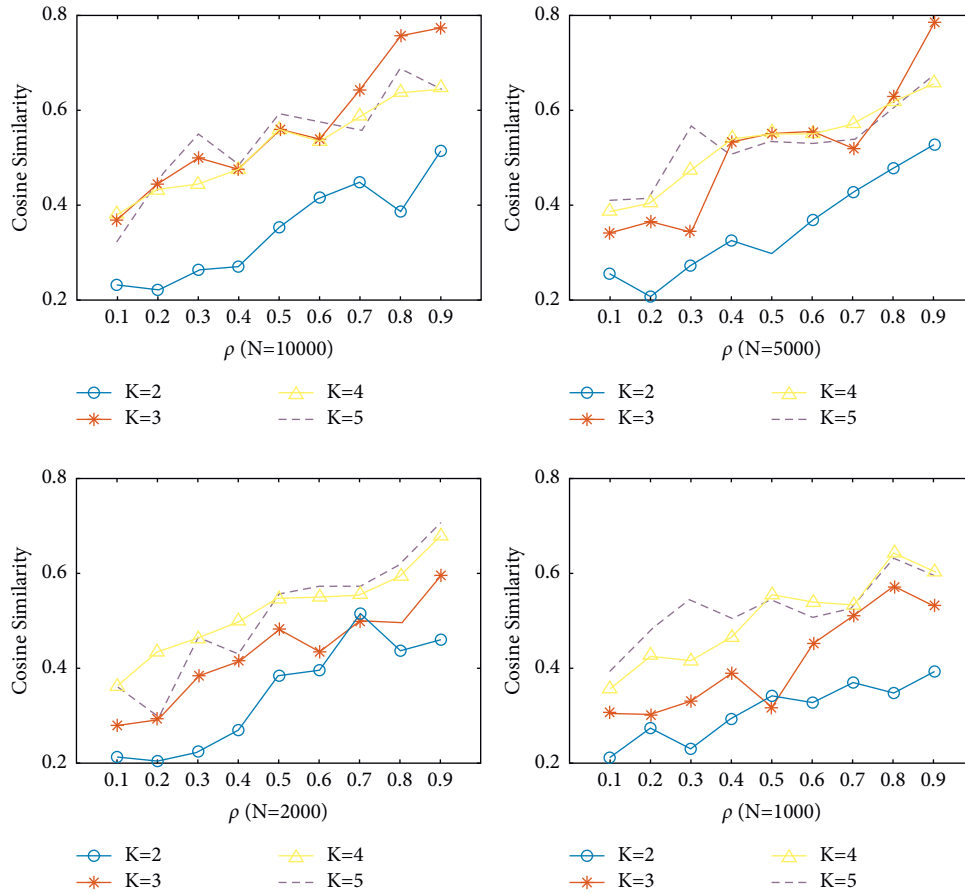


FIGURE 3: Cosine similarity of clustering model.

differential identifiability noise on the clustering performance is demonstrated. In the experiment, the number of clusters was set as $K = 1, 2, 3, 4$, and the experimental results are shown in Figure 3.

The experimental results show that with an increase in the privacy parameters, the privacy noise decreases

gradually, the degree of privacy protection decreases, and the clustering performance improves accordingly. When we set the appropriate number of clustering and privacy parameters, the clustering results after adding the noise can still maintain an acceptable usability. For example, when $N = 10,000$, the number of clustering is 3, and the privacy

parameter is 0.9, the clustering performance reaches the highest, and the cosine similarity is 0.77 compared with the normal model.

7. Conclusion

On the basis of the classical gene sequence clustering algorithm, we added the privacy noise, satisfying the differential identifiability into the clustering model so that the model can resist the member inference attack and also provide a valuable privacy protection method for researchers related to biological information. In this paper, an iterative allocation scheme of differential identifiability privacy parameters was presented to realize privacy protection in complex models. The experimental results show that the clustering utility of the model with differential identifiability noise is reduced, however, the model availability can still be maintained at a high level after adjusting the privacy parameters.

Future research can focus on the basic theory of differential identifiability, how to balance privacy and model performance, and applications in more scenarios.

Data Availability

The gene sequence data used to support the findings of this study is cited from the NCBI repository (<https://www.ncbi.nlm.nih.gov/>).

Conflicts of Interest

The authors declare that no conflicts of interest exist.

Acknowledgments

This project was supported by the National Key Research and Development Program of China (No. 2016YFC1000307) and the National Natural Science Foundation of China (Nos. 61971021 and 61571024).

References

- [1] Z. Alansari, N. B. Anuar, A. Kamsin, S. Soomro, and M. R. Belgaum, *Progress in Advanced Computing and Intelligent Engineering*, Springer, Singapore, 2019.
- [2] P. Mamoshina, A. Vieira, E. Putin, and A. Zhavoronkov, "Applications of deep learning in biomedicine," *Molecular Pharmaceutics*, vol. 13, no. 5, pp. 1445–1454, 2016.
- [3] Y. Erlich and A. Narayanan, "Routes for breaching and protecting genetic privacy," *Nature Reviews Genetics*, vol. 15, no. 6, pp. 409–421, 2014.
- [4] R. Shokri, M. Stronati, and C. Song, "Membership inference attacks against machine learning models," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, IEEE, San Jose, CA, USA, 2017.
- [5] C. Dwork, *Differential Privacy: A Survey of Results. International Conference on Theory and Applications of Models of Computation*, Springer, Berlin, Heidelberg, Getmany, 2008.
- [6] Office for Civil Rights H H S, "Standards for privacy of individually identifiable health information," *Final rule, Federal Register*, vol. 67, no. 157, pp. 53181–53273, 2002.
- [7] J. Lee and C. Clifton, "Differential identifiability," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1041–1049, ACM, New York, NY, USA, August 2012.
- [8] B.-H. Juang and L. R. Rabiner, "A probabilistic distance measure for hidden Markov models," *AT&T Technical Journal*, vol. 64, no. 2, pp. 391–408, 1985.
- [9] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler, "Hidden Markov models in computational biology," *Journal of Molecular Biology*, vol. 235, no. 5, pp. 1501–1531, 1994.
- [10] R. D. Finn, A. Bateman, J. Clements et al., "Pfam: the protein families database," *Nucleic Acids Research*, vol. 42, no. D1, pp. D222–D230, 2014.
- [11] R. D. Finn, J. Clements, W. Arndt et al., "HMMER web server: 2015 update," *Nucleic Acids Research*, vol. 43, no. W1, pp. W30–W38, 2015.
- [12] I. Letunic, T. Doerks, and P. Bork, "SMART: recent updates, new developments and status in 2015," *Nucleic Acids Research*, vol. 43, no. D1, pp. D257–D260, 2015.
- [13] I. Kanter, P. Dalerba, and T. Kalisky, "A cluster robustness score for identifying cell subpopulations in single cell gene expression datasets from heterogeneous tissues and tumors," *Bioinformatics*, vol. 35, no. 6, pp. 962–971, 2019.
- [14] R. Y. Jia, F. B. Song, and S. W. Tang, "Genetic clustering algorithm with double elite genetic strategy," *Journal of Chinese Computer Systems*, vol. 41, no. 7, pp. 1375–1380, 2020.
- [15] P. Smyth, "Clustering sequences with hidden Markov models," *Advances in Neural Information Processing Systems*, pp. 648–654, 1997.
- [16] R. Durbin, S. R. Eddy, A. Krogh, and M. Graeme, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, Cambridge, UK, 1998.
- [17] T. Dalenius, "Towards a methodology for statistical disclosure control," *Statistik Tidskrift*, vol. 15, no. 429–444, pp. 2–1, 1977.
- [18] P. Samarati and L. Sweeney, *Protecting Privacy when Disclosing Information: K-Anonymity and its Enforcement through Generalization and Suppression*, Carnegie Mellon University, Pittsburgh, PA, USA, 1998.
- [19] A. Machanavajhala, J. Gehrke, D. Kifer, and M. Venkatasubramanian, "l-diversity: privacy beyond K-anonymity," in *Proceedings of the International Conference on Data Engineering*, pp. 24–35, IEEE, Atlanta, GA, USA, April 2006.
- [20] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: privacy beyond k-anonymity and l-diversity," in *Proceedings of the International Conference on Data Engineering*, pp. 106–115, IEEE, Istanbul, Turkey, April 2007.
- [21] P. M. V. Kumar and M. Karthikeyan, "L-diversity on k-anonymity with external database for improving privacy preserving data publishing," *International Journal of Computer Applications*, vol. 54, no. 14, 2012.
- [22] F. MeSherry and K. Talwar, "Mechanism design via differential privacy," in *Proceedings of the 48th Annual IEEE Symposium on Foundation of Computer Science*, pp. 94–103, IEEE, Providence, RI, USA, October 2007.
- [23] T. Shang, Z. Zhao, X. Ren, and J. Liu, "Differential identifiability clustering algorithms for big data analysis," *Science China Information Sciences*, vol. 64, no. 5, Article ID 152101, 2021.
- [24] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [25] H. Ding, Y. Tian, C. Peng, Y. Zhang, and S. Xiang, "Inference attacks on genomic privacy with an improved HMM and an

- RCNN model for unrelated individuals,” *Information Sciences*, vol. 512, pp. 207–218, 2020.
- [26] X. J. Ren, T. Shang, and J. W. Liu, “An iterative allocation method of privacy parameter for differential identifiability,” *Journal of Cryptologic Research*, vol. 8, no. 4, pp. 582–590, 2021.
- [27] S. Zhong and J. Ghosh, “A unified framework for model-based clustering,” *Journal of Machine Learning Research*, vol. 4, pp. 1001–1037, 2003.
- [28] J. H. Wu, J. W. Luo, and Y. Wang, “A double k-mean clustering algorithm for sequential gene data based on the hidden Markov model,” *Computer Engineering & Science*, vol. 29, no. 3, pp. 54–56, 2007.
- [29] W. Li and A. Godzik, “Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences,” *Bioinformatics*, vol. 22, no. 13, pp. 1658–1659, 2006.
- [30] A. J. Enright and C. A. Ouzounis, “GeneRAGE: a robust algorithm for sequence clustering and domain detection,” *Bioinformatics*, vol. 16, no. 5, pp. 451–457, 2000.

Research Article

Security and Privacy for Edge-Assisted Internet of Things Security Proof for the SKKE Protocol

Xiangyang Wang ¹, Chunxiang Gu ^{1,2}, Fushan Wei ¹ and Siqi Lu ¹

¹Henan Key Laboratory of Network Cryptography Technology, Zhengzhou, China

²State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China

Correspondence should be addressed to Siqi Lu; 080lusiqi@sina.com

Received 5 July 2021; Accepted 9 November 2021; Published 3 December 2021

Academic Editor: Ding Wang

Copyright © 2021 Xiangyang Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As an Internet of things (IoT) technology, the ZigBee has a wide range of applications in home automation, smart energy, commercial building automation, personal, home and hospital care, telecom, and wireless sensor. The ZigBee standard has the advantage of high reliability, which is based on the security of authentication key agreement protocol, namely, the SKKE protocol. In the ZigBee standard, this protocol based on shared symmetric-key is applied on the security protocol level. It is a full symmetric-key key agreement with key confirmation scheme, while the key confirmation mechanism is provided by a message authentication coding mechanism. In this paper, we consider the security of the SKKE protocol. In the random Oracle model, we reduce the security of the SKKE protocol to the collision of the hash function and the HMAC function and the indistinguishability between the output of the random Oracle and a random number. We also give a theoretical proof with the game-based method. To our knowledge, there is no research on the provable security of the ZigBee protocol at this stage, so it is helpful to promote further research of the ZigBee protocol security.

1. Introduction

The emerging IoT technology usage in different aspects of life results in a very large number of devices connected to each other and to the Internet in a small space. As we have known, IoT devices rely on one or more communication technologies [1–3] (such as ZigBee, Z-Wave, BLE, and WiFi), some of which are well-suited for resource-constrained devices to operate in low-power and lossy networks (LLNs) such as ZigBee and Z-Wave. Among them, the ZigBee protocol, with its low-cost, low-power consumption, high-robustness, and flexibility, has been increasingly applied in the field of short distance communication, daily monitoring, and short distance data transmission. Nowadays, ZigBee-based IoT devices are predominant in the IoT landscape, and the security of the ZigBee plays a more and more important role in the ZigBee communication. In recent years, the researches of the ZigBee security can be divided into implementation level security and specification level security.

At the implementation level, some researchers have studied ZigBee enabled devices in order to find vulnerabilities at the implementation level [4, 5]. These attacks found in the research include replay attack, eavesdropping attack, acquiring key attack, denial of service attack, same-nonce attack, ghost attack, and men-in-the-middle attack. However, these findings do not necessarily reflect the insecurity in the specification. In addition, their methods are usually special and aim to discover low-level vulnerabilities and cannot be effectively transported to the formal analysis of the protocol stack, while, at the specification level, formal verification of ZigBee has been considered in several papers [6–11]. Especially, in [12], the authors focused on the application of formal verification to protocols in the IoT domain and summarized the existing approaches to the formal analysis of the ZigBee protocol. In addition, Li et al. in [13] established the ZigBee symbol model according to the ZigBee standard (ZigBee 1.0 and ZigBee 3.0). This model mainly captures the important elements, such as key sharing,

device joining, and key updating. Next, according to the ZigBee specification, the authors divided the ZigBee protocol into some subprotocols, then the authors used the security protocol verification tool (tamarin) to analyze and verify the security attributes of those subprotocols. Security analysis finds some known security vulnerabilities in ZigBee 1.0 and proves that there are no such vulnerabilities in ZigBee 3.0.

Since the 1980s, two methods for analyzing security protocols have been developed [14]. One method relies on the symbolic model of protocol execution, that is, symbolic (or Dolev Yao or formal) method, in which cryptographic primitives are regarded as black boxes. Symbolic method adopts a highly abstract execution view, in which the messages exchanged by all parties are symbolic terms. This model supports automatic analysis, but the high degree of abstraction makes the security guarantee provided by this method unclear and may miss the possible attacks in the computing model. The other method relies on the computational model which considers the complexity and probability problems, namely, computational (or cryptographic) method. This method captures a powerful security concept and can resist all probabilistic polynomial-time attacks.

All above researches of the ZigBee security at the specification level are in symbolic methods, so it is very necessary and meaningful to take research on the ZigBee security at the specification level under the computational model. In the ZigBee specification, the SKKE protocol is used for two end devices (not including trust center) to establish the application link key for their communication end-to-end (only used in APS layer). Although the process of establishing the end-to-end application link key has been researched in [13], the authors only considered the case that the trust center randomly generates the link key and then distributes it to the initiator and the responder, respectively. As for another case, the initiator and the responder share the master key (randomly generated by the trust center and transmitted to the initiator and the responder, respectively), and then they negotiate the application link key based on the shared master key through the SKKE protocol [15, 16]. The SKKE protocol is a full symmetric-key key agreement with key confirmation mechanism, while message authentication coding mechanism is used to provide key confirmation. Yuksel et al. [15, 16] described the SKKE protocol in detail, but the analysis of the SKKE protocol security is scarce at present. It is necessary to analyze and prove the security of the SKKE protocol. In this paper, the security model of the SKKE protocol is established under the random Oracle model, and its confidentiality is reduced to the collision of hash function and HMAC function and the indistinguishability between random Oracle output and random number. Meanwhile, its authentication is reduced to the collision between hash function and HMAC function, and the proof of the SKKE protocol confidentiality and authentication is given with the game-based method.

The rest of this paper is organized as follows. Section 2 gives the basic knowledge needed in the security proof of the SKKE protocol; Section 3 is the description of the SKKE protocol and its execution details; in Section 4, we model the SKKE protocol and give the definition of indistinguishable

experiment, authentication experiment, and some security notions; in Section 5, we prove the security of the SKKE protocol with the game-based method; Section 6 is the summary of our work.

2. Preliminaries

Denote by $x \leftarrow y$ the operation of assigning y to x . Denote by $x \xleftarrow{\$}$ the operation sampling x uniformly at random from a set X . *PPT* is short for probabilistic polynomial time.

Lemma 1 (see [17]). *Let \mathcal{A} be a distinguisher, then we have*

$$\begin{aligned} & |\Pr[\mathcal{A}(y) = 1 | f \in F^{x,y}, y = f(x)] \\ & - \Pr[\mathcal{A}(y) = 1 | y \xleftarrow{\$} R]| \leq \text{negl}(n), \end{aligned} \quad (1)$$

where $F^{x,y}$: $x \mapsto y$ is a function cluster; $|x| = |y| = n$; $f \in F^{x,y}$ is a random Oracle; $y \xleftarrow{\$}$ denotes sampling y uniformly at random from R ; and $\text{negl}(n)$ is a negligible function.

HMAC [18] is a key-related hash operation message authentication code. HMAC operation uses a hash algorithm to generate a message digest as output with a key and a message as input. A key hash function and a key are used in the definition of HMAC. “text” is used as the plaintext to calculate HMAC; the operation of HMAC function is as follows:

$$H'((K \oplus \text{opad}) \| H'(K \oplus \text{ipad} \| \text{text})), \quad (2)$$

where H' is a hash function; *ipad* and *opad* are two different fixed strings. The detailed operation steps of HMAC function can be referred to [18].

3. The Description of the SKKE Protocol

In this section, we describe the interaction and the execution details of the SKKE protocol.

3.1. The SKKE Protocol. Figure 1 shows the message transmission of the key agreement mechanism, namely, the SKKE protocol [16]. As shown in the figure, U represents the initiator of the protocol and V represents the responder of the protocol. The essential difference between the role of the initiator and the role of the responder is that the initiator sends the first pass of the exchange. As shown in Figure 1, the MK is the master key shared between the initiator U and the responder V . $A \langle B \rangle$ means the transmission of message A that contains payload(s) B . QEX represents the challenge generated by a device X and $MacTag$ represents the hash value. Based on the master key MK , U and V negotiate the session key LK by executing the SKKE protocol.

3.2. Execution Details of the SKKE Protocol. In [15, 16], the authors showed a detailed description of the SKKE protocol. For convenience, let us briefly describe it here.

In the SKKE protocol, an initiator U establishes a LK with a responder V using a shared secret MK . We present the computational details of the SKKE protocol in Figure 2.

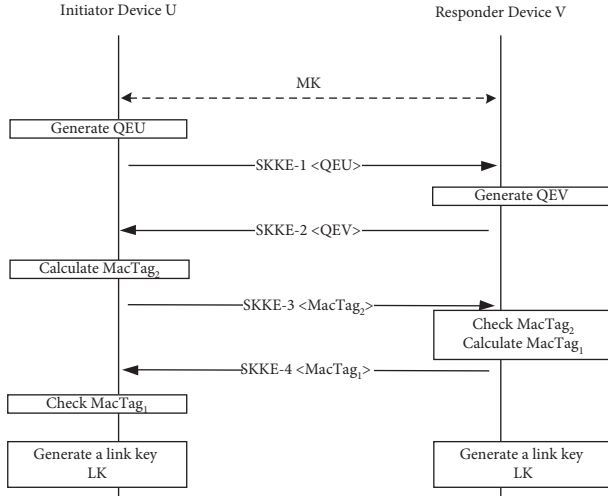


FIGURE 1: Symmetric-key authenticated key agreement scheme.

In the first two messages, the initiator and the responder exchange their challenges. In the last two messages, they exchange the data they have computed using the challenges and their identities. After verifying that they have received the correct value, they use another value as the LK that both of them can compute.

4. Security Model and Security Experiment

4.1. Security Model. We assume that the attacker can completely control the channel, that is, the attacker can eavesdrop, delay, and tamper with the messages transmitted by running the protocol. The following Oracle queries are defined to describe the adversary's ability:

- (i) $\text{Execute}(U, V)$: this query models passive attacks. The output of this query consists of messages that were exchanged during the honest execution of the protocol among U and V .
- (ii) $\text{Send}(U, m)$: this query models active attack by sending message m to the participant U . The output of this query is the message that U would generate on receipt of message m .
- (iii) $\text{Reveal}(U_i, V_i)$: this query models the misuse of session key, namely, link key established between U and V . The output of this query is the link key LK_i established during the i -th communication between U and V . If the i -th communication is used as the test session, this query cannot be executed.
- (iv) $\text{Corrupt}(U_i, V_i)$: this query models the ability of the adversary A to reveal the long-term secret key of the i -th communication between U and V . The output of this query is the master key MK_i shared between U and V during the i -th communication. If the i -th communication is used as the test session, this query cannot be executed.

The adversary A is represented as two procedures A_1 and A_2 that share state. In the RO model, the hash functions H

and H' are modeled as two random oracles, H_A and H'_A , and each of them obtains a list to store queries. When the adversary A queries to H_A , if the query is in the list, the corresponding value is returned, else it returns a random value. It is the same as above that the adversary A queries to H'_A .

4.2. Security Experiment. Before defining the security notion of the SKKE protocol, we define indistinguishability experiment and authentication experiment from the perspectives of confidentiality and authentication of the SKKE protocol.

4.2.1. Indistinguishability Experiment. In this section, we define an indistinguishability experiment (see Figure 3), which is used in the security proof of confidentiality.

The (U_i, V_i) represents i -th ($i = 1, 2, \dots, N$) session established by the initiator U_i and the responder V_i , which is the test session and not revealed. MK_i and LK_i represent the master key and the link key, respectively, which is a bijection. keylen represents the length of the master key and the link key.

Definition 1. For any PPT adversary \mathcal{A} , the advantage of the adversary \mathcal{A} breaks the confidentiality of the SKKE protocol which is defined as follows:

$$\text{Adv}_{\mathcal{A}}^{\text{ind}}(l, N, q_H, q_{H'}) = \left| \Pr \left[\text{Exp}_{\mathcal{A}}^{\text{ind}}(l, N, q_H, q_{H'}) = 1 \right] - \frac{1}{2} \right|. \quad (3)$$

4.2.2. Authentication Experiment. We denote the authentication experiment as $\text{Exp}_{\mathcal{A}}^{\text{auth}}(l, N, q_H, q_{H'})$. Next, we consider the authentication of the initiator U_i to the responder V_i , namely, $\text{Exp}_A^{\text{auth-uv}}(l, N, q_H, q_{H'})$ (see Figure 4), as an example to consider the authentication of the SKKE protocol.

Remark 1. The SKKE protocol is a symmetrical two-way authentication protocol. In Figure 4, $\text{Exp}_A^{\text{auth-uv}}(l, N, q_H, q_{H'})$ denotes the authentication of the initiator U_i to the responder V_i , as for the authentication of the responder V_i to the initiator U_i , similarly.

Definition 2. For any PPT adversary A , the advantage of the adversary A breaks the authentication of the SKKE protocol which is defined as follows:

$$\text{Adv}_{\mathcal{A}}^{\text{auth}}(l, N, q_H, q_{H'}) = \Pr \left[\text{Exp}_{\mathcal{A}}^{\text{auth}}(l, N, q_H, q_{H'}) = 1 \right]. \quad (4)$$

Definition 3. The SKKE protocol has security if for any PPT adversary A , the following advantage is negligible:

		U	V
SKKE-1	$U \rightarrow V$	QEU	
SKKE-2	$V \rightarrow U$	QEV	
Computation		$Z = MAC\{U \parallel V \parallel QEU \parallel QEV\}_{MK}$ $MacKey = H(Z \parallel 0x00000001)$ $KeyData = H(Z \parallel 0x00000002)$ $MacData_2 = 0x03 \parallel U \parallel V \parallel QEU \parallel QEV$ $MacTag_2 = MAC(MacData_2)_{MacKey}$	$Z = MAC\{U \parallel V \parallel QEU \parallel QEV\}_{MK}$ $MacKey = H(Z \parallel 0x00000001)$ $KeyData = H(Z \parallel 0x00000002)$ $MacData_1 = 0x02 \parallel V \parallel U \parallel QEV \parallel QEU$ $MacTag_1 = MAC(MacData_1)_{MacKey}$
SKKE-3	$U \rightarrow V$	$MacTag_2$	
SKKE-4	$V \rightarrow U$	$MacTag_1$	
Verification		$MacData_1 = 0x02 \parallel V \parallel U \parallel QEV \parallel QEU$ Verify $MacTag_1$ using $MacData_1$	$MacData_2 = 0x03 \parallel V \parallel U \parallel QEV \parallel QEU$ Verify $MacTag_2$ using $MacData_2$
U and V use $KeyData = H(MAC\{U \parallel V \parallel QEU \parallel QEV\}_{MK} \parallel 0x00000002)$ as the new LK			

FIGURE 2: The SKKE protocol (H, hash function MAC, HMAC function; \parallel , concatenation; 0x, hexadecimal).

$$Adv_A^{SKKE}(l, N, q_H, q_{H'}) = \max \left\{ \Pr \left[\left| Exp_A^{ind}(l, N, q_H, q_{H'}) \right| = 1 \right] - \frac{1}{2}, \Pr \left[Exp_A^{auth}(l, N, q_H, q_{H'}) = 1 \right] \right\}. \quad (5)$$

5. Security Proof

In this section, firstly, we prove the confidentiality (namely, indistinguishable security) and authentication of the SKKE protocol, namely, Theorem 1 and Theorem 2. In the random Oracle model, we reduce the confidentiality to the collision of hash function and HMAC function and the indistinguishability between the output of random Oracle and

random number, while the authentication is reduced to the collision of the hash function and the HMAC function. Then, we show the SKKE protocol is secure by Theorem 3.

Theorem 1. *For any PPT adversary A against the indistinguishability experiment, the advantage of the adversary A wins the experiment which satisfies the following inequality:*

$$Adv_{\mathcal{A}}^{ind}(l, N, q_H, q_{H'}) \leq \frac{1}{2^{keylen}} + \frac{q_H}{2^{hashklen}} + \frac{q_{H'}}{2^{macklen}} + \text{negl}(\text{hashklen}) + \text{negl}(\text{macklen}), \quad (6)$$

where $keylen$ is the length of master key, $hashklen$ and $macklen$ are the output lengths of the hash function and the HMAC function, q_H and $q_{H'}$ are the times of the adversary queries to the hash functions H and H' , $\text{negl}(\text{hashklen})$ and $\text{negl}(\text{macklen})$ are two negligible functions.

Proof. For each game G_i , we denote $Exp_A^{ind}(l, N, q_H, q_{H'}) = 1$ as S_i . The initial game G_0 is defined as Figure 3.

Game G_1 : the transition from G_0 to G_1 is realized by inlining the calculation process of LK_i and the HMAC function, as shown in Figure 5. So, the two games are equivalent. Then, we have

$$\Pr[S_1] = \Pr[S_0]. \quad (7)$$

Game G_2 : if the adversary obtains the MK shared by the initiator and the responder, the adversary can calculate the LK , and then the adversary can win the game. Thus, as shown in Figure 6, in game G_2 , we raise a flag bad_1 when the adversary obtains the MK by guessing. The difference in the

probability of ($b = b'$) in this game and in game G_1 is bounded by the probability of bad_1 in the latter game, while the probability of bad_1 does not exceed $1/2^{keylen}$. Then, we have

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[G_2: bad_1] \leq \frac{1}{2^{keylen}}. \quad (8)$$

Game G_3 : as shown in Figure 7, in this game, we raise a flag bad_2 to represent that the adversary finds a collision of the outer hash function H' of the HMAC function $H'((r_i \oplus opad) \parallel H'(t(r_i \oplus ipad) \parallel Data_i))$, then the adversary \mathcal{A} obtains Z_i . In the RO model, the hash function is idealized as a random Oracle, and an Oracle $H_{\mathcal{A}}'$ is randomly selected from the function cluster $F^{x,y}: x \mapsto y$ to replace the hash function H' . Then, according to the lemma, that is, the output of the random Oracle which is indistinguishable from the random number, we randomly and evenly select λ_i and replace Z_i with λ_i . Therefore, the upper bound of probability difference of G_2 to G_3 transformation is characterized by the

$\text{Exp}_A^{\text{ind}}(I, N, q_H, q_{H'}) :$ $(U_i, V_i) \leftarrow A_1(0) ;$ $(MK_i, LK_i) \leftarrow \text{ses}(U_i, V_i) ;$ $b \leftarrow \overset{\$}{\leftarrow} \{0, 1\}^l ;$ $LK_i' \leftarrow \overset{\$}{\leftarrow} \{0, 1\}^{\text{keylen}} ;$ $b' \leftarrow A_2(b ? LK_i, LK_i') ;$ $\text{return } (b = b')$	<p>Oracles :</p> <p>Send(U_i, invoke) :</p> $QEU_i \leftarrow \{0, 1\}^l ;$ $\text{return } (QEU_i)$ <p>Send(V_i, QEU_i) :</p> $QEV_i \leftarrow \{0, 1\}^l ;$ $Z_i \leftarrow \text{MAC}\{U \parallel V \parallel QEU_i \parallel QEV_i\}_{MK_i} ;$ $\text{MacKey}_i \leftarrow H(Z_i \parallel 0x00000001) ;$ $\text{MacData}_i \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i \parallel QEU_i ;$ $\text{MacTag}_i \leftarrow \text{MAC}(\text{MacData}_i)_{\text{MacKey}_i} ;$ $\text{return } (QEV_i, \text{MacTag}_i)$ <p>Send(U_i, (QEV_i, MacTag_i)) :</p> $Z_i \leftarrow \text{MAC}\{U \parallel V \parallel QEU_i \parallel QEV_i\}_{MK_i} ;$ $\text{MacKey}_i \leftarrow H(Z_i \parallel 0x00000001) ;$ $\text{MacData}_2 \leftarrow 0x02 \parallel U \parallel V \parallel QEU_i \parallel QEV_i ;$ $\text{MacTag}_{2_i} \leftarrow \text{MAC}(\text{MacData}_{2_i})_{\text{MacKey}_i} ;$ $\text{MacData}_i \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i \parallel QEU_i ;$ $\text{if } (\text{MAC}(\text{MacData}_i)_{\text{MacKey}_i} = \text{MacTag}_{2_i})$ $\quad \text{ses}(U_i, V_i) \leftarrow (MK_i, LK_i) ;$ $\quad \text{return } (\text{MacTag}_{2_i})$ $\text{else return } \perp$ <p>Send(V_i, MacTag_{2_i}) :</p> $\text{MacData}_{2_i} \leftarrow 0x02 \parallel U \parallel V \parallel QEU_i \parallel QEV_i ;$ $\text{if } (\text{MAC}(\text{MacData}_{2_i})_{\text{MacKey}_i} = \text{MacTag}_{2_i})$ $\quad \text{ses}(U_i, V_i) \leftarrow (MK_i, LK_i) ;$ $\text{else return } \perp$ <p>Reveal(U_i, V_i) :</p> $\text{return } (LK_i)$ <p>Corrupt(U_i, V_i) :</p> $\text{return } (MK_i)$
---	---

FIGURE 3: Indistinguishability experiment of the SKKE protocol.

$\text{Exp}_A^{\text{auth-uv}}(I, N, q_H, q_{H'}) :$ $((U_i, V_i), MK_i') \leftarrow A_1(0) ;$ $(MK_i, LK_i) \leftarrow \text{ses}(U_i, V_i) ;$ $QEU_i \leftarrow \overset{\$}{\leftarrow} \{0, 1\}^l ;$ $(QEV_i', \text{MacTag}_i') \leftarrow A_2(QEU_i, MK_i') ;$ $Z_i \leftarrow \text{MAC}\{U \parallel V \parallel QEU_i \parallel QEV_i'\}_{MK_i} ;$ $\text{MacKey}_i \leftarrow H(Z_i \parallel 0x00000001) ;$ $\text{MacData}_i \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i' \parallel QEU_i ;$ $\text{MacTag}_i \leftarrow \text{MAC}(\text{MacData}_i)_{\text{MacKey}_i} ;$ $\text{return } (\text{MacTag}_i = \text{MacTag}_i')$	<p>Oracles :</p> <p>Send(V_i, QEU_i, MK_i') :</p> $QEV_i' \leftarrow \overset{\$}{\leftarrow} \{0, 1\}^l ;$ $Z_i' \leftarrow \text{MAC}\{U \parallel V \parallel QEU_i \parallel QEV_i'\}_{MK_i} ;$ $\text{MacKey}_i' \leftarrow H(Z_i' \parallel 0x00000001) ;$ $\text{MacData}_i \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i' \parallel QEU_i ;$ $\text{MacTag}_i' \leftarrow \text{MAC}(\text{MacData}_i)_{\text{MacKey}_i'} ;$ $\text{return } (QEV_i', \text{MacTag}_i')$ <p>Reveal(U_i, V_i) :</p> $\text{return } (LK_i)$ <p>Corrupt(U_i, V_i) :</p> $\text{return } (MK_i)$
--	--

FIGURE 4: Authentication experiment of the SKKE protocol.

<p>Game G_1:</p> <p>$(U_i, V_i) \leftarrow A_1()$; $(MK_i, LK_i) \leftarrow ses(U_i, V_i)$; $b \leftarrow \mathcal{S}\{0, 1\}^l$; $LK'_i \leftarrow \mathcal{S}\{0, 1\}^{keylen}$; $Data_i \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i$; $b' \leftarrow A_2(b ? H(H'((MK_i \oplus opad) \parallel H'((MK_i \oplus ipad) \parallel Data_i))) \parallel 0x00000002), LK'_i)$; return $(b = b')$</p>
<p>Oracles:</p> <p>Send(U_i, invoke): $QEU_i \leftarrow \{0, 1\}^l$; return (QEU_i)</p> <p>Send(V_i, QEU_i): $QEV_i \leftarrow \{0, 1\}^l$; $Data_i \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i$; $Z_i \leftarrow H'((MK_i \oplus opad) \parallel H'(MK_i \oplus ipad) \parallel Data_i))$; $MacKey_i \leftarrow H(Z_i \parallel 0x00000001)$; $MacData_{1_i} \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i \parallel QEU_i$; $MacTag_{1_i} \leftarrow H'((MacKey_i \oplus opad) \parallel H'(MacKey_i \oplus ipad) \parallel MacData_{1_i}))$; return $(QEV_i, MacTag_{1_i})$</p> <p>Send(U_i, ($QEV_i, MacTag_{1_i}$)): $Z_i \leftarrow H'((MK_i \oplus opad) \parallel H'(MK_i \oplus ipad) \parallel Data_i))$; $MacKey_i \leftarrow H(Z_i \parallel 0x00000001)$; $MacData_{2_i} \leftarrow 0x02 \parallel U \parallel V \parallel QEU_i \parallel QEV_i$; $MacTag_{2_i} \leftarrow H'((MacKey_i \oplus opad) \parallel H'(MacKey_i \oplus ipad) \parallel MacData_{2_i}))$; $MacData_{1_i} \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i \parallel QEU_i$; if $(H'((MacKey_i \oplus opad) \parallel H'(MacKey_i \oplus ipad) \parallel MacData_{1_i})) = MacTag_{1_i}$ $ses(U_i, V_i) \leftarrow (MK_i, H(Z_i \parallel 0x00000001))$; return $(MacTag_{2_i})$ else return \perp</p> <p>Send(V_i, $MacTag_{2_i}$): $MacData_{2_i} \leftarrow 0x02 \parallel U \parallel V \parallel QEU_i \parallel QEV_i$; if $(MAC(MacData_{2_i})_{MacKey_i} = MacTag_{2_i})$; $ses(U_i, V_i) \leftarrow (MK_i, H(Z_i \parallel 0x00000001))$; else return \perp</p> <p>Reveal(U_i, V_i): return (LK_i)</p> <p>Corrupt(U_i, V_i): return (MK_i)</p>

FIGURE 5: Game G_1 of indistinguishability security of the SKKE protocol.

probability of bad_2 event occurrence and the indistinguishability between the output of the random Oracle and a random number. Then, we can get

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[G_3: bad_2] + \text{negl}(macklen) \leq \frac{q_{H'}}{2^{macklen}} + \text{negl}(macklen). \quad (9)$$

<p>Game G_2 :</p> <p>$(U_i, V_i) \leftarrow A_1()$;</p> <p>$(MK_i, LK_i) \leftarrow ses(U_i, V_i)$;</p> <p>$b \xleftarrow{\\$} \{0,1\}^l$; $LK'_i \xleftarrow{\\$} \{0,1\}^{keylen}$;</p> <p>$Data_i \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i$; $r_i \xleftarrow{\\$} \{0,1\}^{keylen}$;</p> <p>if $(r_i = MK_i)$ then</p> <p> $bad_1 \leftarrow true$;</p> <p>else</p> <p> $b' \leftarrow A_2(b ? H(H'((r_i \oplus opad) \parallel H'(r_i \oplus ipad) \parallel Data_i)) \parallel 0x00000002), LK'_i)$;</p> <p> return $(b = b')$</p>
<p>Oracles :</p> <p>Send(U_i, invoke) :</p> <p> $QEU_i \leftarrow \{0,1\}^l$;</p> <p> return (QEU_i)</p> <p>Send(V_i, QEU_i) :</p> <p> $QEV_i \leftarrow \{0,1\}^l$; $Data_i \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i$;</p> <p> $Z_i \leftarrow H'((r_i \oplus opad) \parallel H(r_i \oplus ipad) \parallel Data_i)$;</p> <p> $MacKey_i \leftarrow H(Z_i \parallel 0x00000001)$;</p> <p> $MacData_i \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i \parallel QEU_i$;</p> <p> $MacTag_i \leftarrow H'((MacKey_i \oplus opad) \parallel H'(MacKey_i \oplus ipad) \parallel MacData_i)$;</p> <p> return $(QEV_i, MacTag_i)$</p> <p>Send(U_i, $(QEV_i, MacTag_i)$) :</p> <p> $Z_i \leftarrow H'((r_i \oplus opad) \parallel H(r_i \oplus ipad) \parallel Data_i)$;</p> <p> $MacKey_i \leftarrow H(Z_i \parallel 0x00000001)$;</p> <p> $MacData_2 \leftarrow 0x02 \parallel U \parallel V \parallel QEU_i \parallel QEV_i$;</p> <p> $MacTag_2 \leftarrow H'((MacKey_i \oplus opad) \parallel H'(MacKey_i \oplus ipad) \parallel MacData_2)$;</p> <p> $MacData_i \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i \parallel QEU_i$;</p> <p> if $(H'((MacKey_i \oplus opad) \parallel H'(MacKey_i \oplus ipad) \parallel MacData_i)) = MacTag_i$</p> <p> $ses(U_i, V_i) \leftarrow (r_i, H(Z_i \parallel 0x00000001))$;</p> <p> return $(MacTag_2)$</p> <p> else return \perp</p> <p>Send(V_i, $MacTag_2$) :</p> <p> $MacData_2 \leftarrow 0x02 \parallel U \parallel V \parallel QEU_i \parallel QEV_i$;</p> <p> if $(H'((MacKey_i \oplus opad) \parallel H'(MacKey_i \oplus ipad) \parallel MacData_2)) = MacTag_2$;</p> <p> $ses(U_i, V_i) \leftarrow (r_i, H(Z_i \parallel 0x00000001))$;</p> <p> else return \perp</p> <p>Reveal(U_i, V_i) :</p> <p> return (LK_i)</p> <p>Corrupt(U_i, V_i) :</p> <p> return (MK_i)</p>

FIGURE 6: Game G_2 of indistinguishability security of the SKKE protocol.

Game G_4 : as shown in Figure 8, in this game G_4 , we raise a flag bad_3 to represent that the adversary finds a collision of the hash function H , then the adversary \mathcal{A} obtains LK_i . In the RO model, the hash function is idealized as a random Oracle, and an Oracle $H_{\mathcal{A}}$ is randomly selected from the

function cluster $F^{x,y}: x \mapsto y$ to replace the hash function H . Then, according to the lemma, that is, the output of the random Oracle which is indistinguishable from the random number, we randomly select γ_i and replace $H_{\mathcal{A}}(\lambda_i \parallel 0x00000002)$ with λ_i . Therefore, the upper bound of

<p>Game G_3 :</p> <p>$(U_i, V_i) \leftarrow A_1(0) ; (MK_i, LK_i) \leftarrow ses(U_i, V_i) ;$ $b \leftarrow \mathcal{S}\{0,1\}^l ; LK'_i \leftarrow \mathcal{S}\{0,1\}^{keylen} ;$ $Data_i \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i ; r_i \leftarrow \mathcal{S}\{0,1\}^{keylen} ;$ if $(r_i = MK_i)$ then $bad_1 \leftarrow true ;$ else $\lambda_i \leftarrow \mathcal{S}\{0,1\}^{macklen} ;$ if $(\lambda_i = H'_A((r_i \oplus opad) \parallel H'_A((r_i \oplus ipad) \parallel Data_i)))$ then $bad_2 \leftarrow true ;$ else $b' \leftarrow A_2(b ? H(\lambda_i \parallel 0x00000002), LK'_i) ;$ return $(b = b')$</p>
<p>Oracles :</p> <p>Send(U_i, invoke) : $QEU_i \leftarrow \{0,1\}^l ;$ return (QEU_i)</p> <p>Send(V_i, QEU_i) : $QEV_i \leftarrow \{0,1\}^l ;$ $MacKey_i \leftarrow H(\lambda_i \parallel 0x00000001) ;$ $MacData_{i_1} \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i \parallel QEU_i ;$ $MacTag_{i_1} \leftarrow H'_A((MacKey_i \oplus opad) \parallel H'_A(MacKey_i \oplus ipad) \parallel MacData_{i_1}) ;$ return $(QEV_i, MacTag_{i_1})$</p> <p>Send($U_i, (QEV_i, MacTag_{i_1})$) : $MacKey_i \leftarrow H(\lambda_i \parallel 0x00000001) ;$ $MacData_{i_2} \leftarrow 0x02 \parallel U \parallel V \parallel QEU_i \parallel QEV_i ;$ $MacTag_{i_2} \leftarrow H'_A((MacKey_i \oplus opad) \parallel H'_A(MacKey_i \oplus ipad) \parallel MacData_{i_2}) ;$ $MacData_{i_1} \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i \parallel QEU_i ;$ if $(H'_A((MacKey_i \oplus opad) \parallel H'_A(MacKey_i \oplus ipad) \parallel MacData_{i_1})) = MacTag_{i_1}$ $ses(U_i, V_i) \leftarrow (r_i, H(\lambda_i \parallel 0x00000001)) ;$ return $(MacTag_{i_2})$ else return \perp</p> <p>Send($V_i, MacTag_{i_2}$) : $MacData_{i_2} \leftarrow 0x02 \parallel U \parallel V \parallel QEU_i \parallel QEV_i ;$ if $(H'_A((MacKey_i \oplus opad) \parallel H'_A(MacKey_i \oplus ipad) \parallel MacData_{i_2})) = MacTag_{i_2}$ $ses(U_i, V_i) \leftarrow (r_i, H(\lambda_i \parallel 0x00000001)) ;$ else return \perp</p> <p>Reveal(U_i, V_i) : return (LK_i)</p> <p>Corrupt(U_i, V_i) : return (MK_i)</p>

FIGURE 7: Game G_3 of indistinguishability security of the SKKE protocol.

<p>Game G_4 :</p> <p>$(U_i, V_i) \leftarrow A_1(0)$; $(MK_i, LK_i) \leftarrow ses(U_i, V_i)$; $b \leftarrow \overset{\\$}{\leftarrow} \{0,1\}^l$; $LK'_i \leftarrow \overset{\\$}{\leftarrow} \{0,1\}^{keylen}$; $Data_i \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i$; $r_i \leftarrow \overset{\\$}{\leftarrow} \{0,1\}^{keylen}$; if $(r_i = MK_i)$ then $bad_1 \leftarrow true$; else $\lambda_i \leftarrow \overset{\\$}{\leftarrow} \{0,1\}^{macklen}$; if $(\lambda_i = H'_A((r_i \oplus opad) \parallel H'_A((r_i \oplus ipad) \parallel Data_i)))$ then $bad_2 \leftarrow true$; else $\gamma_i \leftarrow \overset{\\$}{\leftarrow} \{0,1\}^{hashklen}$; if $(\gamma_i = H(\lambda_i \parallel 0x00000002))$ then $bad_3 \leftarrow true$; else $b' \leftarrow A_2(b ? \gamma_i, LK'_i)$; return $(b = b')$</p>
<p>Oracles :</p> <p>Send(U_i, invoke): $QEU_i \leftarrow \{0,1\}^l$; return (QEU_i)</p> <p>Send(V_i, QEU_i): $QEV_i \leftarrow \{0,1\}^l$; $MacKey_i \leftarrow H(\lambda_i \parallel 0x00000001)$; $MacData_i \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i \parallel QEU_i$; $MacTag_i \leftarrow H'_A((MacKey_i \oplus opad) \parallel H'_A(MacKey_i \oplus ipad) \parallel MacData_i)$; return $(QEV_i, MacTag_i)$</p> <p>Send($U_i, (QEV_i, MacTag_i)$): $MacKey_i \leftarrow H(\lambda_i \parallel 0x00000001)$; $MacData_{2_i} \leftarrow 0x02 \parallel U \parallel V \parallel QEU_i \parallel QEV_i$; $MacTag_{2_i} \leftarrow H'_A((MacKey_i \oplus opad) \parallel H'_A(MacKey_i \oplus ipad) \parallel MacData_{2_i})$; $MacData_{1_i} \leftarrow 0x03 \parallel V \parallel U \parallel QEV_i \parallel QEU_i$; if $(H'_A((MacKey_i \oplus opad) \parallel H'_A(MacKey_i \oplus ipad) \parallel MacData_{1_i})) = MacTag_{1_i}$ $ses(U_i, V_i) \leftarrow (r_i, \gamma_i)$; return $(MacTag_{2_i})$ else return \perp</p> <p>Send($V_i, MacTag_{2_i}$): $MacData_{2_i} \leftarrow 0x02 \parallel U \parallel V \parallel QEU_i \parallel QEV_i$; if $(H'_A((MacKey_i \oplus opad) \parallel H'_A(MacKey_i \oplus ipad) \parallel MacData_{2_i})) = MacTag_{2_i}$; $ses(U_i, V_i) \leftarrow (r_i, \gamma_i)$; else return \perp</p> <p>Reveal(U_i, V_i): return (LK_i)</p> <p>Corrupt(U_i, V_i): return (MK_i)</p>

FIGURE 8: Game G_4 of indistinguishability security of the SKKE protocol.

the probability difference of G_3 to G_4 transformation is characterized by the probability of bad_3 event occurrence

and the indistinguishability between the output of the random Oracle and a random number. Then, we have

$$|\Pr[S_4] - \Pr[S_3]| \leq \Pr[G_4: bad_3] + \text{negl}(\text{hashklen}) \leq \frac{q_H}{2^{\text{hashklen}}} + \text{negl}(\text{hashklen}), \quad (10)$$

where hashklen is the output length of the hash function and $\text{negl}(\text{hashklen})$ is a negligible function. While in G_4 , the protocol copies returned are random numbers that have nothing to do with coin tossing, then we have

$$\Pr[S_4] = \frac{1}{2} \quad (11)$$

In summary, from (1) to (5), we can get the adversary's advantage in winning as follows:

$$\text{Adv}_{\mathcal{A}}^{\text{ind}}(l, N, q_H, q_{H'}) \leq \frac{1}{2^{\text{keylen}}} + \frac{q_H}{2^{\text{hashklen}}} + \frac{q_{H'}}{2^{\text{macklen}}} + \text{negl}(\text{hashklen}) + \text{negl}(\text{macklen}). \quad (12)$$

Theorem 2. For any PPT adversary \mathcal{A} against the authentication experiment, the advantage of the adversary winning satisfies the following inequality:

$$\text{Adv}_{\mathcal{A}}^{\text{auth}}(l, N, q_H, q_{H'}) \leq \frac{1}{2^{\text{keylen}-1}} + \frac{q_H}{2^{\text{hashklen}-1}} + \frac{q_{H'}}{2^{\text{macklen}-1}}, \quad (13)$$

where keylen is the length of master key; hashklen and macklen are the output lengths of the HMAC function and the hash function; q_H is the times of the adversary queries to the hash function H ; and $q_{H'} = s_1 + s_2$ is the total times of the adversary queries to the hash function H' .

Proof. For each game G_i , we denote $\text{Exp}_{\mathcal{A}}^{\text{ind}}(l, N, q_H, q_{H'}) = 1$ as S_i . The initial game G_0 is defined as Figure 4. \square

Game G_1 : as shown in Figure 9, the transition from G_0 to G_1 is realized by in-lining the calculation process of HMAC function. Thus, the two games are equivalent. Then, we can have

$$\Pr[S_1] = \Pr[S_0]. \quad (14)$$

Game G_2 : as shown in Figure 10, we raise a flag bad_1 to represent that the adversary finds a collision of the hash function of H' , then the adversary \mathcal{A} obtains a MacKey'_i , which satisfies the following conditions:

$$\text{MacTag}_{g_1} = \text{MacTa}'_{g_1}, \quad (15)$$

$$(\text{MacKey}_i \oplus \text{opad}) \parallel H'(\text{MacKey}_i \oplus \text{ipad}) \parallel \text{MacData}_{1_i} \neq (\text{MacKey}'_i \oplus \text{opad}) \parallel H'(\text{MacKey}'_i \oplus \text{ipad}) \parallel \text{MacData}_{1_i}. \quad (16)$$

Since the probability of the bad_1 event happening is not more than $s_1/2^{\text{macklen}}$, then we can have

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[G_2: bad_1] \leq \frac{s_1}{2^{\text{macklen}}}, \quad (17)$$

where s_1 is the number of times the adversary queries to H' in this game.

Game G_3 : as shown in Figure 11, we raise a flag bad_2 to represent that the adversary finds a collision of the hash function H , then the adversary \mathcal{A} obtains a Z'_i , which satisfies the following conditions: $\text{MacKey}_i = \text{MacKey}'_i$ and $Z_i \neq Z'_i$. Since the probability of the bad_2 event happening is not more than $q_H/2^{\text{hashklen}}$, then we can have

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[G_3: bad_2] \leq \frac{q_H}{2^{\text{hashklen}}}. \quad (18)$$

Game G_4 : as shown in Figure 12, we raise a flag bad_3 to represent that the adversary finds a collision of the hash

function of H' , then the adversary \mathcal{A} obtains a MK'_i , which satisfies $Z_i = Z'_i$ and $\text{MK}_i \neq \text{MK}'_i$. Since the probability of the bad_3 event happening is not more than $s_2/2^{\text{macklen}}$, then we can have

$$|\Pr[S_4] - \Pr[S_3]| \leq \Pr[G_4: bad_3] \leq \frac{s_2}{2^{\text{macklen}}}, \quad (19)$$

where s_2 is the number of times the adversary queries to H' in this game. Due to the inclusion relationship between events $H'((\text{MacKey}_i \oplus \text{opad}) \parallel H'((\text{MacKey}_i \oplus \text{ipad}) \parallel \text{MacData}_{1_i})) = H'(\text{MacKey}'_i \oplus \text{opad}) \parallel H'((\text{MacKey}'_i \oplus \text{ipad}) \parallel \text{MacData}_{1_i})$, $\text{MacKey}_i = \text{MacKey}'_i$, $Z_i = Z'_i$ and $\text{MK}_i \neq \text{MK}'_i$, $\Pr[S_4] = \Pr[G_4: \text{MK}_i = \text{MK}'_i]$ can be obtained from the conditional probability formula. The probability of the adversary obtains the master key MK_i by guessing which is not more than $1/2^{\text{keylen}}$ (keylen is the length of the master key), so we can have

<p>Game G_1:</p> <p>$((U_i, V_i), MK'_i) \leftarrow A_1()$; $(MK_i, _) \leftarrow ses(U_i, V_i)$; $QEU_i \leftarrow \mathcal{S}\{0, 1\}^l$; $(QEV'_i, MacTag'_i) \leftarrow A_2(QEU_i, MK'_i)$; $Data_i \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i$; $Z_i \leftarrow H'(MK_i \oplus opad) \parallel H'((MK_i \oplus ipad) \parallel Data_i))$; $MacKey_i \leftarrow H(Z_i \parallel 0x00000001)$; $MacData_{i_1} \leftarrow 0x03 \parallel V \parallel U \parallel QEV'_i \parallel QEU_i$; return $(H'((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_{i_1})))$ $= H'(MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$</p>
<p>Oracles:</p> <p>Send(V_i, QEU_i, MK'_i):</p> <p>$QEV'_i \leftarrow \mathcal{S}\{0, 1\}^l$; $Data_i \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i$ $Z'_i \leftarrow H'(MK'_i \oplus opad) \parallel H'((MK'_i \oplus ipad) \parallel Data_i))$; $MacKey'_i \leftarrow H(Z'_i \parallel 0x00000001)$; $MacData_{i_1} \leftarrow 0x03 \parallel V \parallel U \parallel QEV'_i \parallel QEU_i$; $MacTag'_{i_1} \leftarrow H'(MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$; return $(QEV'_i, MacTag'_{i_1})$</p> <p>Corrupt(U_i, V_i):</p> <p>return (MK_i)</p>

FIGURE 9: Game G_1 of the authentication of the SKKE protocol.

<p>Game G_2:</p> <p>$((U_i, V_i), MK'_i) \leftarrow A_1()$; $(MK_i, LK_i) \leftarrow ses(U_i, V_i)$; $QEU_i \leftarrow \mathcal{S}\{0, 1\}^l$; $(QEV'_i, MacTag'_i) \leftarrow A_2(QEU_i, MK'_i)$; $Z_i \leftarrow MAC\{U \parallel V \parallel QEU_i \parallel QEV'_i\}_{MK_i}$; $MacKey_i \leftarrow H(Z_i \parallel 0x00000001)$; $MacData_{i_1} \leftarrow 0x03 \parallel V \parallel U \parallel QEV'_i \parallel QEU_i$; if $(H'((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_{i_1})))$ $= H'(MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$ $\wedge ((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_{i_1}))$ $\neq MacKey'_i \oplus opad \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$ then $bad_{i_1} \leftarrow true$; else return $(H'((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_{i_1})))$ $= H'(MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$ $\wedge (MacKey_i = MacKey'_i)$</p>
<p>Oracles:</p> <p>Send(V_i, QEU_i, MK'_i):</p> <p>$QEV'_i \leftarrow \mathcal{S}\{0, 1\}^l$; $Data_i \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i$ $Z'_i \leftarrow H'(MK'_i \oplus opad) \parallel H'((MK'_i \oplus ipad) \parallel Data_i))$; $MacKey'_i \leftarrow H(Z'_i \parallel 0x00000001)$; $MacData_{i_1} \leftarrow 0x03 \parallel V \parallel U \parallel QEV'_i \parallel QEU_i$; $MacTag'_{i_1} \leftarrow H'(MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$; return $(QEV'_i, MacTag'_{i_1})$</p> <p>Corrupt(U_i, V_i):</p> <p>return (MK_i)</p>

FIGURE 10: Game G_2 of the authentication of the SKKE protocol.

<p>Game G_3 :</p> <p>$((U_i, V_i), MK'_i) \leftarrow A_1(); (MK_i, LK_i) \leftarrow ses(U_i, V_i);$ $QEU_i \leftarrow \mathcal{S}\{0, 1\}^l; (QEV'_i, MacTag'_i) \leftarrow A_2(QEU_i, MK'_i);$ $Z_i \leftarrow MAC\{U \parallel V \parallel QEU_i \parallel QEV'_i\}_{MK_i};$ $MacKey_i \leftarrow H(Z_i \parallel 0x00000001);$ $MacData_i \leftarrow 0x03 \parallel V \parallel U \parallel QEV'_i \parallel QEU_i;$ if $(H'((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_i)))$ $= H'((MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_i))$ $\wedge ((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_i))$ $\neq MacKey'_i \oplus opad \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_i))$ then $bad_1 \leftarrow true$; else if $(H'((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_i)))$ $= H'((MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_i))$ $\wedge (H(Z_i \parallel 0x00000001) = H(Z'_i \parallel 0x00000001))$ $\wedge (Z_i \neq Z'_i)$ then $bad_2 \leftarrow true$; else return $(H'((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_i)))$ $= H'((MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_i))$ $\wedge (MacKey_i = MacKey'_i)$ $\wedge (Z_i = Z'_i)$</p>
<p>Oracles :</p> <p>Send(V_i, QEU_i, MK'_i) :</p> <p>$QEV'_i \leftarrow \mathcal{S}\{0, 1\}^l; Data_i \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i$ $Z'_i \leftarrow H'(MK'_i \oplus opad) \parallel H'((MK'_i \oplus ipad) \parallel Data_i);$ $MacKey'_i \leftarrow H(Z'_i \parallel 0x00000001); MacData_i \leftarrow 0x03 \parallel V \parallel U \parallel QEV'_i \parallel QEU_i;$ $MacTag'_i \leftarrow H'(MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_i);$ return $(QEV'_i, MacTag'_i)$</p> <p>Corrupt(U_i, V_i) :</p> <p>return (MK_i)</p>

FIGURE 11: Game G_3 of the authentication of the SKKE protocol.

$$|\Pr[S_4] = \Pr[G_4: MK'_i = MK_i]| \leq \frac{1}{2^{keylen}}. \quad (20)$$

To sum up, from (6) to (10), we can get the advantage of the adversary against the authentication experiment of U to V , and it satisfies the following inequality:

$$Adv_{\mathcal{A}}^{auth-uv}(l, N, q_H, q_{H'}) \leq \frac{1}{2^{keylen}} + \frac{q_H}{2^{hashklen}} + \frac{q_{H'}}{2^{macklen}}. \quad (21)$$

In a word, the advantage of the adversary against the authentication of the SKKE protocol satisfies

$$Adv_{\mathcal{A}}^{auth}(l, N, q_H, q_{H'}) \leq \frac{1}{2^{keylen-1}} + \frac{q_H}{2^{hashklen-1}} + \frac{q_{H'}}{2^{macklen-1}}. \quad (22)$$

Theorem 3. For any PPT adversary \mathcal{A} against the security of the SKKE protocol, the advantage of the adversary winning satisfies the following inequality:

$$Adv_{\mathcal{A}}^{SKKE}(l, N, q_H, q_{H'}) \leq \frac{1}{2^{keylen-1}} + \frac{q_H}{2^{hashklen-1}} + \frac{q_{H'}}{2^{macklen-1}} + \text{negl}(hashklen) + \text{negl}(macklen). \quad (23)$$

<p>Game G_4 :</p> <p>$((U_i, V_i), MK'_i) \leftarrow A_1(); (MK_i, LK_i) \leftarrow ses(U_i, V_i);$ $QEU_i \leftarrow \mathcal{S}\{0, 1\}^l; (QEV'_i, MacTag'_i) \leftarrow A_2(QEU_i, MK'_i);$ $Z_i \leftarrow MAC\{U \parallel V \parallel QEU_i \parallel QEV'_i\}_{MK_i};$ $MacKey_i \leftarrow H(Z_i \parallel 0x00000001);$ $MacData_{i_1} \leftarrow 0x03 \parallel V \parallel U \parallel QEV'_i \parallel QEU_i;$ if $(H'((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_{i_1})))$ $= H'((MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$ $\wedge ((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_{i_1}))$ $\neq MacKey'_i \oplus opad \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$ then $bad_1 \leftarrow true$; else if $(H'((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_{i_1})))$ $= H'((MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$ $\wedge (H(Z_i \parallel 0x00000001) = H(Z'_i \parallel 0x00000001))$ $\wedge (Z_i \neq Z'_i)$ then $bad_2 \leftarrow true$; else if $(H'((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_{i_1})))$ $= H'((MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$ $\wedge (H(Z_i \parallel 0x00000001) = H(Z'_i \parallel 0x00000001))$ $\wedge (H'((MK_i \oplus opad) \parallel H'((MK_i \oplus ipad) \parallel Data_{i_1})))$ $= H'((MK'_i \oplus opad) \parallel H'((MK'_i \oplus ipad) \parallel Data_{i_1}))$ $\wedge ((MK_i \oplus opad) \parallel H'((MK_i \oplus ipad) \parallel Data_{i_1}))$ $\neq (MK'_i \oplus opad) \parallel H'((MK'_i \oplus ipad) \parallel Data_{i_1}))$ then $bad_3 \leftarrow true$; else return $(H'((MacKey_i \oplus opad) \parallel H'((MacKey_i \oplus ipad) \parallel MacData_{i_1})))$ $= H'((MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1}))$ $\wedge (MacKey_i = MacKey'_i)$ $\wedge (Z_i = Z'_i)$ $\wedge (MK_i = MK'_i)$</p>
<p>Oracles :</p> <p>Send(V_i, QEU_i, MK'_i) :</p> <p>$QEV'_i \leftarrow \mathcal{S}\{0, 1\}^l; Data_{i_1} \leftarrow U \parallel V \parallel QEU_i \parallel QEV_i$ $Z'_i \leftarrow H'(MK'_i \oplus opad) \parallel H'((MK'_i \oplus ipad) \parallel Data_{i_1});$ $MacKey'_i \leftarrow H(Z'_i \parallel 0x00000001); MacData_{i_1} \leftarrow 0x03 \parallel V \parallel U \parallel QEV'_i \parallel QEU_i;$ $MacTag'_i \leftarrow H'(MacKey'_i \oplus opad) \parallel H'((MacKey'_i \oplus ipad) \parallel MacData_{i_1});$ return $(QEV'_i, MacTag'_i)$</p> <p>Corrupt(U_i, V_i) :</p> <p>return (MK_i)</p>

FIGURE 12: Game G_4 of the authentication of the SKKE protocol.

Remark 2. The proof of Theorem 3 is obvious. It should be noted that q_H is the largest times of the adversary queries to the hash function H in these two security experiments, and $q_{H'}$ is the largest total times of the adversary queries to the hash function H' , similarly.

6. Conclusions

As an Internet of things communication technology, ZigBee has a wide range of applications in home automation, smart energy, commercial building automation, personal, home and hospital care, telecom, and wireless sensor. At the same time, its security has also received more and more attention. According to the standard of ZigBee 3.0, this paper considers the security of the SKKE protocol, which is used by the initiator and the responder to establish the application link key. In the random Oracle model, we reduce the confidentiality of the SKKE protocol to the collision of the hash function and the HMAC function and the indistinguishability between the output of random Oracle and random number. Then, we reduce the authentication to the collision between the hash function and the HMAC function. Finally, we give the theoretical proof of the confidentiality and authentication of the SKKE protocol with the game-based method. As we known, there is a lack of research on the provable security of the ZigBee protocol at this stage, so it is a meaningful work to study the provable security of the SKKE protocol, and it is helpful to promote further research of the ZigBee protocol security.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

- [1] C. Ge, Z. Liu, J. Xia, and L. Fang, "Revocable identity-based broadcast proxy Re-encryption for data sharing in clouds," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1214–1226, 2021.
- [2] C. Ge, W. Susilo, Z. Liu, J. Xia, P. Szalachowski, and F. Liming, "Secure keyword search and data sharing mechanism for cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, p. 1, 2020.
- [3] J. Durech and M. Franekova, "Security attacks to ZigBee technology and their practical realization," *2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, IEEE, in *Proceedings of the 2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pp. 345–349, January 2014.
- [4] L. N. Whitehurst, T. R. Andel, and J. T. McDonald, "Exploring security in ZigBee networks," *Proceedings of the 9th Annual Cyber and Information Security Research Conference on - CISR '14*, ACM DL, in *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, pp. 25–28, April 2014.
- [5] X. Cao, D. M. Shila, Z. Yang, Y. Zhou, and J. Chen, "Ghost-in-Zigbee: Energy depletion attack on zigbee-based wireless networks," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 816–829, 2016.
- [6] A. Gawanmeh, "Embedding and verification of ZigBee protocol stack in event-B," *Procedia Computer Science*, vol. 5, pp. 736–741, 2011.
- [7] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "A verifiable and fair attribute-based proxy Re-encryption scheme for data sharing in clouds," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2021.
- [8] R. M. Nadeem and A. A. Gill, "A formal model for verification of ZigBee protocol for secure network authentication," *Indian Journal of Science and Technology*, vol. 10, no. 20, pp. 1–7, 2017.
- [9] A. P. Melaragno, D. Bandara, D. Wijesekera, and J. B. Michael, "Securing the ZigBee protocol in the Smart grid," *Computer*, vol. 45, no. 4, pp. 92–94, 2012.
- [10] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "Revocable attribute-based encryption with data integrity in clouds," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2021.
- [11] M. Alshahrani, I. Traore, and I. Woungang, "Anonymous mutual IoT interdevice authentication and key agreement scheme based on the ZigBee technique," *Internet of Things*, vol. 7, p. 100061, Article ID 100061, 2019.
- [12] K. Hofer-Schmitz and B. Stojanović, "Towards formal verification of IoT protocols: a Review," *Computer Networks*, vol. 174, p. 107233, 2020.
- [13] L. Li, P. Podder, and E. Hoque, "A formal security analysis of ZigBee (1.0 and 3.0)," *Proceedings of the 7th Symposium on Hot Topics in the Science of Security*, ACM DL, in *Proceedings of the 7th Symposium on Hot Topics in the Science of Security*, pp. 1–11, September 2020.
- [14] B. Blanchet, "Mechanizing game-based proofs of security protocols," *Software Safety and Security*, vol. 33, pp. 1–25, 2012.
- [15] E. Yuksel, H. R. Nielson, and F. Nielson, "ZigBee-2007 security essentials," in *Proceedings of the 13th Nordic Workshop on Secure IT Systems*, pp. 65–82, Technical University of Denmark, Kongens Lyngby, Denmark, October 2008.
- [16] *ZigBee Specifications*, 2015, <https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf>.
- [17] M. Bellare and P. Rogaway, "Random oracles are practical," *Proceedings of the 1st ACM conference on Computer and communications security - CCS '93*, Computer and Communications Security, in *Proceedings of the 1st ACM conference on Computer and communications security - CCS '93*, pp. 62–73, December 1993.
- [18] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: keyed-hashing for message authentication," *RFC*, vol. 2104, pp. 1–11, 1997.

Research Article

Achieve Efficient and Privacy-Preserving Compound Substring Query over Cloud

Fan Yin,^{1,2} Rongxing Lu ,² Yandong Zheng,² and Xiaohu Tang¹

¹The Information Security and National Computing Grid Laboratory, Southwest Jiaotong University, Chengdu 611756, China

²Faculty of Computer Science, University of New Brunswick, Fredericton, Canada

Correspondence should be addressed to Rongxing Lu; rlu1@unb.ca

Received 2 August 2021; Revised 27 September 2021; Accepted 12 October 2021; Published 3 December 2021

Academic Editor: Qi Jiang

Copyright © 2021 Fan Yin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The cloud computing technique, which was initially used to mitigate the explosive growth of data, has been required to take both data privacy and users' query functionality into consideration. Searchable symmetric encryption (SSE) is a popular solution that can support efficient attribute queries over encrypted datasets in the cloud. In particular, some SSE schemes focus on the substring query, which deals with the situation that the user only remembers the substring of the queried attribute. However, all of them just consider substring queries on a single attribute, which cannot be used to achieve compound substring queries on multiple attributes. This paper aims to address this issue by proposing an efficient and privacy-preserving SSE scheme supporting compound substring queries. In specific, we first employ the position heap technique to design a novel tree-based index to support substring queries on a single attribute and employ pseudorandom function (PRF) and fully homomorphic encryption (FHE) techniques to protect its privacy. Then, based on the homomorphism of FHE, we design a filter algorithm to calculate the intersection of search results for different attributes, which can be used to support compound substring queries on multiple attributes. Detailed security analysis shows that our proposed scheme is privacy-preserving. In addition, extensive performance evaluations are also conducted, and the results demonstrate the efficiency of our proposed scheme.

1. Introduction

The rapid development of information techniques has been promoting the explosive growth of data. In order to mitigate the local storage and computing pressure, an increasing number of individuals and organizations tend to store and process their databases in the cloud [1, 2]. However, since the cloud server may not be fully trustable, those databases with some sensitive information (e.g., electronic health records) have to be encrypted before being outsourced to the cloud. Although the encryption technique can preserve database privacy, it also hides some critical information such that the cloud server cannot well support some users' query functionality over the encrypted database, e.g., attribute query, which returns a collection of records containing a specific queried attribute.

To deal with the above challenge, the concept of searchable symmetric encryption (SSE) [3] was introduced,

which enables the cloud server to search on encrypted records in a very efficient way. Over the past year, in order to improve the query efficiency of SSE, a series of secure index techniques have been designed to match the attributes to corresponding records, such as inverted index [4–7] and tree-based index [8]. Since these index techniques are built with exact attributes, the corresponding SSE schemes can only support the exact attribute query; i.e., *the queried attribute must be exactly the same attribute as that stored in cloud*.

Recently, to solve the situation that a user only remembers a substring of an attribute rather than the exact attribute, some studies [9–11] designed SSE schemes to support substring queries. However, they just considered a substring query on a single attribute, which cannot be used to achieve a compound substring query on multiple attributes, i.e., *the queried records match multiple substring queries for multiple attributes at the same time*. Considering

an example of the compound query that a database DB includes n records $\{f_1, f_2, \dots, f_n\}$ and each record in it has ρ attributes $\{A_1, A_2, \dots, A_\rho\}$, a user can send a compound substring query on two attributes: select f from DB where A_j like $*s_1*$ and A_k like $*s_2*$, to query records whose attributes A_j and A_k contain substring s_1 and s_2 , respectively, where j and k belong to $[1, \rho]$. A straightforward solution to support the compound query is that users query substrings separately and then calculate the intersection of results. Unfortunately, this solution is inefficient because it leads to a large number of communication overheads and computational costs for the data user.

To address the above problems, in this paper, we propose a privacy-preserving SSE scheme, which can efficiently support compound substring queries on multiple attributes. In specific, the main contributions of this paper are threefold:

- (i) First, based on the position heap technique, we design a tree-based index to support substring queries on a single attribute. This tree-based index can support two types of substring patterns: $*s*$ and s_1*s_2 , where s , s_1 , and s_2 represent queried substrings and $*$ represents any string of any length. In addition, we employ pseudorandom functions and fully homomorphic encryption techniques to encrypt this tree-based index, which can well preserve the privacy of records.
- (ii) Second, based on the homomorphism of fully homomorphic encryption, we design an algorithm (see Section 4.2.3) to calculate the intersection of search results for different attributes and therefore achieve the compound substring query on multiple attributes.
- (iii) Finally, we analyze the security of our proposed scheme and conduct extensive experiments to evaluate its performance. The results show that our proposed scheme is efficient in terms of computational cost and storage overhead.

The remainder of the paper is organized as follows. We formalize the system model, security model, and design goals in Section 2. Then, we introduce some preliminaries including the position heap technique [12] and the security notion of substring-of-attribute query in Section 3. After that, we present our proposed scheme in Section 4, followed by security analyses and performance evaluation in Section 5 and Section 6, respectively. Some related works are discussed in Section 7. Finally, we draw our conclusions in Section 8.

2. Models and Design Goals

In this section, we formalize the system model, security model, and identify our design goals.

2.1. System Model. In our system model, we consider two entities, namely, a data user and a cloud server, as shown in Figure 1.

- (i) **Data user:** the data user has a database DB with n records $\{f_1, f_2, \dots, f_n\}$ and ρ attributes $\{A_1, A_2, \dots, A_\rho\}$. Each record f_i ($1 \leq i \leq n$) in DB includes a unique identifier id_i and a set of string-type attributes $\{\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{i\rho}\}$. Due to the limited storage space and computational capability, the data user intends to outsource the database DB and its index, i.e., I , to the cloud server. Later, the data user submits a compound substring query token $Q = \{\Omega, q_1, \dots, q_\rho\}$ to the cloud server to retrieve a set of records \mathcal{F} matching Q , where q_j ($1 \leq j \leq \rho$) is a substring query for attribute A_j and Ω is a compound formula consisting of conjunctive expressions (i.e., \cap) and disjunctive expressions (i.e., \cup) on $\{q_1, \dots, q_\rho\}$. For example, $\Omega = q_1 \cap \dots \cap q_\rho$ means matching records' attribute A_j contains substring q_j for $1 \leq j \leq \rho$ at the same time.
- (ii) **Cloud server:** the cloud server is considered to be powerful in storage space and computational capability. The duties of the cloud server include the following: (i) efficiently store database DB and index I and (ii) process compound substring query token Q and respond a set of matching records $\mathcal{F} \subseteq DB$ to the data user.

2.2. Security Model. In our security model, the data user is considered as trusted, while the cloud server is assumed as honest-but-curious, which means that the cloud server will (i) honestly execute the query processing, return the query results without tampering it and (ii) curiously infer as much sensitive information as possible from the available data. The sensitive information could include the database DB, the index I , and the compound substring query token Q . The formal simulated-based definition for this security model is described in Section 3.4.

2.3. Design Goals. In this work, our design goal is to achieve an efficient and privacy-preserving SSE scheme supporting compound substring queries for the database. In particular, the following two requirements should be achieved.

- (i) **Privacy Preservation.** In the proposed scheme, all the data obtained by the cloud server, i.e., $\{DB, I, Q\}$, should be privacy-preserving during the outsourcing and query phases. Formally, the proposed scheme needs to satisfy security Definition 1 in Section 3.4.
- (ii) **Efficiency.** In order to achieve the above privacy requirement, additional computational costs will inevitably be incurred. Therefore, in this work, we also aim to reduce the query time to be linear with the length of the query token plus the size of matching results.

3. Preliminary

In this section, we give some preliminaries including position heap [12], symmetric key encryption scheme, fully homomorphic encryption, and the security definition of our

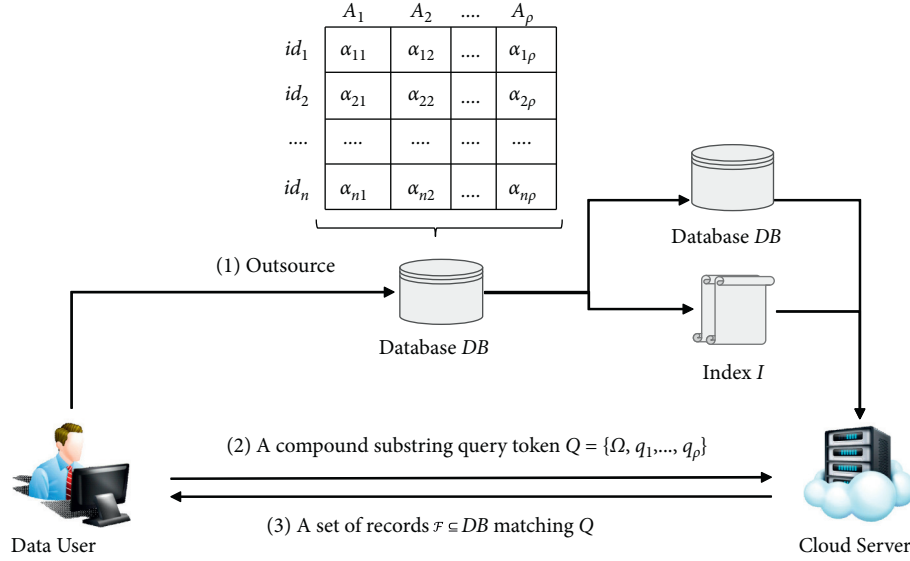


FIGURE 1: System model under consideration.

proposed scheme, which will serve as the basis of our proposed scheme.

3.1. The (Original) Position Heap Technique. Intuitively speaking, the (original) position heap $P(t)$ is a trie built from all the suffixes of t and can be used to achieve efficient substrings search for t . To construct the position heap $P(t)$ from a string $t = c_1c_2 \dots c_p$, a set of suffixes $t[i:p] = c_i \dots c_p$ ($i \in [p, \dots, 1]$) are chosen and inserted to the $P(t)$, which is initialized as a root node. To do this, for each suffix $t[i:p]$ ($i \in [p, \dots, 1]$), its longest prefix $t[i:j]$ ($i \leq j \leq p$) that is already represented by a path in $P(t)$ is found and a new leaf child is added to the last node of this path. The new leaf child is labeled with i and its edge is labeled with $t[j+1]$ (see Figure 2). Compared to other data structures to achieve substrings search, such as suffix tree [9] and suffix array [13], the position heap [12] can achieve high efficiency in both storage and query time.

In the following, we formally describe the PHBuild and PHSearch algorithms of the position heap, which will be used to build a position heap and search on it. Note that we consider each node in the position heap stores two types of data: edge and pos, which present the label of the node's edge and the label of the node, respectively.

3.1.1. PHBuild Algorithm. Given a string $t = c_1c_2 \dots c_p$, the PHBuild (i.e., Algorithm 1) visits the t from the right to left and inserts each position $i \in [p, p-1, \dots, 1]$ to the position heap $P(t)$. In particular, for each position i , the algorithm first finds the longest path from the root node of $P(t)$, where its *path label* is a prefix of $c_i \dots c_m$ (lines 4–7). Assume that the last node of this longest path is N . Then, the algorithm appends a new leaf child N' to the N , where $N' \cdot \text{edge} = c_{j+1}$ and $N' \cdot \text{pos} = i$ (lines 8–11). Figure 2 depicts an example to build such a position heap for a string $t = bbabbbaaba$. During inserting position $i = 1$, this algorithm first finds the

longest path (shown in bold) in $P(t)$ and appends a new leaf child N' to the last node of this path, where $N' \cdot \text{edge} = a$ and $N' \cdot \text{pos} = 1$.

3.1.2. PHSearch Algorithm. Given a substring s and a position heap $P(t)$, the PHSearch (i.e., Algorithm 2) is supposed to find all the positions in t that are occurrences of s . The time complexity of this algorithm is $O(|s|^2 + d_r)$, where $|s|$ is the length of the queried substring and d_r is the number of matching occurrences. The details are as follows:

- (i) The algorithm first finds the longest path from the root node of $P(t)$, where its *path label* denoted by s' is a prefix of s . We refer to this longest path as a search path in the rest of the article. Then, the algorithm lets L_1 be the set of positions stored in the intermediate nodes along the search path and L_2 be the set of positions stored in the descendants of the last node of the search path (lines 3–18). In particular, if $s' \neq s$, the position stored in the last node of the search path is included in L_1 . Otherwise, it is included in L_2 .
- (ii) After completing the previous step, elements in L_2 must be the matching positions, and elements in L_1 may or may not be the matching positions. Next, the algorithm reviews each position $i \in L_1$ in the string t to filter out unmatching positions and removes them from the L_1 . Finally, this algorithm returns $L_1 \cup L_2$ (lines 19–23).

Take an example with Figure 2. Given a substring $s = bb$, the PHSearch algorithm first finds the search path labeled with bb . In this way, L_1 and L_2 are equal to $\{9\}$ and $\{5, 1, 4\}$. Then, this algorithm reviews the string t and makes sure $i = 9 \in L_1$ is not an occurrence of s . Therefore, position 9 is removed from L_1 , and L_1 is an empty set now. Finally, this algorithm returns all the positions in $L_1 \cup L_2 = \{5, 1, 4\}$.

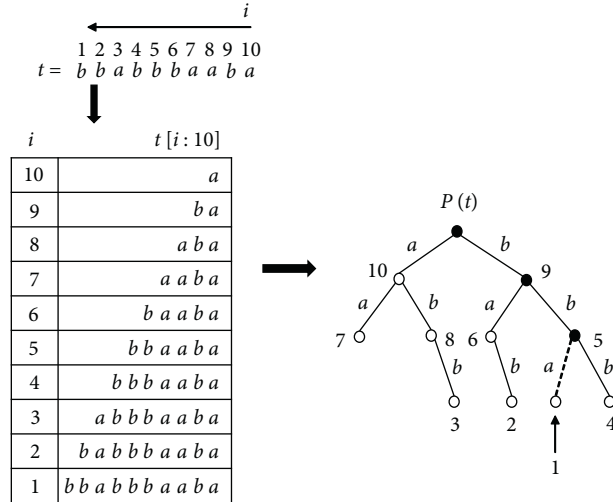


FIGURE 2: An example of building position heap $P(t)$ for string $t = bbabbaaba$. The solid edges in $P(t)$ reflect the insertion for suffix $t[1:10]$.

```

(1) initialize a root node  $R$  as the position heap  $P(t)$ , where  $R \cdot \text{edge} = \text{Null}$  and  $R \cdot \text{pos} = \text{Null}$ ;
(2) for each  $i$  in  $[p, p-1, \dots, 1]$  do
(3)    $N = R$ ;
(4)   for each  $j$  in  $[i, i+1, \dots, p]$  do
(5)     find the child  $N'$  of  $N$ , where  $N' \cdot \text{edge} = c_j$ ;
(6)     if  $N'$  does exist then
(7)        $N = N'$ 
(8)     else
(9)       insert a new child node  $N'$  for the  $N$ ;
(10)       $N' \cdot \text{edge} = c_j, N' \cdot \text{pos} = i$ ;
(11)      break;
(12)     end if
(13)   end for
(14) end for
(15) return  $P(t)$ ;

```

ALGORITHM 1: Build a position heap $P(t)$ for the string $t = c_1 c_2 \dots c_p$.

3.2. *Symmetric Key Encryption Scheme.* A symmetric key encryption scheme (SKE) consists of the following three polynomial-time algorithms (KeyGen, Enc, Dec).

- (i) $K \leftarrow \text{KeyGen}(1^\lambda)$: it takes a security parameter λ as input and outputs a secret key K
- (ii) $C \leftarrow \text{Enc}(K, M)$: it takes a key K and a message M as inputs and then outputs a ciphertext C
- (iii) $M \leftarrow \text{Dec}(K, C)$: it takes a key K and a ciphertext C as inputs and then outputs M

3.2.1. *Correctness.* For any message M in plaintext space, it holds that $\text{Dec}(K, \text{Enc}(K, M)) = M$.

3.2.2. *Security.* In this paper, we consider that the SKE is indistinguishable under a chosen-plaintext attack (IND-CPA) [14], which guarantees that the ciphertext does not

leak any information about the plaintext even an adversary can query an encryption oracle. We note that common private-key encryption schemes such as AES in counter mode satisfy this definition.

3.3. *(Leveled) Fully Homomorphic Encryption.* A leveled fully homomorphic encryption (FHE) scheme [15] consists of four polynomial-time algorithms (KeyGen, Enc, Dec, Eval). The details are described as follows:

- (i) $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda, L)$: it takes a security parameter λ and a maximum multiplicative depth L as inputs and outputs a public key pk and a secret key sk . We assume that a public key pk specifies both the plaintext space \mathcal{P} and the ciphertext space \mathcal{C} .
- (ii) $\bar{m} \leftarrow \text{Enc}(pk, m)$: given the public key pk and a plaintext m , it outputs a ciphertext \bar{m} . For simplicity, we omit the randomness used for encryption.


```

(1) initial empty sets  $L_1$  and  $L_2$ ;
(2) let  $N$  be the root node of the  $P(t)$ ;
(3) for each  $i$  in  $[1, 2, \dots, l]$  do
(4)   find the child  $N'$  of  $N$ , where  $N' \cdot \text{edge} = h_i$ ;
(5)   if  $N'$  does exist then
(6)     if  $i = l$  then
(7)        $L_2 \cdot \text{add}(N' \cdot \text{pos})$ ;
(8)       for each descendant  $X$  of  $N'$  do
(9)          $L_2 \cdot \text{add}(X \cdot \text{pos})$ ;
(10)      end for
(11)     else
(12)        $L_1 \cdot \text{add}(N' \cdot \text{pos})$ ;
(13)     end if
(14)      $N = N'$ ;
(15)   else
(16)     break;
(17)   end if
(18) end for
(19) for each  $i$  in  $L_1$  do
(20)   if  $c_i c_{i+1} \dots c_{i+l-1}$  is not equal to  $h_1 h_2 \dots h_l$  then
(21)      $L_1 \cdot \text{remove}(i)$ ;
(22)   end if
(23) end for
(24) return  $L_1 \cup L_2$ ;

```

ALGORITHM 2: Search substring s in a position heap $P(t)$, where $s = h_1 h_2 \dots h_l$ and $t = c_1 c_2 \dots c_p$.

(iii) $m \leftarrow \text{Dec}(\text{sk}, \overline{m})$: given the secret key sk and a ciphertext \overline{m} , it outputs a message m . $\overline{m}_\phi \leftarrow \text{Eval}(\text{pk}, \phi, \overline{m}_1, \dots, \overline{m}_l)$. It takes the public key pk , a function $\phi: P^l \rightarrow P$, and a set of ciphertexts $\{\overline{m}_1, \dots, \overline{m}_l\}$ as inputs and outputs a ciphertext \overline{m}_ϕ .

3.3.1. *Correctness.* For any $m_i \in \mathcal{P}$ ($1 \leq i \leq l$) and any function $\phi: \mathcal{P}^l \rightarrow \mathcal{P}$ which can be evaluated by a circuit with depth at most L , if $(\text{pk}, \text{sk}) \leftarrow \text{KenGen}(1^\lambda)$, $\overline{m}_i \leftarrow \text{Enc}(\text{pk}, m_i)$, and $\overline{m}_\phi \leftarrow \text{Eval}(\text{pk}, \phi, \overline{m}_1, \dots, \overline{m}_l)$, then it holds that $\text{Dec}(\text{sk}, \overline{m}_\phi) = \phi(m_1, \dots, m_l)$.

3.3.2. *Security.* In this work, we consider an FHE scheme is indistinguishable under a chosen-plaintext attack (IND-CPA), which is described in [15].

3.3.3. *Homomorphic Operations.* In general, an FHE scheme can directly support homomorphic bitwise addition (+) and multiplication (\cdot). Other advanced homomorphic operations can be realized by arithmetic circuits based on \cdot and $+$. In this paper, we consider three types of advanced homomorphic operations: bitwise AND, bitwise OR, and integer equality. In specific, if the FHE ciphertexts of two μ -bit integers $x = x_{\mu-1} \dots x_0$ and $y = y_{\mu-1} \dots y_0$ are $\overline{x} = \{\overline{x}_0, \dots, \overline{x}_{\mu-1}\}$ and $\overline{y} = \{\overline{y}_0, \dots, \overline{y}_{\mu-1}\}$, these arithmetic circuits are defined as follows:

- (i) Bitwise AND $\&$: $\overline{x} \& \overline{y} = \{s_0, \dots, s_{\mu-1}\}$, where $s_i = \overline{x}_i \cdot \overline{y}_i$ for $i \in [0, \mu - 1]$.
- (ii) Bitwise OR $|$: $\overline{x} | \overline{y} = \{s_0, \dots, s_{\mu-1}\}$, where $s_i = \overline{x}_i + \overline{y}_i + \overline{x}_i \cdot \overline{y}_i$ for $i \in [0, \mu - 1]$.
- (iii) Integer equality: $\text{EQ}(\overline{x}, \overline{y}) = \prod_{i=0}^{\mu-1} (1 + \overline{x}_i + \overline{y}_i)$. The output of $\text{EQ}(\overline{x}, \overline{y})$ is $\overline{1}$ in the case of $x = y$ and $\overline{0}$ otherwise, where $\overline{1}$ and $\overline{0}$ are FHE ciphertexts of 1-bit message 1 and 0, respectively.

3.4. *Security Definition of Our Proposed Scheme.* In this section, we follow the security definition in [5] to formalize the simulated-based security definition of our proposed scheme by using the following two experiments: $\text{Real}_{\mathcal{A}, \mathcal{E}}(\lambda)$ and $\text{Ideal}_{\mathcal{A}, \mathcal{S}}(\lambda)$. In the former, the adversary \mathcal{A} , who represents the cloud server, executes the proposed scheme with a challenger \mathcal{E} that represents the data user. In the latter, \mathcal{A} also executes the proposed scheme with a simulator \mathcal{S} that simulates the output of the challenger \mathcal{E} through the leakage of the proposed scheme. The leakage is parameterized by a leakage function collection $\mathcal{L} = (\mathcal{L}_O, \mathcal{L}_Q)$, which describes the information leaked to the adversary \mathcal{A} in the data outsourcing phase and query phase, respectively. If any polynomial adversary \mathcal{A} cannot distinguish the output information between the challenger \mathcal{E} and the simulator \mathcal{S} , then we can say there is no other information leaked to the adversary \mathcal{A} , i.e., the cloud server, except the information that can be inferred from the \mathcal{L} . More formally,

- (i) $\text{Real}_{\mathcal{A}, \mathcal{E}}(1^\lambda) = b \in \{0, 1\}$: given a database DB chosen by the adversary \mathcal{A} , the challenger \mathcal{E} outputs

encrypted index I by following the data outsourcing phase of the proposed scheme. Then, \mathcal{A} can adaptively send a polynomial number of compound substring query tokens to the \mathcal{C} , which outputs corresponding encrypted compound substring query tokens. Eventually, \mathcal{A} returns a bit b as the output of this experiment.

- (ii) $\text{Ideal}_{\mathcal{A},\mathcal{S}}(1^\lambda) = b \in \{0,1\}$: given the leakage function \mathcal{L}_O , the simulator outputs simulated encrypted index I' and simulated encrypted database DB' . Then, for each query token, the adversary \mathcal{A} sends its leakage function \mathcal{L}_Q to the simulator \mathcal{S} , which generates the corresponding simulated encrypted compound substring query token. Eventually, \mathcal{A} returns a bit b as the output of this experiment.

Definition 1. Our proposed scheme is \mathcal{L} -secure against adaptive attacks (i.e., \mathcal{L} -adaptively secure) if, for any probabilistic polynomial-time adversary \mathcal{A} , there exists an efficient simulator \mathcal{S} such that $|\text{Pr}[\text{Real}_{\mathcal{A},\mathcal{E}}(1^\lambda) \rightarrow 1] - \text{Pr}[\text{Ideal}_{\mathcal{A},\mathcal{S}}(1^\lambda) \rightarrow 1]| \leq \text{negl}(\lambda)$.

4. Our Proposed Scheme

In this section, we will present our SSE scheme. Before delving into the details, we first introduce our basic index structure, which is the basic building block of our proposed scheme.

4.1. Basic Index Structure. In order to process efficient compound substring queries, we build an index I_{A_j} for each attribute A_j ($1 \leq j \leq \rho$) in database DB. The index I_{A_j} is a modified position heap and can support two types of substring patterns: $*s*$ and $s_1 * s_2$. Next, we introduce algorithm Index Build and Index Search, which are used to build index I_{A_j} and search on it.

4.1.1. Index Build Algorithm. Given an attribute column $A_j = \{\alpha_{1j}, \alpha_{2j}, \dots, \alpha_{nj}\}$ of database DB, the IndexBuild algorithm outputs an index I_{A_j} as follows. It first transforms A_j to a string $t_{A_j} = \alpha_{1j}\#\alpha_{2j}\#\dots\#\alpha_{nj}$, where $\#$ denotes a character that does not appear in A_j . In the rest of this paper, we call t_{A_j} attribute string. Then, it follows PHBuild algorithm to insert all the positions in t_{A_j} , except positions of character $\#$, to a position heap $P(t_{A_j})$. Finally, this algorithm builds an index I_{A_j} from $P(t_{A_j})$ by replacing its nodes' position data (i.e., pos) to corresponding identifiers (i.e., id).

Figure 3 gives an example of building an index I_{A_1} for the first attribute column of database DB.

4.1.2. Index Search Algorithm. Given a substring s and an index I_{A_j} , the Index Search algorithm follows the PHSearch algorithm to search and outputs a set of identifiers.

4.2. Description of Our Proposed Scheme. In this subsection, we will describe our proposed scheme, which mainly consists of three phases: (i) system initialization; (ii) data outsourcing; (iii) compound substring query. To make the description simple, we first introduce a basic scheme which only supports substring query with $*s*$ pattern, and then extend it to support substring query with $s_1 * s_2$ pattern.

4.2.1. System Initialization. Given a security parameter λ , the data user first initializes a pseudorandom function (PRF) $H: \{0,1\}^\lambda \times \{0,1\}^* \rightarrow \{0,1\}^\lambda$, an IND-CPA secure SKE $\Pi = (\text{KeyGen}, \text{Enc}, \text{Dec})$, and an IND-CPA secure FHE $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval})$. Then, the data user generates keys $k_1 \xleftarrow{R} \{0,1\}^\lambda$, $k_2 = \Pi \cdot \text{KeyGen}(1^\lambda)$, and $(\text{pk}, \text{sk}) = \Sigma \cdot \text{KeyGen}(1^\lambda)$.

4.2.2. Data Outsourcing. Assume that the data user has a database DB with n records $\{f_1, f_2, \dots, f_n\}$, where each record f_i ($1 \leq i \leq n$) includes ρ string-type attributes $\{\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{i\rho}\}$. Then, the data user generates a secure index I and an encrypted database as the following steps:

Step 1: the data user first uses the Index Build algorithm to build I_{A_j} ($1 \leq j \leq \rho$) for each attribute A_j and then encrypts it as follows:

- (i) For each node N (except the root), the data user encrypts its $N \cdot \text{id}$ to $\overline{N \cdot \text{id}} = \Sigma \cdot \text{Enc}(\text{pk}, N \cdot \text{id})$
- (ii) For each node N (except the root), the data user concatenates all the edge labels, i.e., $N \cdot \text{edge}$, along the path from the root to this node, and calculates the PRF output of the concatenation through pseudorandom function $H(k_1, *)$

Consider the example in Figure 4, which is encrypted from the index I_{A_1} in Figure 3(d).

Step 2: the data user encrypts each record $f_j \in \text{DB}$ through $\Pi \cdot \text{Enc}(k_2, *)$ and sends these encrypted records to the cloud server with an encrypted index

$$I = \{I_{A_1}, I_{A_2}, \dots, I_{A_\rho}\}.$$

4.2.3. Compound Substring Query. Given a set of substrings $S = \{s_1, \dots, s_\rho\}$ and a compound formula Ω on S , where Ω consists of conjunctive expressions (i.e., \cap) and disjunctive expressions (i.e., \cup), the data user launches a compound substring query with the cloud server as follows:

Step 1: for each substring $s_j = c_1c_2 \dots c_l$ with $1 \leq j \leq \rho$, the data user calculates $q_j = (H(k_1, c_1), H(k_1, c_1c_2), \dots, H(k_1, c_1 \dots c_l))$ and sends a compound substring query token $Q = \{\Omega, q_1, \dots, q_\rho\}$ to the cloud server.

Step 2: for each $q_j \in Q$, the cloud server performs algorithm Index Search to search over I_{A_j} and outputs a

$$\text{set } R_j = \{\overline{\text{id}}_{j1}, \dots, \overline{\text{id}}_{j|R_j}\}.$$

Step 3: the cloud server generates a function $\phi: \{0,1\}^\rho \rightarrow \{0,1\}$ according to the compound

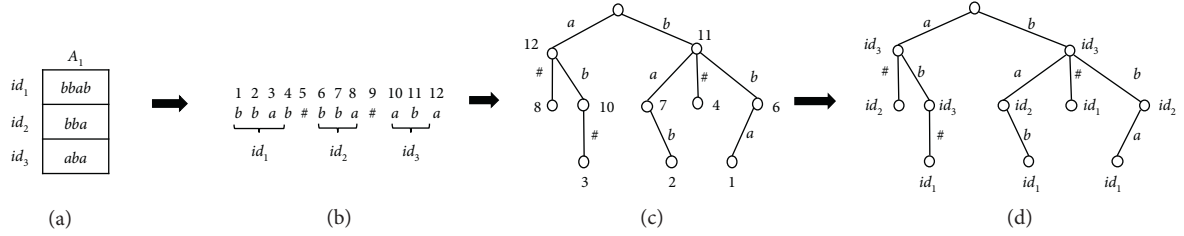


FIGURE 3: An example of building an index I_{A_1} . (a) A_1 is the first attribute column of database DB, where $\alpha_{11} = bbab$, $\alpha_{21} = bba$, and $\alpha_{31} = aba$. (b) To get attribute string t_{A_1} , concatenate all the attributes in A_1 with character #. (c) Build a position heap $P(t_{A_1})$ for t_{A_1} . (d) For each node N in $P(t_{A_1})$, replace its $N \cdot \text{pos}$ with the corresponding identifier, called $N \cdot \text{id}$.

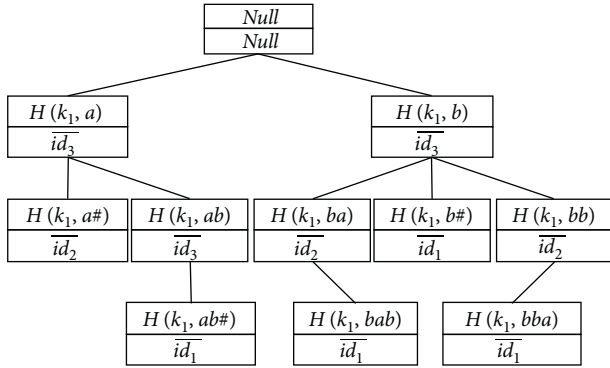


FIGURE 4: An example of the encrypted index I_{A_1} , which is encrypted from Figure 3(d).

formula Ω . In specific, all the \cap and \cup in Ω are replaced by bitwise AND $\&$ and bitwise OR $|$, respectively. For example, if compound formula $\Omega = q_1 \cup (q_2 \cap \dots \cap q_\rho)$, then $\phi = t_1 | (t_2 \& \dots \& t_\rho)$, where $t_j \in \{0, 1\}$ for $j \in [0, \rho]$.

Step 4: the cloud server performs Algorithm 3 to filter matching elements in $\{R_1, \dots, R_\rho\}$ according to function ϕ . In specific, assuming that R_{\min} is the minimum-size element in $\{R_1, \dots, R_\rho\}$, for each encrypted identifier \bar{id} in R_{\min} , this algorithm traverses all encrypted identifiers in $\{R_1, \dots, R_\rho\} \setminus \{R_{\min}\}$ to calculate an encrypted flag $\bar{t} \in \{\bar{0}, \bar{1}\}$, which is $\bar{1}$ if \bar{id} matches the compound formula Ω and $\bar{0}$ otherwise. Then, the cloud server inserts all the (\bar{id}, \bar{t}) to an empty set \mathcal{F} and returns it to the data user. Figure 5 depicts an example of this step where the compound formula $\Omega = q_1 \cap q_2 \cap q_3$. In this example, id_1 is the only identifier that matches the formula Ω . Note that, since an EQ algorithm consumes $\log(\mu)$ multiplicative depth and an AND/OR operation consumes 1 multiplicative depth, this step requires at most $\lceil \log(\mu) \rceil + \lceil \log(n\mu - 1) \rceil + \lceil \log(\rho - 1) \rceil$ multiplicative depth, where μ is the average attribute length of database and n is the number of records in the database.

Step 5: for each $(\bar{id}, \bar{t}) \in \mathcal{F}$, the data user calculates $t = \sum \cdot \text{Dec}(\text{sk}, \bar{t})$. If $t = 1$, then the data user calculates

$\text{id} = \sum \cdot \text{Dec}(\text{sk}, \bar{id})$ and requests a corresponding encrypted record from the cloud server.

4.2.4. Query with $s_1 * s_2$ Substring. We extend our scheme to support substring queries with $s_1 * s_2$ pattern. The extension is very simple, which only makes some small changes in the processes of the outsourcing phase and compound substring query phase. In specific, when the data user builds index I_{A_j} for the attribute column $A_j = \{\alpha_{1j}, \alpha_{2j}, \dots, \alpha_{nj}\}$, the attribute string $t_{A_j} = \alpha_{1j}\#\alpha_{2j}\#\dots\#\alpha_{nj}$ is replaced by $t_{A_j} = \alpha_{1j}_-\alpha_{1j}\#\alpha_{2j}_-\alpha_{2j}\#\dots\#\alpha_{nj}_-\alpha_{nj}$. In other words, each attribute in A_j is copied once and concatenated to its replica with character $_$, and every two adjacent attributes in A_j are concatenated with character #, where $_$ and # denote two separate characters that do not appear in A_j . In this way, the substring query with $s_1 * s_2$ pattern can be transferred to $*s * _$ pattern where $s = s_2_s_1$. Meanwhile, this method leads to double the storage cost of index I_{A_j} since the size of the attribute string t_{A_j} is twice longer than before.

5. Security Analysis

In this section, we prove the security of our proposed scheme based on the security definition described in Section 3.4.

5.1. Leakage Function Collection. We first define the leakage function collection $\mathcal{L} = \{\mathcal{L}_O, \mathcal{L}_Q\}$ of our proposed scheme.

(i) Outsourcing Phase: given the index

$I = \{I_{A_1}, I_{A_2}, \dots, I_{A_\rho}\}$ and the encrypted database DB, the leakage \mathcal{L}_O consists of the following information:

- (a) n : the number of records in DB
- (b) $|f_i|$ ($1 \leq i \leq n$): the size of record $f_i \in \text{DB}$
- (c) ρ : the number of attributes in DB
- (d) $|I_{A_j}|$ ($1 \leq j \leq \rho$): the number of nodes in I_{A_j}
- (e) Γ_j ($1 \leq j \leq \rho$): the structural dependencies between nodes in I_{A_j}

(ii) Query Phase: given the index $I = \{I_{A_1}, I_{A_2}, \dots, I_{A_\rho}\}$ and a compound substring query token Q , the leakage \mathcal{L}_Q consists of the following information:

```

(1) let  $R_{\min}$  be the minimum-size element in  $\{R_1, \dots, R_\rho\}$ .
(2) let  $\mathcal{F}$  be an empty set.
(3) for each  $\bar{id} \in R_{\min}$  do
(4)    $\bar{t}_{\min} \leftarrow \bar{1}$ 
(5)   for each  $j \in \{1, \dots, \rho\} \setminus \{\min\}$  do
(6)      $\bar{t}_j \leftarrow \bar{0}$ 
(7)     for each  $\bar{id}' \in R_j$  do
(8)        $t'_j \leftarrow EQ(\bar{id}, \bar{id}')$ 
(9)        $\bar{t}_j = \bar{t}_j | t'_j$ 
(10)    end for
(11)  end for
(12)   $\bar{t} \leftarrow \sum \cdot \text{Eval}(\text{pk}, \phi, \bar{t}_1, \dots, \bar{t}_\rho)$ 
(13)  insert  $(\bar{id}, \bar{t})$  to  $\mathcal{F}$ ;
(14) end for
(15) return  $\mathcal{F}$ 

```

ALGORITHM 3: Filter elements in $\{R_1, \dots, R_\rho\}$ according to function ϕ .

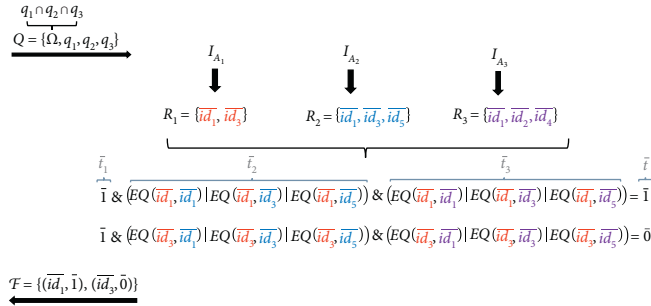


FIGURE 5: An example of Algorithm 3, where the number of attributes $\rho = 3$ and the compound formula $\Omega = q_1 \cap q_2 \cap q_3$.

- (a) path pattern: search paths in I_{A_j} ($1 \leq j \leq \rho$) corresponding to the query token Q
- (b) Ω : compound formula in Q

5.2. Security Proof. Now, we prove the security of our proposed scheme based on the leakage function collection $\mathcal{L} = \{\mathcal{L}_O, \mathcal{L}_Q\}$. Intuitively, we first define a simulator \mathcal{S} based on the leakage function collection \mathcal{L} and then analyze the indistinguishability between the output of the \mathcal{S} in the ideal world and the challenger \mathcal{C} (i.e., the data user) in the real world. Finally, we conclude that our proposed scheme does not reveal any information beyond the leakage function collection \mathcal{L} to the server. The details are as follows.

Theorem 1. *Let H be a pseudorandom function (PRF), let Π be an IND-CPA secure symmetric key encryption scheme (SKE), and let \sum be an IND-CPA secure fully homomorphic encryption (FHE). Then, our proposed scheme is \mathcal{L} -adaptively secure.*

Proof. Based on the leakage function collection \mathcal{L} , we can build a simulator \mathcal{S} as follows:

- (i) Data outsourcing: given the leakage function $\mathcal{L}_O = \{n, |f_i| (1 \leq i \leq n), \rho, |I_{A_j}| (1 \leq j \leq \rho), \Gamma_j (1 \leq j \leq \rho)\}$, the

simulator \mathcal{S} is supposed to generate a simulated index $I' = \{I_{A_1}', I_{A_2}', \dots, I_{A_\rho}'\}$ and a simulated encrypted database DB' . To build $I_{A_j}' \in I'$, the simulator \mathcal{S} first generates $|I_{A_j}'|$ empty nodes and constructs these nodes to a tree (i.e., I_{A_j}') based on Γ_j , which means that I_{A_j}' has the same tree structure as I_{A_j} . Then, for each node N in I_{A_j}' , the simulator \mathcal{S} randomly chooses $N \cdot \text{edge} \in \{0, 1\}^\lambda$ and $N \cdot \text{id} \in \{0, 1\}^d$. Since the outputs of H and $\sum \cdot \text{Enc}$ are pseudorandom, the adversary \mathcal{A} cannot distinguish between I' and I . To build DB' , the simulator \mathcal{S} chooses a random value $r_i \leftarrow \{0, 1\}^{|f_i|}$ for each record $f_i \in DB$ and lets $DB' = \{r_1, \dots, r_n\}$. Since Π is an IND-CPA secure SKE, the adversary \mathcal{A} cannot distinguish between DB' and DB .

- (2) Compound substring query: given the leakage function \mathcal{L}_Q for a compound substring query token $Q = \{\Omega, q_1, \dots, q_\rho\}$, the simulator \mathcal{S} is supposed to generate a simulated encrypted compound substring query token $Q' = \{\Omega, q'_1, \dots, q'_\rho\}$. Note that, at this moment, the simulator \mathcal{S} has not only \mathcal{L}_Q but also \mathcal{L}_O and I' from the data outsourcing phase.

Therefore, for each $I_{A_j}' \in I'$, the simulator \mathcal{S} can follow the corresponding path pattern in \mathcal{L}_Q to find its search path and output all the $N \cdot$ edge stored in the nodes along the search path as Q' . Since H is a pseudorandom function, the adversary \mathcal{A} cannot distinguish between Q' and Q .

In summary, as the adversary \mathcal{A} cannot distinguish between the outputs from the simulator \mathcal{S} in the ideal world and the challenger \mathcal{C} in the real world, we can conclude that our proposed scheme is \mathcal{L} -adaptively secure. \square

6. Performance Evaluation

In this section, we evaluate the performance of our proposed scheme from both theoretical and experimental perspectives. To the best of our knowledge, we are the first to discuss the compound substring query. Although some existing works [9–11] focus on the substring query, they cannot directly support the compound substring query. Therefore, a fair comparison is quite difficult and we just evaluate our proposed scheme in this section.

6.1. Theoretical Analysis. First, we theoretically analyze the query computational cost and storage overhead of our proposed scheme.

For the query computational cost, we analyze the server and user separately. In specific, the query computational cost of the server comes from two steps: search over index I to get a set of collections $\{R_1, \dots, R_\rho\}$ and filter matching elements in $\{R_1, \dots, R_\rho\}$. The former consumes at most $\sum_{j=1}^\rho |s_j|$ integer comparison operations and the latter consumes at most $\rho \cdot |R_{\min}| \cdot |R_{\max}|$ FHE multiplication operations, where R_{\min} and R_{\max} are the minimum-size and maximum-size elements in $\{R_1, \dots, R_\rho\}$. Meanwhile, the query computational cost of the user also comes from two steps: generate query token Q and decrypt the returned FHE ciphertexts, which consumes $\sum_{j=1}^\rho |s_j|$ hash operations and $|R_{\min}|$ FHE decryption operations, respectively. Compared with the above operations, we can see that $\rho \cdot |R_{\min}| \cdot |R_{\max}|$ FHE multiplication operations on the server dominate the query computational cost.

For the storage overhead, most of the storage overhead in our proposed scheme comes from the encrypted identifiers in $I = \{I_{A_1}, I_{A_2}, \dots, I_{A_\rho}\}$, which consumes $\sum_{j=1}^\rho |I_{A_j}| \cdot \log(n)/\ell$ FHE ciphertexts, where $|I_{A_j}|$ is the number of nodes in I_{A_j} and ℓ is the decomposition parameter in FHE (see next subsection).

6.2. Experimental Analysis. Then, we experimentally analyze our proposed scheme. In specific, we implement our proposed scheme in C++ (our code is open source [16]) and conduct experiments on a 64-bit machine with an Intel(R) Core(TM) i5-4300M CPU at 2.6GHZ and 4GB RAM, running Ubuntu 18.4. Note that we implement the data user

and cloud server on the same machine, which means there is no network delay between them. The underlying database in our experiment was extracted from Kaggle [17]. It contains 31,087 movies and 153,584 movie tags. The average length of all these movie tags is about 10. We treat each movie as a record with corresponding tags as its attributes.

6.2.1. FHE Implementation. In order to reduce the storage overhead of FHE, we use the SIMD technique [18] in experiments. Before describing our packing method, we first review the underlying structure of FHE. Specifically, the plaintext space of FHE is $\mathcal{P} = \mathbb{F}_p[x]/\langle \Phi_m(x) \rangle$ for a positive integer $p \geq 2$ where $\Phi_m(x)$ is the m th cyclotomic polynomial. When the plaintext modulus p is prime and not divisible by m , the $\Phi_m(x)$ decomposes into ℓ irreducible factors $f_1(x), \dots, f_\ell(x)$ of degree d modulo p . This induces an isomorphism, via the Chinese Remainder Theorem, between the algebra of the plaintext space $\mathcal{P} = \mathbb{F}_p[x]/\langle \Phi_m(x) \rangle$ and the product of ℓ finite fields $\mathbb{L}_i \cong \mathbb{F}_{p^d}$ for $i \in \{1, \dots, \ell\}$,

$$\mathcal{P} \cong \frac{\mathbb{F}_p[x]}{\langle f_1(x) \rangle} \times \dots \times \frac{\mathbb{F}_p[x]}{\langle f_\ell(x) \rangle} \quad (1)$$

$$= \mathbb{L}_1 \times \dots \times \mathbb{L}_\ell \cong (\mathbb{F}_{p^d})^\ell.$$

With this decomposition, the plaintext of compatible FHE schemes can be regarded as a length ℓ vector $m = (m_1, \dots, m_\ell)$, where $m_i \in \mathbb{F}_{p^d}$ for $i \in \{1, \dots, \ell\}$. Addition and multiplication on ciphertext \overline{m} correspond to component-wise addition and multiplication over the m_i for $i \in \{1, \dots, \ell\}$.

Now we give our packing method. In our proposed scheme, each index I_{A_j} for $j \in [1, \rho]$ includes a set of identifiers $ID_j = \{\text{id}_1, \text{id}_2, \dots, \text{id}_{|I_{A_j}|}\}$, where each identifier id_k in it can be seen as a bit-array $\{\text{id}_{k,1}, \text{id}_{k,2}, \dots, \text{id}_{k, \log(n)}\}$. In the experiments, we pack each ID_j to $(|I_{A_j}|/\ell) \times \log(n)$ FHE ciphertexts, which means each $\text{id}_{k,l}$ for $k \in [1, |A_j|]$ and $l \in [1, \log(n)]$ is encoded to an element in \mathbb{F}_{2^d} and therefore an FHE ciphertext contains ℓ bits in ID_j .

6.2.2. HELib Parameters. We utilize the HELib library [19] to implement the FHE described above. For the parameters, we choose $p = 2$ and $m = 18631$, which lead to $\varphi(m) = 18000$, $d = 25$, and $\ell = 720$. Meanwhile, the recent version of the HELib library (commit c74ffab in [19]) uses a concept of ciphertext capacity instead of depth. The ciphertext capacity of a ciphertext is defined in [20] as $\log(Q/\eta)$, where Q is the current modulus, and η is the current noise bound. In practice, an EQ algorithm costs about 60-bit capacity and an AND/OR operation costs about 25-bit capacity when $p = 2$ and $m = 18631$. Since n in our experiments does not exceed 25000, ρ does not exceed 5, and the average attribute length $\mu = 10$, we set the initial ciphertext capacity to $60 + \lceil \log(2500 \cdot 10 - 1) \rceil \cdot 25 + \lceil \log(5 - 1) \rceil \cdot 25 = 485$, which leads to 84-bit security.

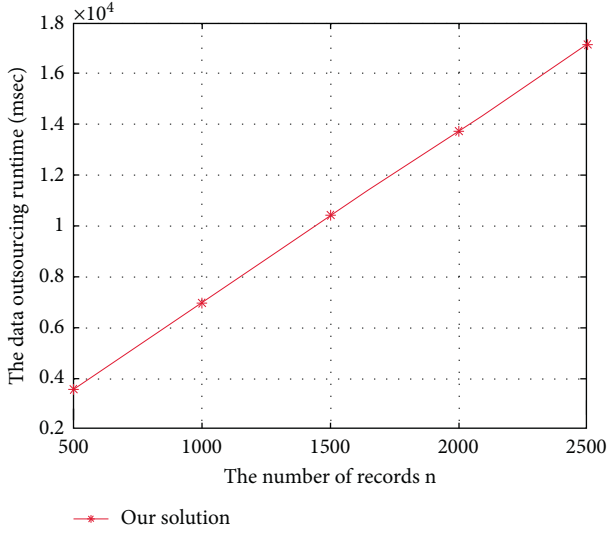


FIGURE 6: The data outsourcing runtime versus the number of records n , where the number of attributes ρ is fixed to 3.

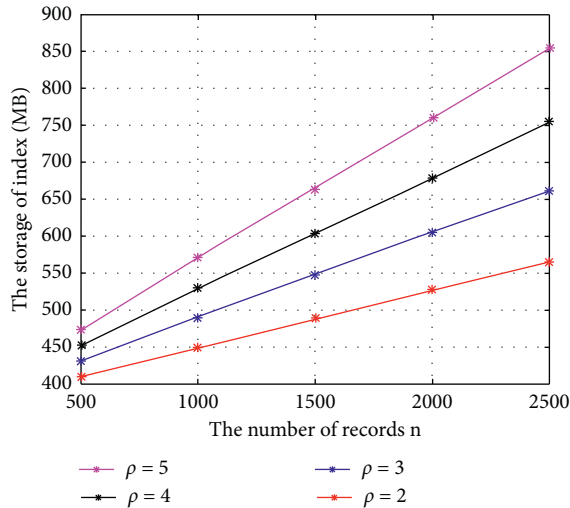


FIGURE 7: The storage overhead versus the number of records n , where the number of attributes ρ is chosen from 2 to 5.

In the following, we evaluate the computational cost and storage overhead of our proposed scheme in terms of two phases: data outsourcing and compound substring query.

6.2.3. Data Outsourcing. First, we consider the computational cost and storage overhead of the data outsourcing phase, which mainly comes from the building runtime and storage overhead of index $I = \{I_{A_1}, I_{A_2}, \dots, I_{A_\rho}\}$. Figures 6 and 7 plot the building runtime and storage overhead of the data outsourcing versus the number of records n and the number of attributes ρ . From these figures, we can see that both computational cost and storage overhead increase linearly with n and ρ .

6.2.4. Compound Substring Query. Next, we consider the computational cost of the compound substring query phase.

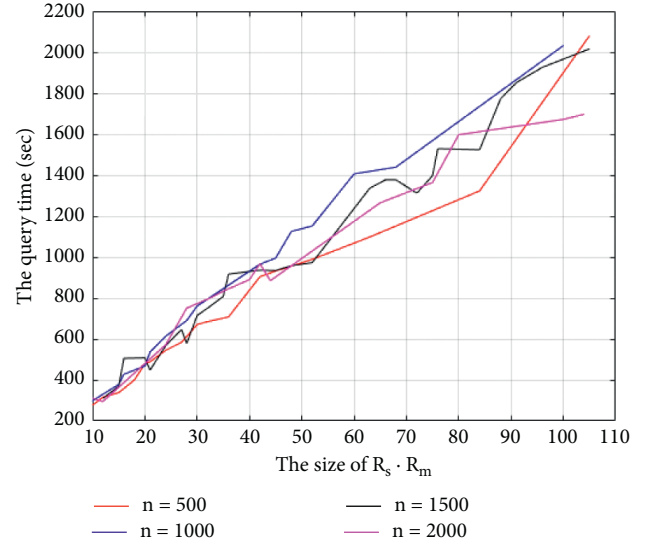


FIGURE 8: The query runtime versus the size of $R_{\min} \cdot R_{\max}$, where the number of records is chosen from 500 to 2000 and the number of attributes ρ is fixed to 3.

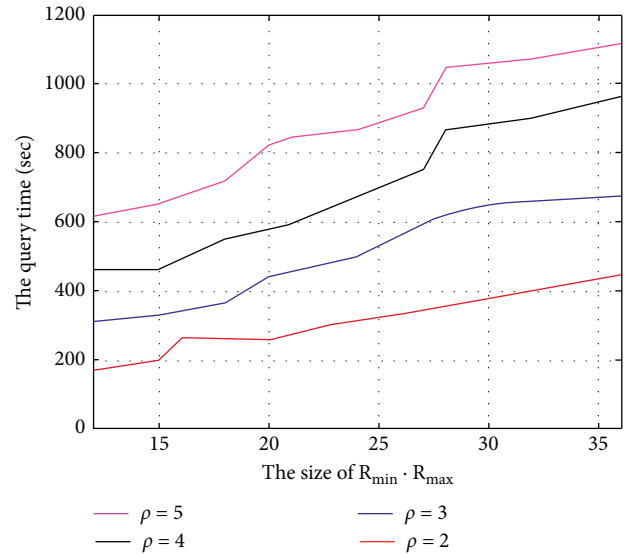


FIGURE 9: The query runtime versus the size of $R_{\min} \cdot R_{\max}$, where the number of attributes ρ is chosen from 2 to 5 and the number of records n is fixed to 500.

As mentioned in the last subsection, most of the query computational cost is from $\rho \cdot |R_{\min}| \cdot |R_{\max}|$ FHE multiplication operations on the server, where R_{\min} and R_{\max} are the minimum-size and maximum-size elements in $\{R_1, \dots, R_\rho\}$. As shown in Figures 8 and 9, the query time is indeed linear with $R_{\min} \cdot R_{\max}$ and ρ . Meanwhile, it is not affected by the number of records n .

7. Related Work

A searchable encryption scheme can be realized with optimal security via powerful cryptographic tools, such as fully homomorphic encryption (FHE) [21, 22] and Oblivious

Random Access Memory (ORAM) [23, 24]. However, these tools are extraordinarily impractical. Another set of works utilize property-preserving encryption (PPE) [25–28] to achieve searchable encryption, which encrypts messages in a way that inevitably leaks certain properties of the underlying message. For balancing the leakage and efficiency, many studies focus on searchable symmetric encryption (SSE). Song et al. [3] first used symmetric encryption to facilitate attribute queries over the encrypted data. Then, Curtmola et al. [5] gave a formal definition of SSE and proposed an efficient SSE scheme. Later, Kamara et al. [29] proposed the first dynamic SSE scheme, which uses a deletion array and a homomorphically encrypted pointer technique to securely update files. Unfortunately, due to the use of fully homomorphic encryption, the update efficiency is very low. In a more recent paper [7], Cash et al. described a simple dynamic inverted index based on [5], which utilizes the data unlinkability of the hash table to achieve secure insertion. Meanwhile, to prevent the file-injection attacks [30], many works [31–34] focused on forward security, which ensures that newly updated attributes cannot be related to previous queried results.

Nevertheless, these above works only can support the exact attribute query. If the queried attribute does not match a preset attribute, the query will fail. Fortunately, the fuzzy query can deal with this problem as it can tolerate minor typos and formatting inconsistencies. Li et al. [35] first proposed a fuzzy query scheme, which used an edit distance with a wildcard-based technique to construct fuzzy attribute sets. For instance, the set of *CAT* with 1 edit distance is $\{CAT, *CAT, *AT, C*AT, C*T, CA*T, CA*, CAT*\}$. Then, Kuzu et al. [36] used LSH (Local Sensitive Hash) and Bloom filter to construct a similarity query scheme. Since an honest-but-curious server may only return a fraction of the results, Wang et al. [37] proposed a verifiable fuzzy query scheme that not only supports fuzzy query service but also provides proof to verify whether the server returns all the queried results. However, these fuzzy query schemes only support single fuzzy attribute queries and address problems of minor typos and formatting inconsistency, which cannot be directly used to achieve substring queries.

In [9], Chase and Shen designed an SSE scheme based on the suffix tree to support substring queries. Although this scheme can be used to implement the substring query and allows for substring query in $O(|s| + d_s)$ time, its storage cost $O(m)$ has a big constant factor. The reason is that the suffix tree only stores position data in leaf nodes and does not utilize the space of inner nodes effectively. This makes the number of nodes in the suffix tree can be up to $2m$, where m is the size of the dataset. In order to reduce the storage cost as much as possible, Leontiadis and Li [13] leveraged Burrows-Wheeler Transform (BWT) to build an auxiliary data structure called a suffix array, which can achieve storage cost $O(m)$ with a lower constant factor. However, its query time is relatively large. Later, Mainardi et al. [11] optimize the query algorithm in [13] to achieve $O(|s| + d_s)$ at the cost of higher index space, i.e., $O(|\Sigma| \cdot m)$, where $|\Sigma|$ is the number of distinct characters in the dictionary. In addition to suffix tree and suffix array, there are some other auxiliary data structures that can be used to support substring queries. In

2018, Hahn et al. [38] designed an index based on k-grams. When a user needs to perform a substring query, the cloud performs a conjunctive keyword query for all the k-grams of the queried substring. In the same year, Moataz et al. [10] proposed a new substring query scheme based on the idea of letter orthogonalization, which allows testing of string membership by performing an efficient inner product. Although the above schemes can support substring queries, they can only solve the substring query problem for a single attribute, which cannot be used to achieve compound substring queries efficiently.

8. Conclusion

In this paper, we have proposed a novel efficient and privacy-preserving compound substring query scheme. Specifically, based on the position heap technique, we first designed a tree-based index to support a substring query on a single attribute and then applied PRF and FHE techniques to protect its privacy. In addition, based on the homomorphism of fully homomorphic encryption, we designed an algorithm to support compound substring queries on multiple attributes. Detailed security analysis and performance evaluation show that our proposed scheme is indeed privacy-preserving and efficient. In our future work, we will consider extending the proposed scheme to support wildcard queries.

Data Availability

Data will be available at the following link <https://www.kaggle.com/rounakbanik/the-movies-dataset>.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by NSERC Discovery Grants (Rgpin 04009) and National Natural Science Foundation of China (U1709217), and NSFC Grant (61871331).

References

- [1] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Efficient and privacy-preserving similarity range query over encrypted time series data," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2021.
- [2] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, "Towards practical and privacy-preserving multi-dimensional range query over cloud," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2021.
- [3] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pp. 44–55, Berkeley, CA, USA, May 2000.
- [4] Y. Zheng, R. Lu, J. Shao, F. Yin, and H. Zhu, "Achieving practical symmetric searchable encryption with search pattern

- privacy over cloud,” *IEEE Transactions on Services Computing*, p. 1, 2020.
- [5] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption,” in *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS*, pp. 79–88, Alexandria, VA, USA, October 2006.
 - [6] D. Cash, S. Jarecki, C. Jutla, and H. Krawczyk, M.-C. Roşu and M. Steiner, Highly-scalable searchable symmetric encryption with support for boolean queries,” in *Proceedings of the Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference*, pp. 353–373, Santa Barbara, CA, USA, August 2013.
 - [7] D. Cash, J. Jaeger, S. Jarecki et al., “Dynamic searchable encryption in very-large databases: data structures and implementation,” in *Proceedings of the 2014 Network and Distributed System Security Symposium*, pp. 23–26, San Diego, CA, USA, February 2014.
 - [8] E.-J. Goh, “Secure indexes,” 2003, <https://eprint.iacr.org/2003/216.pdf>.
 - [9] M. Chase and E. Shen, “Substring-searchable symmetric encryption,” *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 263–281, 2015.
 - [10] T. Moataz, I. Ray, I. Ray, A. Shikfa, F. Cuppens, and N. Cuppens, “Substring search over encrypted data,” *Journal of Computer Security*, vol. 26, no. 1, pp. 1–30, 2017.
 - [11] N. Mainardi, A. Barenghi, and G. Pelosi, “Privacy preserving substring search protocol with polylogarithmic communication cost,” in *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 297–312, New York, NY, United States, December 2019.
 - [12] A. Ehrenfeucht, R. M. McConnell, N. Osheim, and S.-W. Woo, “Position heaps: a simple and dynamic text indexing data structure,” *Journal of Discrete Algorithms*, vol. 9, no. 1, pp. 100–121, 2011.
 - [13] I. Leontiadis and M. Li, “Storage efficient substring searchable symmetric encryption,” in *Proceedings of the 6th International Workshop on Security in Cloud Computing*, pp. 3–13, Arizona, USA, May 2018.
 - [14] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, CRC Press, Boca Raton, FL, USA, 2014.
 - [15] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) fully homomorphic encryption without bootstrapping,” *ACM Transactions on Computation Theory*, vol. 6, no. 3, pp. 1–36, 2014.
 - [16] Y. Fan, “An implementation of our proposed scheme,” 2021, [Online]. Available: <https://github.com/YinFFF/Compound-substring-SSE>.
 - [17] 2020 kaggle. <https://www.kaggle.com/rounakbanik/the-movies-dataset>.
 - [18] N. P. Smart and F. Vercauteren, “Fully homomorphic simd operations,” *Designs, Codes and Cryptography*, vol. 71, no. 1, pp. 57–81, 2014.
 - [19] shah, “Helib library,” 2019, [Online]. Available: <https://github.com/homenc/HElib/tree/1.0.0-beta1-Aug2019>.
 - [20] S. Halevi and V. Shoup, “Bootstrapping for helib,” in *Proceedings of the Annual International conference on the theory and applications of cryptographic techniques*, pp. 641–670, Springer, Sofia, Bulgaria, April 2015.
 - [21] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pp. 169–178, Bethesda, MD, USA, May 2009.
 - [22] G.. Craig, “Computing arbitrary functions of encrypted data,” *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, 2010.
 - [23] R. Ostrovsky, “Efficient computation on oblivious rams,” in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pp. 514–523, Baltimore, MA, USA, May 1990.
 - [24] O. Goldreich and R. Ostrovsky, “Software protection and simulation on oblivious rams,” *Journal of the ACM*, vol. 43, no. 3, pp. 431–473, 1996.
 - [25] M. Bellare, A. Boldyreva, and A. O’Neill, “Deterministic and efficiently searchable encryption,” in *Proceedings of the 27th Annual International Cryptology Conference*, pp. 535–552, Santa Barbara, CA, USA, August 2007.
 - [26] A. Boldyreva, N. Chenette, Y. Lee, and A. O’Neill, “Order-preserving symmetric encryption,” in *Proceedings of the Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 224–241, Cologne, Germany, April 2009.
 - [27] A. Boldyreva, N. Chenette, and A. O’Neill, “Order-preserving encryption revisited: improved security analysis and alternative solutions,” in *Proceedings of the Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference*, pp. 578–595, Santa Barbara, CA, USA, August 2011.
 - [28] W. Yang, Y. Xu, Y. Nie, Y. Shen, and L. Huang, “TRQED: secure and fast tree-based private range queries over encrypted cloud,” in *Proceedings of the Database Systems for Advanced Applications - 23rd International Conference, DASFAA*, pp. 130–146, Gold Coast, QLD, Australia, May 2018.
 - [29] S. Kamara, C. Papamanthou, and T. Roeder, “Dynamic searchable symmetric encryption,” in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 965–976, ACM, Vienna, Austria, October 2012.
 - [30] Y. Zhang, J. Katz, and C. Papamanthou, “All your queries are belong to us: the power of file-injection attacks on searchable encryption,” in *Proceedings of the 25th USENIX Security Symposium, USENIX Security 16*, pp. 707–720, Austin, TX, USA, August 2016.
 - [31] R. Bost, “Σοφοϛ: forward secure searchable encryption,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds., pp. 1143–1154pp. 1143–, Vienna, Austria, October 2016.
 - [32] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W.-H. Kim, “Forward secure dynamic searchable symmetric encryption with efficient updates,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1449–1463, ACM, New York, NY, United States, October 2017.
 - [33] C. Zuo, S.-F. Sun, J. K. Liu, J. Shao, and J. Pieprzyk, “Dynamic searchable symmetric encryption schemes supporting range queries with forward (and backward) security,” in *Proceedings of the European Symposium on Research in Computer Security*, pp. 228–246, Springer, Berlin, Germany, August 2018.
 - [34] C. Zuo, S.-F. Sun, J. K. Liu, J. Shao, and J. Pieprzyk, “Dynamic searchable symmetric encryption with forward and stronger backward privacy,” in *Proceedings of the European Symposium on Research in Computer Security*, pp. 283–303, Springer, Verlag, Berlin, Heidelberg, September 2019.
 - [35] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, “Fuzzy keyword search over encrypted data in cloud computing,” in *Proceedings of the 2010 IEEE INFOCOM*, pp. 1–5, IEEE, San Diego, CA, USA, March 2010.

- [36] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pp. 1156–1167, IEEE, Arlington, VA, USA, April 2012.
- [37] J. Wang, H. Ma, Q. Tang, J. Li, and S. Ma, "Efficient verifiable fuzzy keyword search over encrypted data in cloud computing," *Computer Science and Information Systems*, vol. 10, no. 2, pp. 667–684, 2013.
- [38] F. Hahn, N. Loza, and F. Kerschbaum, "Practical and secure substring search," in *Proceedings of the 2018 International Conference on Management of Data*, pp. 163–176, New York, NY, United States, May 2018.

Research Article

Publicly Verifiable $M + 1$ -st-Price Auction Fit for IoT with Minimum Storage

Po-Chu Hsu ¹ and Atsuko Miyaji ^{1,2}

¹Graduate School of Engineering, Osaka University, Suita, Japan

²Japan Advanced Institute of Science and Technology, Nomi, Japan

Correspondence should be addressed to Atsuko Miyaji; miyaji@comm.eng.osaka-u.ac.jp

Received 20 May 2021; Revised 21 July 2021; Accepted 28 September 2021; Published 30 November 2021

Academic Editor: David Megias

Copyright © 2021 Po-Chu Hsu and Atsuko Miyaji. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In an $M + 1$ -st-price auction, all bidders submit their bids simultaneously, and the M highest bidders purchase M identical goods at the $M + 1$ st bidding price. Previous research is constructed based on trusted managers such as a trusted third party (TTP), trusted mix servers, and honest managers. All of the previous auctions are not fit for edge-assisted IoT since they need TTP. In this paper, we formalize a notion of commutative bi-homomorphic multiparty encryption and achieve no-TTP $M + 1$ -st auction based on blockchain with public verifiability. Our $M + 1$ st auction guarantees financial fairness, robustness, and correctness without TTP and is secure under a malicious model for the first time. Our $M + 1$ st auction can be executed over a distributed network and is thus fit for edge-assisted IoT. Furthermore, our formalized commutative bi-homomorphic multiparty encryption can be used in various applications for edge-assisted IoT, which needs to protect privacy and correctness.

1. Introduction

Since the introduction of Ethereum [1], a smart contract has become a robust decentralized execution environment for many applications. A variety of applications have been built on Ethereum. For example, cryptocurrency wallets, decentralized markets, and games are just a few examples. In this paper, we study secure auctions as an example of smart contracts.

Vickrey auction is also called second-price auction. In a second-price auction, the bidder who placed the highest bid only needs to pay the second-highest bidding price for the goods. Except for the identity of the highest bidder and the second-highest price, all other information is kept a secret. A 2nd-price auction is a type of sealed-bid auction. In a sealed-bid auction, bidders submit written bids without knowing other bidders' bids.

$M + 1$ -st-price auction is an extension of second-price auction. In $M + 1$ -st-price auction, there are M identical goods. The bidders who placed top M bids only need to pay the $M + 1$ st bidding price for the goods. Ideally, all

information except the $M + 1$ st price and the bidders who placed the M highest bids are kept as a secret. Figure 1 shows an example of $M + 1$ st auction for $M = 2$ goods. The top two bidders B_4 and B_2 can buy the goods by \$500. Except the boxed information, all other information is kept as a secret. Table 1 shows the public and secret information in a $M + 1$ -st-price auction.

There are many applications using IoT devices such as smart agriculture [2], smart grid [3], and raw materials [4], all of which will use price decision procedures. In the price decision procedures, the second-price auction is said to be a method that reflects market prices and is used in various situations. The $M + 1$ st auction is the second-price auction that deals with M products. This is why $M + 1$ st auctions are exactly one of the important applications since they can use M products at once.

To achieve a secure $M + 1$ st auction, there are two main issues: correctness and public verifiability. Usually, correctness and public verifiability are realized by using trusted third party (TTP) as an auctioneer and mix and match [5]. Abe and Suzuki [6] introduced one of the first auction protocols based

\$200 \$300 \$500 \$800 \$1000
 B_3 B_5 B_1 B_4 B_2

FIGURE 1: All but boxed information should be secret (example of $M + 1$ st auction for $M = 2$).

TABLE 1: Public and secret information in $M + 1$ st-price auction.

	Other bidders	Top $M + 1$ st bidder	Top M winning bidders
Identity	<i>Secret</i>	<i>Secret</i>	Public
Price	<i>Secret</i>	Public	<i>Secret</i>

on homomorphic encryption. The bidding price is a bounded range. Bidders can only choose one bid from this pricing list. Many related research studies [7, 8] have a similar design such that the time or communication complexity is related to the length of the bidding price list P . To solve this problem, Mitsunaga et al. [9] proposed a scheme that uses a binary format to represent bidding vectors. This improvement reduced the time complexity to $\log P$. In this research, instead of further improving complexity, we focus on removing the trusted parties, such as manager and mix servers. Based on Abe and Suzuki’s scheme [6], we removed TTP by letting bidders collaboratively act as managers and mix servers. This strategy can also be used in follow-up research studies [7–9]. In our preliminary version of [10], we construct a verifiable $M + 1$ st-price auction without manager. In the journal version, we implement the smart contract and reduce the computation cost by using elliptic curve cryptography.

Auctioneer in a sealed-bid auction is usually a trusted person since he can see the secret information. To construct a scheme that no party, such as manager, bidder, smart contract, and trusted third party (TTP), knows secret information, a typical method is to divide a function of manager into two independent parties as in [7]. In their scheme, no single managers or bidders know this secret information. However, this scheme is based on the p th root problem, which causes high time complexity and requires massive memory usage by public parameters and bid values.

Mix and match [5] is widely used in auction protocols such as bit-slice-based auction [9, 11, 12] as well as [6] to verify if each encrypted bit is either $E(0)$ or $E(1)$. Mix and match can be divided into mix phase and match phase. The mix phase is based on mix net [13, 14] to shuffle a set of encrypted messages. The match phase is usually based on zero-knowledge proofs to prove that the encrypted message is in the shuffled set. Mix net not only relied on TTP but also increased communication costs. In summary, previous schemes need TTP.

1.1. Our Contribution. We proposed a secure $M + 1$ st auction, which satisfies the following features:

- (1) *Multiparty encryption:* our multiparty encryption is asynchronous.

The order of decryption does not depend on the order of encryption, which makes the decryption

asynchronous for the decryptors. So, decryptors do not need to wait for each other.

- (2) *Publicly verifiable $M + 1$ st auction:* our $M + 1$ st auction does not use mix and match in bid verification phase and does not use TTP in the whole protocol.
 - (a) *No mix and match protocol in bid verification:* our protocol does not rely on mix and match unlike [6, 9, 11, 12]. Our new approach can also be used on bit-slice-based auction protocols [9, 11, 12] and all other auction protocols, which is based on encrypted bits.
 - (b) *No TTP:* in the $M + 1$ st-price decision phase, we modify mix and match by letting bidders act as mix servers, and thus no trusted mix server is used. Our $M + 1$ st-price auction is secure under a malicious model that guarantees financial fairness, robustness, and correctness without any other party such as manager and TTP. We remark that our scheme can be operated without TTP, and thus any bidder can start auction freely by using their IoT devices. Our decentralized auction exactly fits for IoT environment.

This paper is organized as follows. We introduce the related studies on smart contract auction protocols in Section 2 and explain cryptographic backgrounds in Section 3. We propose an efficient and secure $M + 1$ st-price auction protocol in Section 4 and compare our work with the related works in Section 6. Finally, we conclude our work in Section 7.

2. Related Work

This section reviews previous auction protocols which use a smart contract.

2.1. Overview of Auction. We define necessary features for auction protocol

- (i) Bid binding: bidders cannot change their bidding price after the bid submission phase is closed.
- (ii) Bid secrecy: all bidding prices except the $M + 1$ st-price should be kept secret.
- (iii) Bidder anonymity: except for the winning bidder, the identity of other bidders, including the bidder who placed the $M + 1$ st-price, is kept secret.
- (iv) Manager-based secrecy and anonymity: except those explained above, all secrets are also disclosed to the manager.
- (v) Posterior secrecy and anonymity: all secrets can be a secret even after the auction.
- (vi) Robustness: malicious behaviors in each phase can be found immediately. Timeouts are set to make sure the auction ends within a determined time.
- (vii) Public verifiability: all procedures can be publicly verified.

- (viii) Correctness: the auction has correctly proceeded as defined in the auction protocol.
- (ix) Financial fairness: malicious parties need to compensate other parties. Usually, all parties need to deposit some Ether in the smart contract as a stake.

2.2. Hawk. Hawk executes smart contracts while keeping secrets [15] when a bidder and a manager exchange data in the following way. (1) When the protocol starts, the bidders and the manager deposit some Ether to the smart contract as a stake. (2) Zero-knowledge proof (zk-SNARK) is attached with all messages sent by the manager and bidders. (3) Smart contract verifies the zero-knowledge proof and the time limit in each phase. In the Hawk protocol, since the bid amount of all bidders is leaked to the manager by design, it is necessary to assume that the manager does not leak any information inappropriately or collude with any party. In other words, the manager has to be a trusted third party. The Hawk protocol achieves bid binding, bid secrecy, bidder anonymity, manager-based secrecy and anonymity, posterior secrecy and anonymity, robustness, correctness, public verifiability, and financial fairness.

2.3. Verifiable Sealed-Bid Auction Protocol. Another verifiable sealed-bid auction protocol [8] was proposed by using a smart contract and Pedersen commitment scheme [16]. In this protocol, there are bidders and managers, just as in Hawk. The manager and a bidder pay a deposit to the smart contract, and the bidder simultaneously sends a bid commitment to the smart contract. Next, the bidder encrypts the value by the manager's public key and opens the commitment to the smart contract. The manager determines the winning bid and winner, sends the commitment of the winning bid to the smart contract, and proves to the smart contract that the winning bid is higher than any other bid. This scheme achieves bid binding, bid secrecy, manager-based secrecy, posterior secrecy, robustness, correctness, and financial fairness.

2.4. Other Related Works. Wu et al. proposed a smart contract-enabled collusion-resistant e-auction [17], which is well organized and provides many experimental data. However, bidder anonymity is revealed when commitments have been opened. Blockchain-based smart contract for bidding system [18] has the same problem as [17] although they used a blockchain called MinerGate. It also reveals secrets at the end of the auction. Galal also proposed a full privacy-preserving Vickrey auction on top of Ethereum [19], whose design is similar to Hawk. All bidders encrypt their bids by managers of public key. The manager uses Intel SGX to decrypt each bids and determine a winner. Thus, it only satisfies manager-based secrecy and anonymity.

3. Preliminaries

Definition 1 (DDH assumption). *Let t be a security parameter. A decisional Diffie-Hellman (DDH) parameter generator \mathcal{G} is a probabilistic polynomial time (PPT) algorithm that takes an input 1^k and outputs the description of a*

finite field \mathbb{F}_p and a basepoint $g \in \mathbb{F}_p$ with the prime order q . We say that \mathcal{G} satisfies the DDH assumption if $\epsilon = |p_1 - p_2|$ is negligible (in K) for all PPT algorithms A , where $p_1 = \Pr[(\mathbb{F}_p, g) \leftarrow \mathcal{G}(1^k); y_1 = g^{x_1}, y_2 = g^{x_2} \leftarrow \mathbb{F}_p; A(\mathbb{F}_p, g, y_1, y_2, g^{x_1 x_2}) = 0] = 0$ and $p_2 = \Pr[(\mathbb{F}_p, g) \leftarrow \mathcal{G}(1^k); y_1 = g^{x_1}, y_2 = g^{x_2}, z \leftarrow \mathbb{F}_p; A(\mathbb{F}_p, g, y_1, y_2, z) = 0]$.

Definition 2 (ElGamal encryption [20]). Let p and q be large primes. Let $\langle g \rangle$ denote a prime subgroup of \mathbb{Z}_p^* generated by g whose order is q . Given a message $m \in \mathbb{Z}_p^*$, we define ElGamal [20] encryption as $Enc_y(m) = (g^r, m \cdot y^r)$, where y is the public key and $r \in \mathbb{Z}_q^*$. Given a ciphertext $c = (g^r, m \cdot y^r)$, decryption is defined as $Dec_x(c) = m$, where x is the private key.

Theorem 1 (plaintext equivalence proof of ElGamal ciphertext). *Given an ElGamal ciphertext $(g^r, m y^r)$ and a plaintext m' , the encryptor can prove the value of $m = m'$ without revealing r . This type of proof is based on the proof of equality of two discrete logarithms. i.e., g^r and $m y^r / m'$ share the same discrete logarithm r .*

Theorem 2 (proof of knowledge of a discrete logarithm). *Given $g \in \mathbb{Z}_p^*$ and $x \in \mathbb{Z}_q^*$, the knowledge of g^{x^2} discrete logarithm x can be proved without revealing x .*

Theorem 3 (proof of equality of two discrete logarithms). *Given $g_1, g_2 \in \mathbb{Z}_p^*$ and $x_1, x_2 \in \mathbb{Z}_q^*$, the proof of $g_1^{x_1} = g_2^{x_2}$ has same discrete logarithm $x_1 = x_2$ that can be proved without revealing x_1 and x_2 . This type of proof is a variation of the proof of knowledge of a discrete logarithm (Theorem 2).*

Protocol 1 (mix and match). Mix and match [5] is a technique used to examine whether the decryption of a ciphertext $Dec(c)$ belongs to a set of plaintexts. It is based on mix net, which can perform verifiable secret shuffle. The mix server only knows their own permutation, and they do not know the plaintext. $MixMatch[c, S]$:

Input: ciphertext c and a set of plaintexts $S = \{m_1, m_2, \dots, m_n\}$.

Output: true if $Dec(c) \in S$. Otherwise, output false.

- (1) Construct n ciphertext by $C = \{c/E(m_1), \dots, c/E(m_n)\} = \{C_1, \dots, C_n\}$.
- (2) For each mix server
 - (a) Calculate $C' = \{C_1^{r_1}, \dots, C_n^{r_n}\}$ where r_i is a secret random value.
 - (b) Generate a random permutation pm .
 - (c) Shuffle these ciphertexts: $Shuffle[C', pm] = (C'', \pi)$.
 - (d) Send C'' to next mix server and publish π .
- (3) Use a publicly verifiable way to decrypt the output C'' of the last mix server.
- (4) If there exists one plaintext 1, then output true. Otherwise, output false.

price = [1 2 3 4 5 6 7 8]
V = [E (0) E (0) E (0) E (0) E (0) E (0) E (0) E (0)]

FIGURE 2: Example of bit-slice bidding vector.

price = [2 ⁰ 2 ¹ 2 ² 2 ³]
V = [E (0) E (1) E (1) E (0)]

FIGURE 3: Example of bit-slice bidding vector in binary format.

Protocol 2 (verifiable secret shuffle). Verifiable secret shuffle is a technique for mix and match, which is designed to use multiple mix servers (TTP) to secretly shuffle ciphertexts. $Shuffle[C, pm] = (C', \pi)$:

Input: n ciphertexts $C = \{Enc(m_1), \dots, Enc(m_n)\}$.
Output: shuffled ciphertexts
 $C' = \{Enc(m_{pm(1)}), \dots, Enc(m_{pm(n)})\}$ and the proof π ,
where $pm(\cdot)$ is a secret permutation.

Protocol 3 (bit-slice). Bit-slice is a common technique used in many auction protocols. Assume that the maximum bidding price is 8 dollars. If bidder wants to buy the goods by 6 dollars, the bidder can compose a bidding vector V by $E(0)$ and $E(1)$. Figure 2 gives an example of a bit-slice bidding vector. The bidding vector can also be presented by a binary format [9, 12] to reduce its size. Figure 3 gives an example of a binary format bidding vector.

Algorithm 1 (binary search). Binary search [21] is an algorithm used to search an ordered list; it can reduce the time complexity to $O(\log(N))$, where N is the number of elements; $BiSearch[A, cmp] = i$:

Input: given an ordered list A of N elements and a compare function cmp .

Output: the index j of the target element $A[j]$. If there is no element that satisfies the condition, output the largest i where $cmp(i) = -1$.

4. Our Scheme

Mix servers in mix and match perform verifiable secret shuffle. However, the mix servers can leak the secret permutation to the bidder or the manager. In an auction, let bidder act as mix server that can still be secure because each bidder can only know their permutation. As bidders are counterparties, they have no incentives to leak their secret permutation to others. The design of this protocol is simple, and no additional parties such as manager and TTP are needed.

A notion of commutative encryption is extended to multiparty encryption, which is fit for encryption schemes in smart contract protocol. During decryption, each secret key holder can send their decryption message to the smart contract concurrently. Without this, if the ciphertext is not decrypted before timeout, the responsibility of failure is hard

to define. The last honest decryptor might be punished simply because the previous decryptor takes too much time. Also, transaction settlement takes time. It can be time costly for decryptors to decrypt things one by one.

4.1. Multiparty Encryption. A concept of commutative encryption was introduced in [22]. However, to better fit the asynchronous nature of smart contract protocol, we reformalize it as multiparty encryption here. Without losing generality, we use $n = 2$ case to explain.

Definition 3 (multiparty encryption). Let the public and private key pairs of the public key cryptosystem of the party be (y_1, x_1) and (y_2, x_2) , respectively. The ciphertext of plaintext using the public key y_i is $Enc_{y_i}(m)$, and the decryption is $Dec_{y_i}(Enc_{y_i}(m))$. When public key cryptosystem satisfies the following, the public key cryptosystem is called a multiparty encryption cryptosystem.

- (i) *Full Encryption.* We define the encryption using the public key y_2 for the ciphertext encrypted with public key y_1 for a plaintext m through the following operation.

$$Enc_{y_1 \cdot y_2}(m) = Enc_{y_2}(Enc_{y_1}(m)). \quad (1)$$

- (ii) *Partial Decryption.* We define the decryption performed by each party for the ciphertext $Enc_{y_1 \cdot y_2}(m)$ encrypted with the public keys y_1, y_2 for the plaintext m through the following operation.

$$\begin{aligned} Dec_{y_1}(Enc_{y_1 \cdot y_2}(m)) &= Enc_{y_2}(m), \\ Dec_{y_2}(Enc_{y_1 \cdot y_2}(m)) &= Enc_{y_1}(m). \end{aligned} \quad (2)$$

- (iii) *Full Decryption.* We define the decryption performed by both parties for the ciphertext $Enc_{y_1 \cdot y_2}(m)$ encrypted with the public keys y_1, y_2 for the plaintext m through the following operation.

$$Dec_{y_1}(Dec_{y_2}(Enc_{y_1 \cdot y_2}(m))) = m. \quad (3)$$

Definition 4 (commutative). For a given encryption $Enc_{y_1 \cdot y_2}(m)$, if

$$\begin{aligned} Dec_{y_1}(Dec_{y_2}(Enc_{y_1 \cdot y_2}(m))) \\ = Dec_{y_2}(Dec_{y_1}(Enc_{y_1 \cdot y_2}(m))) = m, \end{aligned} \quad (4)$$

is satisfied, then the encryption is a commutative.

Definition 5 (bi-homomorphic). When a encryption satisfies the following properties, we say that the encryption is bi-homomorphic.

- (i) $Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_1}(\tilde{m}) = Enc_{y_1 \cdot y_2}(m \cdot \tilde{m})$
(ii) $Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_2}(\tilde{m}) = Enc_{y_1 \cdot y_2}(m \cdot \tilde{m})$
(iii) $Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_1 \cdot y_2}(\tilde{m}) = Enc_{y_1 \cdot y_2}(m \cdot \tilde{m})$
(iv) $(Enc_{y_1 \cdot y_2}(m))^\omega = Enc_{y_1 \cdot y_2}(m^\omega)$

The multiparty encryption based on the ElGamal encryption is defined as follows, which will be proved to be bi-homomorphism.

- (i) *Full Encryption*. The encryption using the public keys y_2 and y_1 is as follows.

$$Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c), \text{ where,} \quad (5)$$

$$u_1 = g^{r_1}, u_2 = g^{r_2} \text{ and } c = m \cdot y_1^{r_1} y_2^{r_2}.$$

- (ii) *Partial Decryption*. The partial decryption for a ciphertext $Enc_{y_1 \cdot y_2}(m)$ by each party y_1 or y_2 is as follows.

$$Dec_{y_1}(Enc_{y_1 \cdot y_2}(m)) = (u_2, c \cdot u_1^{-x_1}) = (u_2, m y_2^{r_2})$$

$$= Enc_{y_2}(m), \quad (6)$$

$$Dec_{y_2}(Enc_{y_1 \cdot y_2}(m)) = (u_1, c \cdot u_2^{-x_2}) = (u_1, m y_1^{r_1})$$

$$= Enc_{y_1}(m).$$

- (iii) *Full Decryption*. The decryption performed by both parties for the ElGamal ciphertext $Enc_{y_1 \cdot y_2}(m)$ encrypted with the public keys y_1 and y_2 for the plaintext m is defined as follows.

$$Dec_{y_1}(Dec_{y_2}(Enc_{y_1 \cdot y_2}(m))) = Dec_{y_1}(Enc_{y_1}(m)) = m. \quad (7)$$

Thus, the multiparty ElGamal encryption scheme satisfies partial decryption, full decryption, and commutative encryption. We also show that the multiparty ElGamal encryption satisfies the commutative encryption with bi-homomorphism as follows.

Theorem 4. *The multiparty ElGamal encryption satisfies the commutative encryption with bi-homomorphism.*

Proof. For two given ciphertexts $Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c)$ and $Enc_{y_1}(\tilde{m}) = (\tilde{u}_1, \tilde{c})$ with $u_1 = g^{r_1}, u_2 = g^{r_2}, \tilde{u}_1 = g^{\tilde{r}_1}, \tilde{c} = \tilde{m} \cdot y_1^{\tilde{r}_1}$, the multiplication of the ciphertexts is as follows.

$$Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_1}(\tilde{m}) = (u_1, u_2, c) \cdot (\tilde{u}_1, \tilde{c})$$

$$= (u_1 \tilde{u}_1, u_2, c \tilde{c}) = Enc_{y_1 \cdot y_2}(m \cdot \tilde{m}). \quad (8)$$

For two given ciphertexts $Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c)$ and $Enc_{y_2}(\tilde{m}) = (\tilde{u}_2, \tilde{c})$ with $u_1 = g^{r_1}, u_2 = g^{r_2}, \tilde{u}_2 = g^{\tilde{r}_2}, \tilde{c} = \tilde{m} \cdot y_2^{\tilde{r}_2}$, the multiplication of the ciphertexts is as follows.

$$Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_2}(\tilde{m}) = (u_1, u_2, c) \cdot (\tilde{u}_2, \tilde{c})$$

$$= (u_1, u_2 \tilde{u}_2, c \tilde{c}) = Enc_{y_1 \cdot y_2}(m \cdot \tilde{m}). \quad (9)$$

For two given ciphertexts $Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c)$ and $Enc_{y_1 \cdot y_2}(\tilde{m}) = (\tilde{u}_1, \tilde{u}_2, \tilde{c})$ with $u_1 = g^{r_1}, u_2 = g^{r_2}, \tilde{c} = \tilde{m} \cdot y_1^{\tilde{r}_1} y_2^{\tilde{r}_2}$, the multiplication of the ciphertexts is as follows.

$y_2^{\tilde{r}_2}, \tilde{u}_1 = g^{\tilde{r}_1}, \tilde{u}_2 = g^{\tilde{r}_2}, \tilde{c} = \tilde{m} \cdot y_1^{\tilde{r}_1} y_2^{\tilde{r}_2}$, the multiplication of the ciphertexts is as follows.

$$Enc_{y_1 \cdot y_2}(m) \cdot Enc_{y_1 \cdot y_2}(\tilde{m}) = (u_1, u_2, c) \cdot (\tilde{u}_1, \tilde{u}_2, \tilde{c})$$

$$= (u_1 \tilde{u}_1, u_2 \tilde{u}_2, c \tilde{c}) \quad (10)$$

$$= Enc_{y_1 \cdot y_2}(m \cdot \tilde{m}).$$

For a given ciphertext $Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c)$ with $u_1 = g^{r_1}, u_2 = g^{r_2}, c = m \cdot y_1^{r_1} y_2^{r_2}$, the exponentiation of the ciphertext is as follows.

$$(Enc_{y_1 \cdot y_2}(m))^{\omega} = (u_1^{\omega}, u_2^{\omega}, c^{\omega})$$

$$= (g^{r_1 \omega}, g^{r_2 \omega}, m^{\omega} \cdot y_1^{r_1 \omega} y_2^{r_2 \omega}) \quad (11)$$

$$= Enc_{y_1 \cdot y_2}(m^{\omega}).$$

By using Theorem 1, we can make the verifiable partial decryption as follows. \square

Theorem 5 (verifiable partial decryption). *A decryptor (party 2) can prove that a decrypted ciphertext $Enc_{y_1} = (u_1, c_1)$ is a partial decryption of $Enc_{y_1 \cdot y_2}(m) = (u_1, u_2, c)$ without revealing x_2 by showing a same discrete logarithm proof that $y_2 = g^{x_2}$ and $u_2^{-x_2} = (u_2^{-1})^{x_2}$ has same discrete log x_2 .*

$$SPK[(\alpha): c/c_1 \equiv y_2^{\alpha} \wedge u_2 \equiv g^{\alpha}] (m). \quad (12)$$

4.2. Our Auction Protocol. Our protocol consists of a seller, who sells M goods at a fixed set of prices $\{1, \dots, P\}$, and bidders, who bid a price for the goods. The top M bidders can buy the goods by the $M + 1$ st price. Remark that no other party except the seller of bidders is required in our protocol.

Our protocol consists of 7 phases as follows.

- (1) *Smart contract deployment*: the seller sets necessary parameters and initialization.
- (2) *Bidder initialization*: bidders who are interested in the auction submit their stake and join the auction.
- (3) *Submission of bids by bidders*: bidders decide their bid and submit it to the smart contract.
- (4) *Bid verification*: verify all bidders' bid.
- (5) *$M + 1$ st-price decision*: find out the $M + 1$ st bidding price.
- (6) *Winner decision*: find out the highest M bidders who are winners of the auction.
- (7) *Payment*: the winning bidders pay the seller the $M + 1$ st price to buy the goods.

The protocol is described in detail as follows.

4.2.1. Smart Contract Deployment. The seller decides the following parameter and deploys the smart contract (SC) to start the auction.

- (i) Cryptographic parameters: large prime p , a base-point g with prime order q , and an auction base $z \leftarrow \mathbb{Z}_p \setminus \{0, 1\}$.
- (ii) Seller initialization:
 - (1) Bidding price list $\{p_1, \dots, p_P\}$.
 - (2) Timeouts for each phase T_1, \dots, T_6 : if a bidder failed to submit valid messages to smart contract within the timeout, the bidder will be treated as a malicious bidder and be financially penalized.
 - (3) Stake requirement d : bidders are required to submit d amount of Ether as stake to join the auction.

4.2.2. *Phase 1: Bidder Initialization.* Bidders submit (y_i, π_i^1) to SC within time T_1 as follows. We assume that there are more than $M + 1$ bidders.

- (i) Each bidder $B_i, i \in \{1, \dots, B\}$ computes the following and submits it to SC:
 - (1) Compute a public key $y_i = g^{x_i}$ for the ElGamal encryption, where $x_i \leftarrow \mathbb{Z}_q$ is a randomly chosen secret key.
 - (2) Compute a proof π_i^1 of $y_i = g^{x_i}$ as a conventional way of a proof of knowledge of y_i 's discrete logarithm x_i described in Section 3 and d amount of Ether as stake.
- (ii) After this phase ends, an aggregated public key $Y = \prod_{i=1}^B y_i$ can be calculated by SC.

4.2.3. *Phase 2: Submission of Bids by Bidders.* All bidders submit (V_i, V'_i, π_i^2) to SC within time T_2 as follows: see Figure 4 for V_i and V'_i . Bidders B_2 and B_3 win the auction and pay $p_{j_{M+1}} = p_2$ for the goods.

- (i) $B_i, i \in \{1, \dots, B\}$, computes the below and submits it to SC:
 - (1) Bidding vector $V_i = \{V_{i1}, \dots, V_{iP}\}$.
 - (2) Shuffled bidding vector $V'_i = \{V_{i,pm(1)}, \dots, V_{i,pm(P)}\}$.
 - (3) A proof of secure shuffle π_i^2 .

$$V_{ij} = \begin{cases} Z^1, & \text{if } j = b_i, \\ Z^0, & \text{if } j \neq b_i. \end{cases} \quad (13)$$

A bidder B_i first chooses a bidding point $b_i \in \{1, \dots, P\}$ corresponding to bidding price p_{b_i} , constructs a bidding vector V_i , and shuffles V_i to V'_i , while making a proof of the secure shuffle:

$$(\pi_i^2, V'_i) = \text{shuffle}(V_i, pm_i), \quad (14)$$

where pm_i is a secret permutation generated by B_i . This V'_i is used for the verification of V_i . In phase 3, we can decrypt V'_i to verify V_i without leaking V_i .

- (i) SC calculates the following array $\{a_i\}$ for B_i and c for bidding price after receiving all (V_i, pm_i) (see Figure 4 for $\{a_i\}$ and c).

		P_1	P_2	P_3	P_4	P_5
$b_1 = 2,$	V_1	= [Ency (z^0), Ency (z^1), Ency (z^0), Ency (z^0), Ency (z^0)],				
$b_2 = 4,$	V_2	= [Ency (z^0), Ency (z^0), Ency (z^0), Ency (z^1), Ency (z^0)],				
$b_3 = 3,$	V_3	= [Ency (z^0), Ency (z^0), Ency (z^1), Ency (z^0), Ency (z^0)],				
(shuffled)	V_1'	= [Ency (z^0), Ency (z^0), Ency (z^0), Ency (z^1), Ency (z^0)],				
(shuffled)	V_2'	= [Ency (z^0), Ency (z^0), Ency (z^0), Ency (z^1), Ency (z^0)],				
(shuffled)	V_3'	= [Ency (z^0), Ency (z^0), Ency (z^0), Ency (z^0), Ency (z^1)],				
	a_1	= [Ency (z^1), Ency (z^1), Ency (z^0) , Ency (z^0), Ency (z^0)]				
	a_2	= [Ency (z^1), Ency (z^1), Ency (z^1) , Ency (z^1), Ency (z^0)] win				
	a_3	= [Ency (z^1), Ency (z^1), Ency (z^1) , Ency (z^0), Ency (z^0)] win				
	c	= [Ency (z^3), Ency (z^3), Ency (z^2), Ency (z^1), Ency (z^0)]				
	MixMatch	= [False, False, True, True, True]				
						j_{M+1}

FIGURE 4: Example of $M = 2$ with 3 bidders and 5 bidding prices.

- (1) Compute array $a_i = (a_{i1}, \dots, a_{iP})$ for each bidder B_i :

$$a_{ij} = \prod_{k=j}^P V_{ik} = Z \sum_{k=j}^P t_{ik} \quad (1 \leq j \leq P). \quad (15)$$

For any j , a value of a_{ij} is equal to Z^1 if $b_i \geq j$, where b_i is B_i 's bidding point. Otherwise, $a_{ij} = Z^0$. The time complexity is $O(P)$, by computing $a_{ij} = a_{i(j+1)} \cdot V_{ij}$.

- (2) Compute array $c = (c_1, \dots, c_P)$ for each bidding price p_1, \dots, p_P :

$$c_j = \prod_{i=1}^B a_{ij} = Z \sum_{i=1}^B \sum_{j=1}^P t_{ij} \quad (1 \leq j \leq P). \quad (16)$$

The value of c_j is the number of bidders whose bid b_i is larger than or equal to j .

4.2.4. *Phase 3: Bid Verification.* A bidder can only bid on one price. So, correct bidding vectors V_i consists of $P - 1$ times of Z^0 and one Z^1 . Since bidder B_i also submitted V'_i , a shuffled V_i , we can simply decrypt V'_i to verify if it consists of $P - 1$ times of Z^0 and one Z^1 within time T_3 . The decryption will not leak any secret since the shuffled bidding vector V'_i does not contain any secret.

- (i) SC verifies all bidder B_i 's bid V_i :

- (1) Decrypt the shuffled bidding vector V'_i :

$$\{m_{i1}, \dots, m_{iP}\} = \{\text{Dec}(V'_{i1}), \dots, \text{Dec}(V'_{iP})\}. \quad (17)$$

- (2) Verify if there is exactly $P - 1$ times of Z^0 and one Z^1 in $\{m_{i1}, \dots, m_{iP}\}$.

4.2.5. *Phase 4: $M + 1$ st-Price Decision.* By using mix and match, SC and all B_i determine the $M + 1$ st price within T_4 . In array $c = (c_1, \dots, c_P)$, the value of c_j is the number of bidders B_i whose bid b_i is larger than or equal to bidding point j . Thus, we can use mix and match to find out the bidding point j where $\text{Dec}(c_j) \notin \{z^0, z^1, \dots, z^M\}$, but $\text{Dec}(c_{j+1}) \in \{z^0, z^1, \dots, z^M\}$. This j is the $M + 1$ st price's

bidding point. Since the c array is a decrementing array, binary search can speed up the search. We use j_{M+1st} as a symbol to this j .

- (i) By all bidders' help, SC finds the bidding point j that $MixMatch[c_j, \{z^0, z^1, \dots, z^M\}] = False$ but $MixMatch[c_{j+1}, \{z^0, z^1, \dots, z^M\}] = True$. Assume cmp is the compare function of binary search (Algorithm 1). $BiSearch$ will return the j where $cmp(c_j) = -1$ but $cmp(c_{j+1}) = 1$. This j is the $M + 1st$ price.

$$j_{M+1st} = BiSearch[c, cmp],$$

$$cmp(c_j) = \begin{cases} 1, & \text{if } MixMatch[c_j, \{z^0, z^1, \dots, z^M\}] = True, \\ -1, & \text{otherwise.} \end{cases} \quad (18)$$

4.2.6. Phase 5: Winner Decision. In this phase, by decrypting all bidders' $a_{i,j_{M+1st}+1}$ within time T_5 , the winners can be easily found. In this phase, bid secrecy holds because the bid b_i can be any number between $j_{M+1st} + 1$ and P .

- (i) By all bidders' help, SC decrypts $a_{i,j_{M+1st}+1}$ for all $i \in \{1, \dots, B\}$.

$$\delta_i = DEC(a_{i,j_{M+1st}+1}) = z^{\sum_{k=j}^P t_{ik}}. \quad (19)$$

Bidder B_i with $\delta_i = z^1$ wins the auction.

4.2.7. Phase 6: Payment. The bidders who win the auction send $d_{P_{j_{M+1st}}}$ amount of Ether (in Wei) to the seller through SC.

- (i) All winning bidders B_i send $d_{P_{j_{M+1st}}}$ amount of Ether (in Wei) to SC.
(ii) SC sends $d_{P_{j_{M+1st}}}$ M amount of Ether (in Wei) to the seller and refunds the stakes to all bidders.

4.3. Features and Security. In this section, we discuss the features and security of our protocol.

- (i) *Bid binding*: according to our implementation, the functions in SC will not allow bidders to change their bidding point after the bid submission phase is closed.
(ii) *Bid secrecy*: the bidding vectors V_1, \dots, V_B are encrypted by all bidders' public keys $Y = y_1 \cdots y_B$. Without all bidders' collaboration, the bid is kept a secret. Even though V' is decrypted, it is a permutation of V . Thus, decrypting V' will not leak any secret. The secrecy of top M bids is protected in phase 5. If $a_{i,j_{M+1st}+1}$ is an encrypted z^1 , all bids from $a_{i,j_{M+1st}+1}$ to a_{iP} can contain z^1 .

Unless all bidders collude, the bid secrecy will be satisfied in the mix and match used in Phase 4. In the mix part, ciphertexts are re-randomized and

secretly shuffled by all bidders. Without all bidders colluding, the permutation of shuffled ciphertexts cannot be identified. In the match part, the shuffled ciphertexts can be decrypted by only all bidders' collaboration.

- (iii) *Bidder anonymity*: since c_1, \dots, c_P are products of ciphertexts generated by all bidders, i.e., $c_j = \prod_{i=1}^B a_{ij}$. The proof of $Dec(c_j) = z^M$ will not leak $a_{ij}, i = 1, \dots, B$. Thus, the identity of the $M + 1st$ bidder is still a secret.
(iv) *Posterior secrecy and anonymity*: the bidding points $b_i, i = 1, \dots, B$ and bidding vectors $V_i, i = 1, \dots, B$ are still secrets even after the auction. Thus, posterior secrecy and anonymity still hold.
(v) *Robustness*: since all messages sent to the smart contract are attached with a proof, malicious behaviors in each phase can be found immediately. The timeouts T_1, \dots, T_6 are also set for each phase to ensure that the auction ends within a determined time. Therefore, this protocol provides robustness even in malicious models.
(vi) *Public verifiability*: all messages sent to SC are attached with a publicly verifiable non-interactive proof, which can be verified by smart contract immediately. Therefore, the correctness of the protocol is publicly verifiable.

- (1) *Phase 1: Bidder Initialization.* A public key $y_i = g^{x_i}$ is submitted by each bidder B_i to SC, $i = 1, \dots, B$. The proof of knowledge of x_i is publicly verifiable. Thus, the correctness of the public key is publicly verifiable.
(2) *Phase 2: Submission of Bids by Bidders.* A secure shuffled bidding vector V'_i is submitted by each bidder B_i to SC, $i = 1, \dots, B$. The proof that V'_i is a secure shuffle of V_i is publicly verifiable.
(3) *Phase 3: Bid Verification.* Let $V'_i = (V'_{i1}, \dots, V'_{iP}) = ((u_{i1}, v_{i1}), \dots, (u_{iP}, v_{iP}))$. BP decryption messages $u_{ij}^{x_j}; i = 1, \dots, B; j = 1, \dots, P$ are submitted by each bidder B_i . The proof of the same discrete logarithm that $u_{ij}^{x_j}$ and public key $y_i = g^{x_i}$ have the same discrete logarithm x_i is publicly verifiable. If V'_i is valid, then V_i is a valid bidding vector. Thus, the verification on bidding vectors $V_i, i = 1, \dots, B$ is publicly verifiable.
(4) *Phase 4: $M + 1st$ -Bid Decision.* A verifiable mix and match [5] is performed in this phase. The validity of the $M + 1st$ bidding point j where $MixMatch[c_j, \{z^0, z^1, \dots, z^M\}] = False$ but $MixMatch[c_{j+1}, \{z^0, z^1, \dots, z^M\}] = True$ is publicly verifiable. Thus, the $M + 1st$ bidding price is publicly verifiable.
(5) *Phase 5: Winner Decision.* Let $a_{ij} = (u_{ij}, v_{ij})$. B decryption messages $u_{i,j_{M+1st}+1}^{x_i}; i = 1, \dots, B$ are submitted by each bidder B_i . Therefore, the

decryption on $a_{i,j_{M+1st}+1}$ is verifiable. The decision of winners is publicly verifiable.

(vii) *Correctness:*

- (1) *Phase 1: Bidder Initialization.* All public keys $y_i = g^{x_i}, i = 1, \dots, B$ are publicly verified. Thus, the correctness holds.
- (2) *Phase 2: Submission of Bids by Bidders.* All secure shuffled bidding vectors $V'_i, i = 1, \dots, B$ are publicly verified. V'_i is a secure shuffle of V_i . The correctness holds.
- (3) *Phase 3: Bid Verification.* A correct bidding vector V_i should contain $P - 1$ amount of Z^0 and one Z^1 . Since V'_i is a secure shuffle of V_i , V'_i is correct if and only if V'_i also contains $P - 1$ amount of Z^0 and one Z^1 . Thus, the correctness holds.
- (4) *Phase 4: $M + 1st$ -Bid Decision.* The bidding point j_{M+1st} is correct since $MixMatch[c_{j_{M+1st}}, \{z^0, z^1, \dots, z^M\}] = False$, but $MixMatch[c_{j_{M+1st}+1}, \{z^0, z^1, \dots, z^M\}] = True$. There are less than M bidders' bids that are larger than or equal to $j_{M+1st} + 1$, but there are more than M bidders' bids that are larger than or equal to j_{M+1st} . Thus, j_{M+1st} is the $M + 1st$ bidding point. The correctness holds.
- (5) *Phase 5: Winner Decision.* If $a_{i,j_{M+1st}+1} = Z^1$, then the bidder B_i 's bidding point b_i is larger than or equal to $j_{M+1st} + 1$. Thus, this bidder is a winning bidder. The correctness holds.

(viii) *Financial fairness:* all bidders deposit d amount Ether (in Wei) in the smart contract as a stake in phase 1. In the failure condition part, malicious bidders will compensate other bidders by their stake.

5. Implementation and Optimization

In this section, we introduce our implementation, optimization, and benchmarks. This protocol consists of smart contract (https://github.com/tonypottera24/m-1st_auction_dlp_sol) and web3 clients (https://github.com/tonypottera24/m-1st_auction_dlp_py). The smart contract is implemented by Solidity 0.7.x with experimental ABIEncoderV2. We also used the “solidity-BigNumber” library (<https://github.com/zcoinofficial/solidity-BigNumber>) for big number computation. In the client part, we use Python library web3.py (<https://github.com/ethereum/web3.py>) for better big number support. In terms of the simulation environment, we use Ganache (<https://www.trufflesuite.com/ganache>), a testnet built by Truffle, to execute and estimate the gas usage.

The computational costs of discrete logarithm problem (DLP) based algorithms are usually significantly affected by the key size. Figure 5 shows the gas usage by using 1024-, 2048-, and 3072-bit DLP and ECC P256. A common way to

solve this problem is to use elliptic curve cryptography (ECC). Our ECC contract used the “elliptic-curve-solidity” (<https://github.com/witnet/elliptic-curve-solidity>) library, which is well tested and provides common NIST series curves such as secp256r1 (P256). As an Ethereum virtual machine always uses 256-bit integers, the gas consumption should not have significant differences between P192, P224, and P256. The ECC P256 version can save 69% gas on average compared with 3072-bit DLP. Table 2 shows implementation results of AS [6] and our scheme. Without using full mix and match and applying some optimization (see Section 6), we can reduce 33% gas on a 3-bidder and 6-bidding price setting.

6. Comparison

In this section, we compared the performance of our scheme with those of the AS [6], OM [7], GY [8], and MMO [9], as shown in Table 3; compared with other schemes, we do not need a trusted manager. The only role in our scheme is bidder. This design matches the ultimate goal of smart contract protocol, decentralized and trustless. As one of our main contributions in the bid verification phase, we use verifiable shuffle, i.e., the mix part, instead of the whole mix and match protocol as other related works do.

Detailed comparisons of phases 1 and 3 are as follows:

- (1) *Phase 1: Bidder Initialization.* In previous research [6, 9, 11, 12], the bids are encrypted only by managers' public key. The bid secrecy, bidder anonymity, and many other properties all relied on the trusted manager. On the other hand, in our scheme, the ciphertexts are encrypted by all bidders' public keys. Without all bidders' collaboration, no one can break the bid secrecy and bidder anonymity.
- (2) *Phase 3: Bid Verification.* A valid bidding vector $V_i = (V_{i1}, \dots, V_{iP})$ should contain exactly $P - 1$ amounts of Z^0 and one Z^1 . In previous research [6, 9, 11, 12], bidding vector verification contains two parts: (1) $V_{i1} \in \{Z^0, Z^1\}$; (2) $\sum_{j=1}^P V_{ij} = Z^1$. The first part is accomplished by mix and match [5]. This requires T (trusted) mix servers to perform mix (secure shuffle [23–25]) and match (zk equality proof). The second part is accomplished by asking trusted manager to decrypt $\sum_{j=1}^P V_{ij}$.

In our scheme, we only use the match part (secure shuffle) of the mix and match protocol to prove these two parts simultaneously. By all bidders' collaboration, we decrypt V'_i (a publicly verifiable shuffle of V_i). SC can verify if there is exactly $P - 1$ amounts of Z^0 and one Z^1 in V'_i without leaking any secret.

Table 4 compares our protocol with previous protocols. In previous research, trusted managers need to use mix and match on M values to verify P ciphertexts in bidding vector V_i for all B bidders. Therefore, the time complexity of manager is $O(BPM)$ [6–8]. The time complexity and storage complexity on SC are proportional to the number of trusted manager T , i.e., $O(TBPM)$ overall. By removing the manager and letting bidders act as managers, the time complexity of our bidder is as

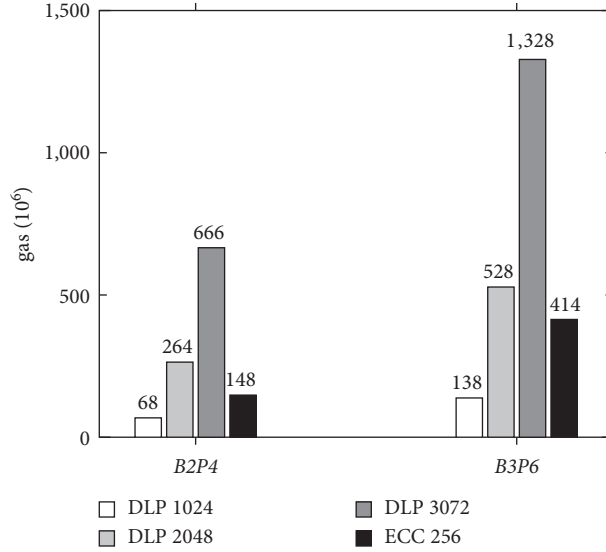


FIGURE 5: Gas usage of 1024-, 2048-, and 3072-bit DLP and 256-bit ECC (B2P4: 2 bidders and 4 bidding prices; B4P6: 4 bidders and 6 bidding prices).

TABLE 2: Comparison of AS [6] gas usage (10⁶) and our scheme (ECC 256: 3 bidders and 6 bidding prices).

	AS [6]		This scheme	
	Manager	Bidder	Bidder	
(1) Initialization	1	—	1	(-0%)
(2) Submission of bids by bidders	5	5	3	(-40%)
(3) Bid verification	520	—	350	(-33%)
(4) $M + 1$ st-price decision	79	—	52	(-34%)
(5) Winner decision	9	—	8	(-11%)
(6) Payment	4	4	2	(-50%)
Overall	618	9	416	(-33%)

TABLE 3: Comparison of related works and our scheme (TM: trusted manager).

	Trusted manager	Bid verification	Posterior secrecy and anonymity	Public verifiability	Robustness	Financial fairness
AS [6]	Yes	Mix and match	TM based	No	TM based	No
OM [7]	Yes	Mix and match	TM based	No	TM based	No
MMO [9]	Yes	Mix and match	TM based	Interactive	TM based	No
GY [8]	Yes	Commitment	TM based	No	TM based	Yes
Our scheme	No	Verifiable shuffle	Yes	Yes	Yes	Yes

TABLE 4: Comparison of the complexity with related works (B : the number of bidders, P : the number of bidding prices, M : number of goods, and T : the number of trusted managers and trusted mix servers).

	Time per manager	Time per bidder	Time SC	Storage SC
AS [6]	$O(BPM)$	$O(P)$	$O(TBPM)$	$O(TBPM)$
OM [7]	$O(BPM)$	$O(P)$	$O(TBPM)$	$O(TBPM)$
MMO [9]	$O(B \log PM)$	$O(\log PM)$	$O(TB \log PM)$	$O(TB \log PM)$
GY [8]	$O(BPM)$	$O(P)$	$O(TBPM)$	$O(TBPM)$
Our scheme	—	$O(BPM)$	$O(B^2PM)$	$O(BPM)$

good as the time complexity of the manager in previous research. On the other hand, as one of our contributions, we applied optimization to reduce the storage complexity from $O(B^2PM)$ to $O(BPM)$. In previous research, the multiparty

ElGamal ciphertext is in a format of $(g^{r_1}, \dots, g^{r_T}, g^m y_1^{r_1} \dots y_T^{r_T})$. However, it is meaningless to use different randomness r_1, \dots, r_T since the ciphertexts are created by the same bidder. Only $(g^r, g^m (y_1 \dots y_T)^r)$ is enough.

7. Conclusion

We propose an efficient and secure $M + 1$ -st-price auction protocol without a trusted manager under a malicious model. Our protocol satisfies anonymity, robustness, correctness, public verifiability, and financial fairness.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was partially supported by enPiT (Education Network for Practical Information Technologies) of MEXT and Innovation Platform for Society 5.0 of MEXT.

References

- [1] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.
- [2] J. Song, Q. Zhong, W. Wang, C. Su, Z. Tan, and Y. Liu, "FPDP: flexible privacy-preserving data publishing scheme for smart agriculture," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 17430–17438, 2020.
- [3] W. Wang, H. Huang, L. Zhang, and C. Su, "Secure and efficient mutual authentication protocol for smart grid under blockchain," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2681–2693, 2020.
- [4] W. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, and C. Su, "Blockchain-based reliable and efficient certificateless signature for IIOT devices," *IEEE Transactions on Industrial Informatics*, p. 1, 2021.
- [5] M. Jakobsson and A. Juels, "Mix and match: secure function evaluation via ciphertexts," in *Proceedings of the Advances in Cryptology - ASIACRYPT 2000. In: International Conference on the Theory and Application of Cryptology and Information Security*, pp. 162–177, Springer, Kyoto, Japan, December 2000.
- [6] M. Abe and K. Suzuki, "M + 1-st price auction using homomorphic encryption," in *Proceedings of the Key Cryptography. In: International Workshop on Public Key Cryptography*, pp. 115–124, Springer, Paris, France, February 2002.
- [7] K. Omote and A. Miyaji, "A second-price sealed-bid auction with verifiable discriminant of p 0-th root," in *Proceedings of the International Conference on Financial Cryptography*, pp. 57–71, Springer, Southampton, Bermuda, March 2002.
- [8] H. S. Galal and A. M. Youssef, "Verifiable sealed-bid auction on the ethereum blockchain," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 265–278, Springer, Nieuwpoort, Curaçao, March 2018.
- [9] T. Mistunaga, Y. Manabe, and T. Okamoto, "A secure $M + 1$ st price auction protocol based on bit slice circuits," *IEICE-Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E99.A, no. 8, pp. 1591–1599, 2016.
- [10] P. C. Hsu and A. Miyaji, "Verifiable $m + 1$ -st-price auction without manager," in *Proceedings of the IEEE Conference on Dependable and Secure Computing (DSC)*, pp. 1–8, IEEE, Aizuwakamatsu, Japan, January 2021.
- [11] K. Kurosawa and W. Ogata, "Bit-slice auction circuit," in *Proceedings of the Computer Security - ESORICS 2002. In: European Symposium on Research in Computer Security*, pp. 24–38, Springer, Zurich, Switzerland, October 2002.
- [12] T. Mistunaga, Y. Manabe, and T. Okamoto, "A secure $M + 1$ st price auction protocol based on bit slice circuits," in *Proceedings of the Advances in Information and Computer Security: International Workshop on Security*, pp. 51–64, Springer, Tokyo, Japan, November 2011.
- [13] M. Abe, "Universally verifiable mix-net with verification work independent of the number of mix-servers," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 437–447, Springer, Espoo, Finland, May 1998.
- [14] D. Wikström, "A universally composable mix-net," in *Proceedings of the Theory of Cryptography Conference*, pp. 317–335, Springer, Cambridge, MA, USA, February 2004.
- [15] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: the blockchain model of cryptography and privacy-preserving smart contracts," in *Proceedings of the IEEE symposium on security and privacy (SP)*, pp. 839–858, IEEE, San Jose, CA, USA, May 2016.
- [16] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pp. 28–36, Singapore, November 1999.
- [17] S. Wu, Y. Chen, Q. Wang, M. Li, C. Wang, and X. Luo, "Cream: a smart contract enabled collusion-resistant e-auction," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1687–1701, 2018.
- [18] Y. H. Chen, S. H. Chen, and I. C. Lin, "Blockchain based smart contract for bidding system," in *Proceedings of the IEEE International Conference on Applied System Invention (ICASI)*, pp. 208–211, IEEE, Taiwan, China, April 2018.
- [19] H. S. Galal and A. M. Youssef, "Trustee: full privacy preserving vickrey auction on top of ethereum," arXiv preprint, 2019, <https://arxiv.org/pdf/1905.06280>.
- [20] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.
- [21] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
- [22] K. Huang and R. Tso, "A commutative encryption scheme based on elgamal encryption," in *Proceedings of the 2012 International Conference on Information Security and Intelligent Control*, pp. 156–159, IEEE, Yunlin, Taiwan, August 2012.
- [23] C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in *Proceedings of the 8th ACM conference on Computer and Communications Security*, pp. 116–125, Philadelphia, PA, USA, November 2001.
- [24] J. Furukawa and K. Sako, "An efficient scheme for proving a shuffle," in *Proceedings of the Annual International Cryptology Conference*, pp. 368–387, Springer, Santa Barbara, California, USA, August 2001.
- [25] J. Groth and S. Lu, "A non-interactive shuffle with pairing based verifiability," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 51–67, Springer, Kuching, Malaysia, December 2007.

Research Article

A Commitment Scheme with Output Locality-3 Fit for the IoT Device

Hideaki Miyaji ¹, Yuntao Wang ², Akinori Kawachi ³ and Atsuko Miyaji ^{1,2}

¹Graduate School of Engineering, Osaka University, Osaka, Japan

²School of Information Science, JAIST, Nomi Shi, Japan

³Graduate School of Engineering, Mie University, Tsu, Japan

Correspondence should be addressed to Hideaki Miyaji; hideaki@cy2sec.comm.eng.osaka-u.ac.jp

Received 13 July 2021; Revised 14 September 2021; Accepted 11 October 2021; Published 29 November 2021

Academic Editor: Shifeng Sun

Copyright © 2021 Hideaki Miyaji et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Low output locality is a property of functions, in which every output bit depends on a small number of input bits. In IoT devices with only a fragile CPU, it is important for many IoT devices to cooperate to execute a single function. In such IoT's collaborative work, a feature of low output locality is very useful. This is why it is desirable to reconstruct cryptographic primitives with low output locality. However, until now, commitment with a constant low output locality has been constructed by using strong randomness extractors from a nonconstant-output-locality collision-resistant hash function. In this paper, we construct a commitment scheme with output locality-3 from a constant-output-locality collision-resistant hash function for the first time. We prove the computational hiding property of our commitment by the decisional (M, δ) -bSVP assumption and prove the computational binding property by the (M, δ) -bSVP assumption, respectively. Furthermore, we prove that the (M, δ) -bSVP assumption can be reduced to the decisional (M, δ) -bSVP assumption. We also give a parameter suggestion for our commitment scheme with the 128 bit security.

1. Introduction

The computational complexity of cryptographic primitives is a fundamental problem in the construction of highly efficient and secure protocols [1, 2]. In ITCS 2017, Applebaum et al. achieved pioneering results for low-complexity cryptographic constructions of fundamental primitives [3]. Their technique provides a general framework for converting relatively high-complexity cryptographic functions to low-complexity ones, including one-way and pseudorandom functions of low output localities. Furthermore, Applebaum et al. proposed constructions for collision-resistant hash functions of a constant output locality from computationally hard problems of lattices and multivariate polynomials [4]. Interestingly, one of their collision-resistant hash functions with low output locality relies on the hardness assumption of the lattice problem called (M, δ) -bSVP assumption.

The output locality is a natural complexity measure of computational efficiency for Boolean functions. It is known

that a Boolean function has output locality k if each output bit depends on a maximum of k input bits. It is obvious that low-output locality functions are implementable by low-depth circuits, implying high parallelizability. In extreme cases, if a function has a constant output locality, it can be decomposed into smaller functions computed using constant-depth circuits in parallel. In IoT devices with only a fragile CPU, it is difficult to execute a single rather large function. For this reason, it is important for many IoT devices to cooperate to execute a single function. In such IoT's collaborative work, the decomposition property into smaller functions is very useful. Low-depth cryptographic functions play crucial roles in certain protocols as well as IoT devices. For example, the bootstrapping method requires a low-depth decryption function as in lattice-based fully homomorphic public-key encryption [5].

There are several quantum-resistant cryptosystems, such as homogeneous cryptosystems and lattice cryptosystems. Output locality is a technology that encourages collaborative

work on cryptography. In particular, the construction of cryptographic primitives that are secure against quantum cryptography and satisfy output locality is significant for the widespread use of IoT devices. This paper aims to construct cryptographic primitives that have output locality and are secure against quantum cryptography.

On the contrary, a commitment scheme is a fundamental protocol and a key building block of basic cryptographic tasks such as zero-knowledge identification [6]. The scheme is conducted between two parties (i.e., a sender and a receiver) through commitment and decommitment phases. In the commitment phase, the sender converts a message into a commitment string and sends it to the receiver. Then, in the decommitment phase, the sender sends the decommitment string where the message is embedded, which allows the receiver to verify if the commitment string was indeed generated from the message or not. A commitment scheme's security is formalized based on two properties: the hiding property and the binding property. The hiding property guarantees that no receiver can receive partial information of messages before the decommitment phase. Simultaneously, the binding property ensures that no sender can choose one of more than two candidate messages by switching the decommitment strings in the decommitment phase.

The related work is as follows. Note that neither standard commitment schemes such as Pedersen [7] nor Halevi-Micali [8] have low output localities. To achieve a commitment scheme with low output locality, two approaches have been investigated until now. One is proposed in [3], where a transformation from collision-resistant hash functions to commitment schemes that preserve low output locality by using strong randomness extractors in order to obtain the hiding property is provided. Their commitment schemes using this general transformation satisfy the output locality of four.

Another one is to avoid using such strong randomness extractors and to construct a commitment scheme directly from a hash function [9, 10], which are our preliminary works. Remark that, in [9], it only proves that the output locality is smaller than the input length, and in [10], it is only claimed that the hiding property is based on the decisional (M, δ) -bSVP assumption, whereas no concrete proof was given nor the relation between the decisional (M, δ) -bSVP assumption and (M, δ) -bSVP assumption was shown. In other words, no secure commitment with output locality-3 has been proposed so far without using strong randomness extractors.

Our contributions are as follows. In this paper, we propose a commitment scheme with an output locality of three for the first time. Our construction does not use strong randomness extractors. We construct a commitment scheme directly from a collision-resistant hash function in NC^0 without using a strong randomness extractor. We prove its computational hiding property and its computational binding property by using the decisional (M, δ) -bSVP assumption and (M, δ) -bSVP assumption, respectively. Furthermore, we prove that the (M, δ) -bSVP assumption can be reduced to the decisional (M, δ) -bSVP assumption.

To construct such a commitment scheme, we focus on two primitives. The first is a commitment scheme from the

short integer solution (SIS) problem [11]. This scheme makes use of a lattice-based collision-resistant hash function of a "matrix-vector multiplication" form, i.e., $y = M \cdot x$ for a matrix $M \in \mathbb{Z}_q^{m \times n}$, and a vector $x \in \mathbb{Z}_2^n$. Our commitment also follows such a simple construction. As for the lattice-based collision-resistant hash function of low output locality, we use the next primitive of a function $f(x) = M \cdot \text{ex}(x)$, where ex is an expanding function that dilutes the Hamming weight on the input x to achieve collision-resistant properties from the intractability of bSVP [3]. Then, a randomized encoding technique [4] is applied to the function $f(x)$ to achieve low output locality. Here, a randomized encoding of $f(x)$ is a randomized mapping $\hat{f}(x, r)$ that generates an output distribution dependent only on $f(x)$.

Compared to previous works [10] in CANDAR 2020, this paper is the full version of the paper presented at CANDAR 2020. In our preliminary work [10], we have constructed a commitment scheme with output locality-3. However, it does not include any security consideration. In this article, we reconstruct a commitment scheme with output locality-3 based on the (M, δ) -bSVP assumption and decisional (M, δ) -bSVP assumption. We describe what we have achieved in this paper in the following:

- (i) Prove that the (M, δ) -bSVP assumption can be reduced to the decisional (M, δ) -bSVP assumption
- (ii) Prove that our commitment scheme satisfies the computational binding property based on the (M, δ) -bSVP assumption and satisfies the computational hiding property based on the decisional (M, δ) -bSVP assumption
- (iii) Compare our commitment scheme with other previous studies

Roadmap: the remainder of this paper is organized as follows. Section 2 summarizes the commitment scheme, the hash function, and the output locality. Section 3 describes the building blocks of our construction. Then, we present our commitment scheme in Section 4. In Section 5, we suggest the parameter of our commitment scheme. Finally, we conclude our work in Section 6.

2. Preliminaries

First, we summarize the notations used in this paper.

- (1) 1^k : security parameter
- (2) a : message string
- (3) r : random string
- (4) com : commitment string
- (5) dec : decommitment string
- (6) $\varepsilon(k)$: negligible function in k
- (7) ex : expand function
- (8) pp : public parameters
- (9) $S(1^k, \text{pp})$: probabilistic polynomial-time party
- (10) $R(\text{com}, \text{dec})$: probabilistic polynomial-time party which executes in the decommitment phase

- (11) $R_{\text{com}}(\text{pp}, \text{com})$: probabilistic polynomial-time party which executes in the commitment phase
- (12) c, d : output locality in the ex function
- (13) \perp : rejection symbol output by R for invalid inputs
- (14) $\text{Hw}(x)$: Hamming weight of x
- (15) $\Delta(x)$: the ratio of “1”s in x
- (16) H_{Mex} : the hash function we used in this paper
- (17) $\text{Comm}_{\text{Mex}}(S, R)$: our proposed commitment scheme
- (18) \mathbb{N} : set of natural numbers
- (19) $m < n \in \mathbb{N}$
- (20) $\mathcal{M}(1^n)$: matrix sampler that generates a uniformly random $m \times n$ matrix.
- (21) $H_2(p) = -p \log_2(p) - (1-p) \log_2(1-p)$ denotes the binary entropy function, where $p \in [0, 1]$
- (22) ε : a negligible function throughout this paper

Next, we define the commitment scheme, which is as follows [12].

Definition 1 (commitment scheme). A commitment scheme, $\text{Comm}(S, R)$, is a two-phase protocol between two probabilistic polynomial-time parties S and R , which are called the sender and receiver, respectively.

During the first phase (commitment phase), S commits string a to a pair of keys (com, dec) by executing $(\text{com}, \text{dec}) \leftarrow S(1^k, \text{pp})$. Then, S sends com (commitment string) to R .

During the second phase (decommitment phase), S sends the keys dec (decommitment string) with a to R . Then, R verifies whether the decommitment string is valid by executing $R(\text{com}, \text{dec})$. If invalid, $R(\text{com}, \text{dec})$ outputs a special string, \perp , meaning that R rejects the decommitment of S . Otherwise, $R(\text{com}, \text{dec})$ can efficiently compute the string a revealed by S and verifies whether a was indeed chosen by S during the first phase.

In the following discussion, we provide the security notions of the commitment scheme $\text{Comm}(S, R)$.

Definition 2 (computational binding property; see [8]). We state that $\text{Comm}(S, R)$ is computationally binding if it is computationally infeasible to generate a commitment string com and two decommitment strings, dec, dec' ($\text{dec} \neq \text{dec}'$), such that R will compute a message a from (com, dec) and a different message a' from (com, dec) . In detail, for every probabilistic polynomial-time adversary $S'(1^k, \text{pp})$, the following occurs:

$$\Pr \left[\begin{array}{l} R(\text{com}, \text{dec}) \neq \perp \\ (\text{com}, \text{dec}, \text{dec}') \leftarrow S'(1^k, \text{pp}): \\ \quad R(\text{com}, \text{dec}') \neq \perp \\ \quad R(\text{com}, \text{dec}) \\ \quad \neq R(\text{com}, \text{dec}') \end{array} \right] < \varepsilon(k), \quad (1)$$

where $\varepsilon(k)$ is a negligible function of k . We then say that the commitment scheme $\text{Comm}(S, R)$ is computationally binding.

Definition 3 (computational hiding property). A commitment scheme $\text{Comm}(S, R)$ is computationally hiding if for every probabilistic polynomial-time party R_{com} , it satisfies

$$\left| \Pr_{y_1}[R_{\text{com}}(\text{pp}, y_1) = 1] - \Pr_{y_2}[R_{\text{com}}(\text{pp}, y_2) = 1] \right| < \varepsilon(k), \quad (2)$$

where pp is a public parameter generated randomly according to the commitment scheme and y_i is a commitment string generated from pp and x_i by S for random x_i sampled from an unknown distribution to R_{com} ($i = 1, 2$).

The computational security of a commitment scheme in this study uses the following assumption.

Definition 4 ((M, δ) -bSVP assumption; see [3]). For a weight parameter, $\delta(n), \delta: \mathbb{N} \rightarrow (0, 1/2)$, and an efficient sampler $\mathcal{M}(1^n)$ that samples $m \times n$ binary matrices, the (M, δ) -bSVP assumption asserts that, for every efficient algorithm Adv , the probability is given by

$$\Pr_{M \leftarrow \mathcal{M}(1^n)}^R [\text{Adv}(M) = x \text{ s.t. } Mx = 0 \text{ and } \Delta(x) \leq \delta] < \varepsilon(n). \quad (3)$$

We introduce a feature of the output locality. We start from the definition of a hash function. A hash function converts input bits of arbitrary length into compressed output bits of shorter lengths. We define the collision resistance of a hash function in Definition 5.

Definition 5 (collision resistance). We have an arbitrary probabilistic polynomial algorithm, Adv , given a description of the hash function and length parameter as inputs. If the probability of Adv that outputs $x, x' \in \{0, 1\}^k$ satisfying $x \neq x'$ and $f(x) = f(x')$ is negligible, the function is a collision-resistant hash function.

$$\Pr[\text{Adv}(f, 1^k) \rightarrow (x, x') \text{ s.t. } x \neq x', f(x) = f(x')] < \varepsilon(k). \quad (4)$$

Next, we define the output locality.

Definition 6 (output locality). We say that the function h has output locality d if each of the output bits depends on at most d input bits.

Finally, we define perfect randomized encoding (PRE). PRE is a technique that can make the output locality a constant.

Definition 7 (perfect randomized encoding; see [3]). Let $f: \{0, 1\}^n \rightarrow \{0, 1\}^l$ be a function. We say that a function $\hat{f}: \{0, 1\}^n \times \{0, 1\}^m \rightarrow \{0, 1\}^s$ is a PRE of f if there exist an efficient decoding algorithm C and a randomized simulator S that satisfy the following:

- (i) Perfect correctness: for every input $x \in \{0, 1\}^n$ and $r \in \{0, 1\}^m$, $C(\hat{f}(x; r)) = f(x)$ holds

- (ii) Perfect privacy: for every $x \in \{0, 1\}^n$, the distribution $\hat{f}(x; r)$ induced by a uniform choice of $r \leftarrow \{0, 1\}^m$ is identical to the distribution of $S(f(x))$
- (iii) Balanced simulation: the distribution $S(y)$ induced by choosing $y \leftarrow \{0, 1\}^l$ is identical to the uniform distribution over $\{0, 1\}^s$
- (iv) Length preserving: the difference between the output length and the total input length of the encoding $s - (n + m)$ is equal to the difference $l - n$ between the output length and the input length of f

3. Building Blocks

In this section, we first define an expanding function ex [3] in Section 3.1. The expanding function is created for the function to apply the (M, δ) -bSVP assumption. We then show an example of PRE and how to make the output locality constant by using PRE in Section 3.2. We also show how to gain $f(x)$ from encoded function $\hat{f}(x)$, which is called perfect correctness in PRE.

3.1. Expand Function ex . We give one expanding function ex used in Theorem 4, where ex is a function of $\{0, 1\}^k \rightarrow \{0, 1\}^n$ that dilutes the relative Hamming weight of the input bits. In order to satisfy the (M, δ) -bSVP assumption, the relative Hamming weight β of the outputs of $\text{ex}(x)$ has to satisfy $\beta \leq \delta/2$ ($\delta \in (0, 1/2)$).

Next, we will explain how the function ex expands the input bits. First, we divide k bit blocks to k/d bit blocks, in which each bit block has d bits, as shown in Figure 1. We execute a function ex0 to each of the d bit blocks, where ex0 expands d bit blocks to c bit blocks, shown in Algorithm 1. Then, every block of the output of ex0 is concatenated as an output of ex ($c \cdot (k/d) = n$). The whole algorithm of ex is given in Algorithm 2. The feature of ex is given in Lemma 1.

Lemma 1 (expand function with low output locality; see [3]). *For $\delta \in (0, 1/2)$, let $\beta \leq \delta/2$ be the relative Hamming weight of ex . Set $n/k \geq \lceil 1/H_2(\beta) \rceil$ and $c \geq \lceil 1/H_2(\beta) \rceil d$ for the natural numbers n, k . Then, there exists an efficiently computable function $\text{ex}: \{0, 1\}^k \rightarrow \{0, 1\}^n$ such that (1) ex is injective, (2) $\Delta(\text{ex}(x)) \leq \beta$ for every x , and (3) ex has output locality d .*

In this study, the hash function H_{Mex} uses an expanding function defined in Lemma 1.

3.2. Construction of PRE. We give one construction of PRE for a given function $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ in 1.

Construction 1 (see [1]). Let f be a function $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$. Then, we separate $f(x)$ to v functions $T_1, \dots, T_v: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ as follows:

$$f(x) = T_1(x) + \dots + T_v(x), \quad (5)$$

where $T_j(x)$ can be written by monomial ($j = 1, \dots, v$). For $r_1, \dots, r_v, r'_1, \dots, r'_{v-1} \in \mathbb{F}_2$, we define a function $\hat{f}: \mathbb{F}_2^n \times \mathbb{F}_2^{2^v-1} \rightarrow \mathbb{F}_2^{2^v}$ by

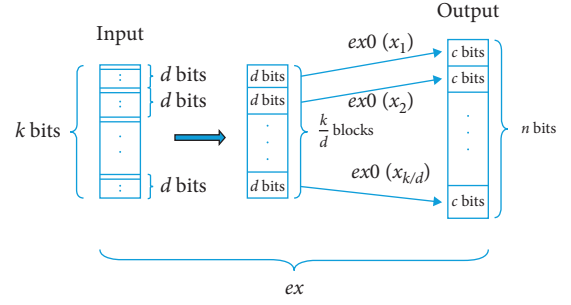


FIGURE 1: ex function.

$$\begin{aligned} \hat{f}(x, (r_1, \dots, r_v, r'_1, \dots, r'_{v-1})) \\ = (T_1(x) - r_1, T_2(x) - r_2, \dots, T_v(x) - r_v, r_1 \\ - r'_1, r'_1 + r_2 - r'_{v-2} + r_{v-1} - r'_{v-1} + r_v). \end{aligned} \quad (6)$$

1 satisfies PRE in Definition 7. Let $f(x) = x_1x_2 + x_2x_3 + x_4$ where $v = 3$ and $n = 4$. Then, f can be encoded as the following equation:

$$\begin{aligned} \hat{f}(x, (r_1, r_2, r_3, r'_1, r'_2)) \\ = (x_1x_2 - r_1, x_2x_3 - r_2, x_4 - r_3r_1 - r'_1, r'_1 + r_2 - r'_2, r'_2 + r_3). \end{aligned} \quad (7)$$

Equation (7) is an example of 1. Denote by $C(z)$ adding all bits in z over \mathbb{F}_2 . Then, we can gain $f(x)$ from $\hat{f}(x)$ by using C as follows:

$$\begin{aligned} C(\hat{f}(x)) &= x_1x_2 - r_1 + x_2x_3 - r_2 + x_4 - r_3 + r_1 \\ &\quad - r'_1 + r'_1 + r_2 - r'_2 + r'_2 + r_3 \\ &= x_1x_2 + x_2x_3 + x_4. \end{aligned} \quad (8)$$

It satisfies “perfect correctness” since $C(\hat{f}(x)) = f(x)$. From the example of equation (7), the output locality of function $f(x)$ can be reduced to a constant by using PRE. A quantitative evaluation of the output locality is given in Lemma 2.

Lemma 2 (see [1]). *Let $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ be a function. Then, let \hat{f} be given as in 1. In particular, if f is a degree- d polynomial written as a sum of monomials, then \hat{f} is a PRE of f with degree d and output locality $\max\{d + 1, 3\}$.*

4. Proposed Commitment Scheme

In this section, we propose a commitment scheme $\text{Comm}_{\text{Mex}}(S, R)$ which is constructed by using ex and \hat{H}_{Mex} . The hash function \hat{H}_{Mex} is PRE of H_{Mex} . We define the decisional (M, δ) -bSVP assumption and show that the (M, δ) -bSVP assumption can be reduced to the decisional (M, δ) -bSVP assumption. Furthermore, we show that our proposed commitment scheme satisfies the binding property and hiding property.

4.1. Difference between $\text{Comm}_{\text{Mex}}(S, R)$ and the Commitment in [3]. In [3], Applebaum et al. showed how to construct a

Input: $x \in \{0, 1\}^d$
Output: $\text{ex0}(x) \in \{0, 1\}^c$
(1) Identify $x \in \{0, 1\}^d$ as a binary representation of natural numbers in $\{0, \dots, 2^d - 1\}$
(2) Set $y \in \{0, 1\}^c$ to the $(x + 1)$ -th string of a relative Hamming weight of maximum value β in the lexicographic order
(3) Return $y \in \{0, 1\}^c$

ALGORITHM 1: ex0 function.

Input: $x \in \{0, 1\}^k$
Output: $\text{ex}(x) \in \{0, 1\}^n$
(1) Partition k -bit inputs into k/d input blocks of d bits each
(2) Apply ex0 to each input block, and generate k/d output blocks of c bits
(3) Return $\text{ex0}(x_1)\text{ex0}(x_2) \dots \text{ex0}(x_{k/d}) = \text{ex}(x)$

ALGORITHM 2: ex function.

statistically hiding commitment scheme with output locality-4 from their collision-resistant hash function under the (M, δ) -bSVP assumption. Their commitment scheme executes a hash function based on a randomness extractor and an ordinary hash function with output locality-4. As a result, two hash functions are required. Furthermore, the randomness extractor is the universal hash function family, so it requires additional random bits to choose a function from the function family. Here, additional bits correspond to the input of the hash function.

On the contrary, our commitment scheme has to only execute an ordinary hash function once. Compared with their commitment scheme, our scheme is more efficient. Furthermore, our commitment scheme achieves output

locality-3 by introducing the new notion of decisional (M, δ) -bSVP assumption.

4.2. Decisional (M, δ) -bSVP Assumption. We introduce a new notion of decisional (M, δ) -bSVP assumption, which is a decisional version of the (M, δ) -bSVP assumption defined in Definition 4.

Definition 8 (decisional (M, δ) -bSVP assumption). For a weight parameter $\delta(n): \mathbb{N} \rightarrow (0, 1/2)$, a uniform distribution $U \in \mathbb{Z}_2^m$, and an efficient sampler $\mathcal{M}(1^n)$ that samples $m \times n$ binary matrices, the decisional (M, δ) -bSVP assumption asserts that, for any polynomial algorithm Adv and for every $x \in \{0, 1\}^n$ where $\delta \leq \Delta(x) \leq 1 - \delta$,

$$\left| \Pr_{M \leftarrow \mathcal{M}(1^n), y_1 \leftarrow M \cdot x} [\text{Adv}(M, y_1) = 1] - \Pr_{M \leftarrow U, y_2 \leftarrow U} [A \text{Adv}(M, y_2) = 1] \right| < \varepsilon(n). \quad (9)$$

We show that the (M, δ) -bSVP assumption can be reduced to the decisional (M, δ) -bSVP assumption by referring to the methodology presented in Lemma 4.2 of [13], where Decisional LWE is reduced to Search LWE.

Theorem 1. Let $y: \{0, 1\}^n \rightarrow \{0, 1\}^m$ be a function, and define (M, δ) -bSVP distribution on m -bit strings obtained by choosing $x \in \{0, 1\}^n$ and outputting $y = M \cdot x$. Assume that we have an access to a procedure D which distinguishes the input y sampled from the distribution of (M, δ) -bSVP or sampled from a uniform distribution U with nonnegligible probability. Then, there exists a polynomial-time algorithm D' such that given samples from (M, δ) -bSVP distribution, D' can output x with nonnegligible probability.

Proof. Let D be a distinguisher which distinguishes an element sampled from the (M, δ) -bSVP distribution or

sampled from a uniform distribution U . Then, we construct D' which finds $x \in \mathbb{Z}_2^n$ of Mx . We first show how D' finds $x_1 \in \mathbb{Z}_2$ which denotes the first coordinate of x . The remaining coordinates can be recovered by the same way.

Given an input of D' , $A = (M, y)$, where y is selected from an (M, δ) -bSVP distribution. The input of D can be defined as follows. Let x be denoted as $x = [x_1, \dots, x_n]$ and M be denoted by the following equation:

$$M = \begin{pmatrix} c_{11} & \cdots & c_{1j} & \cdots & c_{1n} \\ \vdots & \ddots & & & \vdots \\ c_{i1} & & c_{ij} & & c_{in} \\ \vdots & & & \ddots & \vdots \\ c_{m1} & \cdots & c_{mj} & \cdots & c_{mn} \end{pmatrix}. \quad (10)$$

Then, $y = Mx \in \mathbb{F}_2^m$ can be written as

$$y = \begin{pmatrix} c_{11} \cdot x_1 \oplus c_{12} \cdot x_2 \oplus \cdots \oplus c_{1n} \cdot x_n \\ \vdots \\ c_{m1} \cdot x_1 \oplus c_{m2} \cdot x_2 \oplus \cdots \oplus c_{mn} \cdot x_n \end{pmatrix}. \quad (11)$$

For randomly chosen $k \in \mathbb{Z}_2$ and $l_{i1} \in \mathbb{Z}_2$ ($i = 1, \dots, m$), compute a pair

$$A' = \left(M \oplus \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{m1} & 0 & \cdots & 0 \end{pmatrix}, y \oplus \begin{pmatrix} l_{11} \cdot k \\ \vdots \\ l_{m1} \cdot k \end{pmatrix} \right). \quad (12)$$

Denote the value obtained in equation (12) as $A' = (M', y')$. Now, D' sends $A' = (M', y')$ to D . If $k = x_1$, then y' can be written as the following equation:

$$y' = \begin{pmatrix} (c_{11} \oplus l_{11}) \cdot x_1 \oplus \cdots \oplus c_{1n} \cdot x_n \\ \vdots \\ (c_{m1} \oplus l_{m1}) \cdot x_1 \oplus \cdots \oplus c_{mn} \cdot x_n \end{pmatrix}. \quad (13)$$

Since equation (13) can be expressed in the form $y' = M'x$, D can distinguish that equation (13) is contained in the (M, δ) -bSVP distribution. Then, D can distinguish that A' is in the (M, δ) -bSVP distribution. In contrast, if $k \neq x_1$, then y' will be expressed as

$$y' = Mx \oplus \begin{pmatrix} l_{11}k \\ \vdots \\ l_{m1}k \end{pmatrix}, \quad (14)$$

which is clearly not a sample from the (M, δ) -bSVP distribution. Then, D can distinguish that A' is in the uniform distribution.

Finally, D' outputs $k = x_1$ if D outputs (M, δ) -bSVP distribution. On the contrary, D' outputs $k = \bar{x}_1$ if D outputs uniform distribution.

All other remaining coordinates in x can be recovered in the same way. Therefore, D' can output x by using D with nonnegligible probability.

From the contraposition of Theorem 1, we can get Corollary 1. \square

Corollary 1. *There is no polynomial algorithm that can break the decisional (M, δ) -bSVP assumption under the hardness of the (M, δ) -bSVP assumption.*

4.3. Proposed Commitment Scheme $\text{Comm}_{\text{Mex}}(S, R)$. We analyze the hash function \hat{H}_{Mex} in Section 4.3.1 and show our commitment scheme $\text{Comm}_{\text{Mex}}(S, R)$ in Section 4.3.2.

4.3.1. A Hash Function \hat{H}_{Mex} for the Commitment Scheme. We first explain a hash function H_{Mex} [3], containing a matrix M and an expand function ex , as shown in Algorithm 3.

Then, we show the hash function \hat{H}_{Mex} which is PRE of H_{Mex} .

$$\hat{H}_{\text{Mex}}: \{0, 1\}^k \times \{0, 1\}^{nm} \longrightarrow \{0, 1\}^{(1+n)m}. \quad (15)$$

We consider the matrix M as follows:

$$M = \begin{pmatrix} c_{11} & \cdots & c_{1j} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ c_{i1} & & c_{ij} & & c_{in} \\ \vdots & & & \ddots & \vdots \\ c_{m1} & \cdots & c_{mj} & \cdots & c_{mn} \end{pmatrix} = \begin{pmatrix} M[1] \\ \vdots \\ M[i] \\ \vdots \\ M[m] \end{pmatrix}, \quad (16)$$

for $c_{i,j} \in \mathbb{Z}_2$ and $M[i] \in \{0, 1\}^n$ ($i = \{1, \dots, m\}$, $j = \{1, \dots, n\}$). Also, we define the random number $t \in \{0, 1\}^{nm}$ as $t = [T[1], \dots, T[m]]$ where $T[i]$ is taken over uniform $T[i] \in \{0, 1\}^n$ in any $i \in \{1, \dots, m\}$. Furthermore, define $\text{ex}(x)$ as $\text{ex}(x) = [\text{ex}(x)[1], \dots, \text{ex}(x)[n]]$ where $\text{ex}(x)[i]$ is taken over $\text{ex}(x)[i]$ in any $i \in \{1, \dots, n\}$. Note that we write the first coordinate of $T[1]$ as $t[1][1]$. An algorithm of \hat{H}_{Mex} is shown in Algorithm 4. Note that a matrix $M \in \mathcal{M}(1^n)$ is treated as a part of the description of the algorithm.

The hash function \hat{H}_{Mex} is PRE of H_{Mex} since the construction of \hat{H}_{Mex} is as same as 1. Here, we only give a theorem about PRE of H_{Mex} and \hat{H}_{Mex} .

Theorem 2. *\hat{H}_{Mex} satisfies perfect correctness, perfect privacy, balanced simulation, and length preserving for H_{Mex} . We show the output locality of \hat{H}_{Mex} in Theorem 3.*

Theorem 3. *\hat{H}_{Mex} has 3 output localities.*

Proof. Let us investigate the output locality of \hat{H}_{Mex} . From the structure of Algorithm 4, the maximum number of input bits on which the output bits depend is 3. Therefore, the output locality of \hat{H}_{Mex} is 3.

Next, let us discuss the collision resistance of \hat{H}_{Mex} . If a function satisfies the collision resistance, then its PRE also satisfies the collision resistance [1]. Applebaum et al. proved the collision resistance of H_{Mex} . Therefore, the collision resistance of \hat{H}_{Mex} follows from [1]. The collision resistance of \hat{H}_{Mex} is described in Lemma 3. \square

Lemma 3 (collision resistance of \hat{H}_{Mex} ; see [3]). *Let the hash function \hat{H}_{Mex} be a perfectly randomized encoding of H_{Mex} . Then, \hat{H}_{Mex} has a collision resistance under the (M, δ) -bSVP assumption.*

4.3.2. Commitment Scheme $\text{Comm}_{\text{Mex}}(S, R)$. We show the commitment scheme $\text{Comm}_{\text{Mex}}(S, R)$ based on \hat{H}_{Mex} , which consists of initialization, a commitment phase, and a decommitment phase. In this construction, we use the same matrix M , but we can also refresh a matrix M in a certain period, and the computational binding property and computational hiding property also hold using refreshed matrix M . $\text{Comm}_{\text{Mex}}(S, R)$:

Initialization:

Input: $x \in \{0, 1\}^k$
Output: $\mathbf{M} \cdot \text{ex}(x) \in \{0, 1\}^m$

- (1) Partition k -bit inputs into k/d input blocks of d bits each
- (2) Apply $\text{ex}0$ to each input block, and generate k/d output blocks of c bits
- (3) Set $\text{ex}(x)$ as $\text{ex}(x) = \text{ex}0(x_1) \text{ex}0(x_2) \dots \text{ex}0(x_{k/d})$
- (4) Compute $\mathbf{M} \cdot \text{ex}(x)$
- (5) Return $\mathbf{M} \cdot \text{ex}(x)$

ALGORITHM 3: Hash function: $H_{\text{Mex}}(x)$.

Input: $x \in \{0, 1\}^k, t \in \{0, 1\}^{nm}$
Output: $\hat{y} \in \{0, 1\}^{(n+1)m}$

- (1) Compute $\text{ex}(x)$ from x
- (2) **for** $1 \leq i \leq m$ **do**
- (3) **for** $1 \leq j \leq n+1$ **do**
- (4) $v \leftarrow (n+1) * (i-1) + j$
- (5) **if** $v = (n+1) * (i-1) + 1$ **then**
- (6) $\hat{y}[v] \leftarrow (M[i][1] \wedge \text{ex}(x)[1]) \oplus t[i][1]$
- (7) **else if** $v = (n+1) * (i-1) + n+1$ **then**
- (8) $\hat{y}[v] \leftarrow t[i][n]$
- (9) **else**
- (10) $\hat{y}[v] \leftarrow (M[i][j] \wedge \text{ex}(x)[j]) \oplus t[i][j] \oplus t[i][j-1]$
- (11) **end if**
- (12) **end for**
- (13) **end for**
- (14) **return** \hat{y}

ALGORITHM 4: Algorithm of \hat{H}_{Mex} .

Before the commitment phase, both S and R share the following information:

- (i) Algorithm of $\text{ex}: \{0, 1\}^k \rightarrow \{0, 1\}^n$
- (ii) Matrix $M \in \mathcal{M}(1^n)$
- (iii) 1^k : security parameter

Commitment phase by S :

- (1) Choose a random number $r \in \{0, 1\}^{k/2}$ as the key of the hash functions
- (2) Choose a message string $a \in \{0, 1\}^{k/2}$, and concatenate a and r as $x = a \| r$
- (3) Choose a random number $t \in \{0, 1\}^{nm}$ which is used for PRE
- (4) Compute $\text{ex}(x) \in \{0, 1\}^n$
- (5) Compute $\hat{H}_{\text{Mex}}(a, r, t) \in \{0, 1\}^{(n+1)m}$
- (6) Send $\text{com}(a, r, t) = \hat{H}_{\text{Mex}}(a, r, t)$ as a commitment string com

Decommitment phase from S to R :

S executes the following:

- (1) S sends $(a, r) \in \{0, 1\}^{k/2} \times \{0, 1\}^{k/2}$ and $t \in \{0, 1\}^{nm}$ to R as a decommitment string dec

R executes the following:

- (1) Compute $x = a \| r$ from dec .

- (2) Compute $\text{ex}(x)$.

- (3) Compute the commitment string $\hat{H}_{\text{Mex}}(a, r, t)$ and check whether $\hat{H}_{\text{Mex}}(a, r, t) = \text{com}$. If this is satisfied, R outputs a . Otherwise, R outputs \perp .

Next, we prove the computational binding property and computational hiding property of $\text{Comm}_{\text{Mex}}(S, R)$. We first show the computational binding property.

Theorem 4. $\text{Comm}_{\text{Mex}}(S, R)$ satisfies the computational binding property under the (M, δ) -bSVP assumption.

Proof. We assume that there exists a probabilistic polynomial-time (PPT) adversary Adv that breaks the computational binding property of the commitment scheme $\text{Comm}_{\text{Mex}}(S, R)$. Then, Adv can derive the following equation, with nonnegligible function $\epsilon'(k)$ from Definition 2.

$$\Pr[\text{Adv}(1^k, M) \rightarrow (\text{com}, \text{dec}, \text{dec}') \wedge \text{dec} \neq \text{dec}'] > \epsilon'(k) \quad (17)$$

From equation (17), $\text{com} = \hat{H}_{\text{Mex}}(\text{dec})$, and another PPT adversary Adv' , we can lead the following equation:

$$\Pr[\text{Adv}'(1^k, M) \rightarrow (\text{dec}, \text{dec}') \wedge \hat{H}_{\text{Mex}}(\text{dec}) = \hat{H}_{\text{Mex}}(\text{dec}') \wedge \text{dec} \neq \text{dec}'] > \epsilon'(k) \quad (18)$$

This shows that if PPT Adv can break the computational binding property, it can also break the collision resistance of \widehat{H}_{Mex} from equation (18). However, we showed that \widehat{H}_{Mex} has a collision resistance under the (M, δ) -bSVP assumption in Lemma 3. Therefore, the commitment scheme $\text{Comm}_{\text{Mex}}(S, R)$ satisfies the computational binding property under the (M, δ) -bSVP assumption based on the contradiction.

Next, we will prove the computational hiding property of $\text{Comm}_{\text{Mex}}(S, R)$. \square

Theorem 5. *Comm_{Mex}(S, R) satisfies the computational hiding property under the decisional $(M, d/4c)$ -bSVP assumption for a constant $c/d = n/k$.*

Proof. We assume that there exists a probabilistic polynomial-time adversary Adv that breaks the computational hiding property of $\text{Comm}_{\text{Mex}}(S, R)$. For some distinct $a, a' \in \{0, 1\}^{k/2}$, $r, r' \in \{0, 1\}^{k/2}$, $t, t' \in \{0, 1\}^{nm}$, and some nonnegligible function ϵ' , we can derive the following equation:

$$\begin{aligned} & \left| \Pr[\text{Adv}(M, \widehat{H}_{\text{Mex}}(a\|r, t)) = 1] \right. \\ & \left. - \Pr[\text{Adv}(M, \widehat{H}_{\text{Mex}}(a'\|r', t')) = 1] \right| > \epsilon'(k). \end{aligned} \quad (19)$$

Since the decoding procedure C of PRE is a polynomial-time algorithm, there exists a polynomial-time adversary Adv' , which is a composition of the decoding procedure and Adv such that

$$\begin{aligned} & \left| \Pr[\text{Adv}'(M, M \cdot \text{ex}(a\|r)) = 1] \right. \\ & \left. - \Pr[\text{Adv}'(M, M \cdot \text{ex}(a'\|r')) = 1] \right| > \epsilon'(k). \end{aligned} \quad (20)$$

By the hybrid argument, for some a ,

$$\begin{aligned} & \left| \Pr[\text{Adv}'(M, M \cdot \text{ex}(a\|r)) = 1] \right. \\ & \left. - \Pr[\text{Adv}'(M, U) = 1] \right| > \frac{\epsilon'(k)}{2}. \end{aligned} \quad (21)$$

Since r is uniformly random over $\{0, 1\}^{k/2}$, for every $a \in \{0, 1\}^{k/2}$, we have $\Delta(a\|r) \in (1/8, 7/8)$, and hence, $\Delta(\text{ex}(a\|r)) \in (d/(8c), 7d/(8c))$ for a constant $c/d = n/k$ with probability $1 - \exp(-\Omega(k))$ from the Chernoff bounds. This contradicts the decisional (M, δ) -bSVP assumption. \square

4.4. Comparison. We compare our proposed commitment scheme with related works of [BDLOP18] and [KTX08] in Table 1. Both [BDLOP18] and [KTX08] are also based on lattice-based functions and consist of “matrix-vector multiplication” in the same way as us.

A commitment scheme [KTX08] can prove its hiding property statistically and its binding property by the SIS problem. However, it did not achieve constant output locality. A commitment scheme [BDLOP18] can prove its hiding property and binding property by DKS and SKS problems, respectively. Nevertheless, it also did not achieve constant output locality.

TABLE 1: Comparison of our proposed commitment scheme.

	Hiding	Binding	Output locality
[KTX08] [11]	Statistical	SIS	—
[BDLOP18] [14]	DKS	SKS	—
[AHIKV17] [3]	Statistical	bSVP	4
This paper	D-bSVP	bSVP	3

On the contrary, the commitment scheme [AHIKV17] has achieved output locality-4 with its statistically hiding property and its binding property based on the (M, δ) -bSVP assumption (bSVP). However, their commitment scheme was to execute hash functions twice with a randomness extractor. It was also difficult to construct a commitment scheme with output locality-3 by using a randomness extractor.

Our commitment scheme $\text{Comm}_{\text{Mex}}(S, R)$ satisfies output locality-3 by proving its hiding property and binding property by the decisional (M, δ) -bSVP assumption (D-bSVP) and (M, δ) -bSVP assumption (bSVP), respectively. Our commitment scheme only executes the hash function \widehat{H}_{Mex} once and does not use a randomness extractor.

5. Parameter Suggestion for $\text{Comm}_{\text{Mex}}(S, R)$

This section suggests some parameter settings of $\text{Comm}_{\text{Mex}}(S, R)$ under evaluation based on the short integer solution (SIS) problem in Definition 9.

Definition 9 (SIS _{q,m,b} ; see [11]). Given a prime q , a positive number b , and a matrix $MA \in \mathbb{Z}_q^{n \times m}$, the short integer solution (SIS _{q,m,b}) problem is to find a nonzero vector $z \in \mathbb{Z}^m$ such that $Az \equiv 0 \pmod{q}$ and $\|z\| \leq b$.

Let M be a matrix in $\mathbb{F}_2^{m \times n}$. Under the condition of $\Delta(x) \leq \delta$, the (M, δ) -bSVP can be reduced to a SIS _{q,m,b} problem in the lattice spanned by vectors in $\text{Ker}(M)$, where $q = 2$ and $b = \sqrt{n} \cdot \delta$, namely, to solve our scheme is reduced to find a short vector v ($\|v\| \leq \|x\|$) in a lattice $L = \{v \in \mathbb{Z}^n: Mv = 0 \pmod{2}\}$. Denote the norm of the shortest nonzero vector b_1 in ML and the second shortest vector independent with b_1 by λ_1 and λ_2 , respectively. We estimate parameters as follows:

(1) Estimation of δ :

- (a) $\lambda_1(L) = \|x\| = \sqrt{n} \cdot \delta$.
- (b) $\lambda_2(L) \approx \sqrt{n/2\pi e} \cdot 2^{m/n}$ by Gaussian heuristic, where the volume of lattice ML is $\text{vol}(L) = 2^m$ and e is the mathematical constant.
- (c) Denote by $\alpha = m/n$ and $\delta < \alpha/2$ because of the algebraic attack due to [3]. α shows the ratio between input length and output length.

Therefore, we can get a bound of $\delta \leq 0.07$ and $0.14 \leq \alpha$ by $\lambda_1/\lambda_2 < 1.0$ according to the definition of λ_1 and λ_2 above.

- (2) Evaluate the asymptotic complexity to solve a SVP by using Alkim et al.’s estimate proposed in [15], and it had been experimentally verified in [16]. We heuristically set $n = m/\alpha$, $\alpha = 5 \cdot \delta$, and $\delta \leq 0.07$. Then, we

TABLE 2: Parameter suggestions for m, n , and δ being used in our scheme $\text{Comm}(S, R_{\text{com}})$.

Security level	m	n	δ	β_{BKZ}
AES-128	128	426	0.06	376
AES-192	128	731	0.035	601
AES-256	128	1066	0.024	837

input the parameters of $(m, n, \delta, q = 2)$; Alkim et al.'s estimate can evaluate the minimal β_{BKZ} which means the target block size used in the lattice reduction algorithm BKZ [17].

Please refer to [18] for a lucid explanation of Alkim et al.'s estimate. We consider the scenario that one hashes 128 bit information, namely, we fix $m = 128$ in the estimate.

Table 2 shows parameter suggestions of our scheme $\text{Comm}(S, R_{\text{com}})$ with respect to security levels of NIST AES-128, AES-192, and AES-256, where β_{BKZ} is the required block size when using the BKZ algorithm to solve (M, δ) -bSVP. The security levels of "AES-128," "AES-192," and "AES-256" refer to three categories in the NIST PQC standardization project [19] in that the brute force attack on AES key search requires at least 2^{143} , 2^{207} , and 2^{272} classical computing gates, respectively.

6. Concluding Remarks

In this paper, we achieved the following:

- (i) We proposed a new output locality-3 commitment scheme
- (ii) We proved that the (M, δ) -bSVP assumption is reduced to the decisional (M, δ) -bSVP assumption
- (iii) We proved that its computational binding property and computational hiding property are reduced to the (M, δ) -bSVP assumption and decisional (M, δ) -bSVP assumption, respectively
- (iv) We evaluated a secure parameter set against the short integer solution (SIS) problem

Generally, it is easy to build protocols based on the decisional (M, δ) -bSVP assumption compared with the (M, δ) -bSVP assumption. Therefore, our proof would shed light on the new construction of protocols whose security is based on the decisional (M, δ) -bSVP assumption. Also, our method can be used with IoT devices with small CPUs since our method satisfies constant output locality and can be achieved in smaller CPUs. However, it is expected to achieve an output locality-3 commitment scheme with statistical hiding, which is considered an open problem in this work.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was partially supported by enPiT (Education Network for Practical Information Technologies) at MEXT, JSPS KAKENHI (Grant no. JP21H03443) and Ovation Platform for Society 5.0 at MEXT. This work was also supported by JSPS KAKENHI (Grant no. JP21K11751), Japan, JSPS Grant-in-Aid for Scientific Research (A) (nos. 16H01705 and 21H04879), (B) (no. 17H01695), and (C) (no. 21K11887), JSPS Grant-in-Aid for Young Scientists (B) (no. 17K12640), and MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) (Grant no. JPMXS0120319794).

References

- [1] B. Applebaum, Y. Ishai, and E. Kushilevitz, "Cryptography in NCO," *SIAM Journal on Computing*, vol. 36, no. 4, pp. 845–888, 2006.
- [2] M. Naor and O. Reingold, "Synthesizers and their application to the parallel construction of pseudo-random functions," *Journal of Computer and System Sciences*, vol. 58, no. 2, pp. 336–375, 1999.
- [3] B. Applebaum, N. Haramaty, Y. Ishai, E. Kushilevitz, and V. Vaikuntanathan, "Low-complexity cryptographic hash functions," in *Proceedings of the 8th Innovations in Theoretical Computer Science Conference, ITCS*, Berkeley, CA, USA, January 2017.
- [4] B. Applebaum, "Garbled circuits as randomized encodings of functions, a primer," *Electronic Colloquium on Computational Complexity*, vol. 24, 2017.
- [5] J. Groth, "Homomorphic trapdoor commitments to group elements," *Cryptology ePrint Archive*, vol. 7, 2009.
- [6] I. Damgard, "Commitment schemes and zero-knowledge protocols," in *Lectures On Data Security, Modern Cryptology In Theory And Practice, Volume 1561 of LNCS*, I. Damgard, Ed., Springer, Berlin, Germany, 1998.
- [7] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proceedings of the CRYPTO '91, 11th Annual International Cryptology Conference*, Santa Barbara, CA, USA, August 1991.
- [8] S. Halevi and S. Micali, "Practical and provably-secure commitment schemes from collision-free hashing," in *Proceedings of the Advances in Cryptology-CRYPTO '96, 16th Annual International Cryptology Conference*, pp. 201–215, Santa Barbara, California, USA, August 1996.
- [9] H. Miyaji, A. Kawachi, and A. Miyaji, "String commitment scheme with low output locality," in *Proceedings of the 2019 14th Asia Joint Conference on Information Security*, pp. 32–39, Kobe, Japan, August 2019.
- [10] H. Miyaji, A. Miyaji, and Y. Wang, "Homomorphic commitment scheme with low output locality," in *Proceedings of the 2020 the Eighth International Symposium on Computing and Networking, CANDAR*, November 2020.
- [11] A. Kawachi, K. Tanaka, and K. Xagawa, "Concurrently secure identification schemes based on the worst-case hardness of lattice problems," in *Proceedings of the Advances in Cryptology-ASIACRYPT 2008*, pp. 372–389, Melbourne, Australia, November 2008.
- [12] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith, "Efficient and non-interactive non-malleable commitment," in *Proceedings of the EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques*, pp. 40–59, Innsbruck Austria, May 2001.

- [13] O. Regev, “On lattices, learning with errors, random linear codes, and cryptography,” *Journal of the ACM*, vol. 56, no. 6, p. 34, 2009.
- [14] C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert, “More efficient commitments from structured lattice assumptions,” in *Proceedings of the Security and Cryptography for Networks-11th International Conference, SCN 2018*, D. Catalano and R. D. Prisco, Eds., pp. 368–385, Amalfi, Italy, September 2018.
- [15] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, “Post-quantum key exchange-a new hope,” *USENIX Security Symposium*, pp. 327–343, 2016, Report number: 2015/1092.
- [16] M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, “Revisiting the expected cost of solving usvp and applications to LWE,” in *Proceedings of the Advances in Cryptology-ASIACRYPT 2017*, pp. 297–322, Hong Kong, China, December 2017.
- [17] Y. Aono, Y. Wang, T. Hayashi, and T. Takagi, “Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator,” in *Proceedings of the EUROCRYPT 2016-35th Annual International*, pp. 789–819, Vienna, Austria, May 2016.
- [18] W. Wang, Y. Wang, A. Takayasu, and T. Takagi, “Estimated cost for solving generalized learning with errors problem via embedding techniques,” in *Proceedings of the Advances in Information and Computer Security 2018*, pp. 87–103, Sendai, Japan, September 2018.
- [19] Us Department of Commerce and National Institute of Standards and Technology, *Post-Quantum Cryptography*, <https://csrc.nist.gov/projects/post-quantum-cryptography/>, 2020.

Research Article

NSGA-II-Based Granularity-Adaptive Control-Flow Attestation

Jing Zhan ^{1,2}, Yongzhen Li ^{1,2}, Yifan Liu ^{1,2}, Hongchao Li ^{1,2}, Shuai Zhang ^{1,2},
and Li Lin ^{1,2}

¹School of Computer Science, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China

²Beijing Key Laboratory of Trusted Computing, Beijing 100124, China

Correspondence should be addressed to Jing Zhan; zhanjing@bjut.edu.cn, Yongzhen Li; liyongzhen97@163.com, and Li Lin; linli_2009@bjut.edu.cn

Received 17 September 2021; Accepted 27 October 2021; Published 18 November 2021

Academic Editor: Chunhua Su

Copyright © 2021 Jing Zhan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Since the widespread adoption of edge computing and IoT technology, Control-Flow Hijacking (CFH) attacks targeting programs in resource-constrained embedded devices have become prevalent. While the Coarse-Grained Control-Flow integrity Attestation (CGCFA) lacks accuracy for the CFH attacks detection, the Fine-Grained Control-Flow integrity Attestation (FGCFA) detect the attacks more accurately but with high overheads, which can be a big burden (e.g., to industrial control system with strict performance requirements). In this paper, we propose a NSGA-II (Nondominated Sorting Genetic Algorithm-II) based Granularity-Adaptive Control-Flow Attestation (GACFA) for the programs in embedded devices. Specifically, we propose a Granularity-Adaptive Control-Flow representation model to reduce the complexity of programs' control-flow graph and propose NSGA-II-based granularity-adaptive strategy generation algorithm to balance the security and performance requirements. Besides, runtime protection for the GACFA at the program end with SGX is proposed to protect the integrity and confidentiality of control-flow measurement data. The experiments show that our work can find out the best-so-far control-flow granularity with stability and provide secure program attestation for the verifier. In addition, the security/performance benefit of adopting our proposal over CGCFA is 13.7, 25.1, and 43.0 times that of adopting FGCFA over ours in different threat scenarios.

1. Introduction

With the rapid development of edge computing and IoT technology, more and more embedded devices are connected together and reach people's daily works and lives, which brings us both convenience and security concerns. For example, the connection between the industrial control system which contains lots of resource-constrained embedded devices and the corporate intranet or the Internet has increased the attack surface, leading to more remote attacks [1]. In recent years, Control-Flow Hijacking (CFH) attacks, which can directly tamper with the behaviour of programs running in the memory, make it a challenge to ensure the security of programs of embedded devices.

Remote attestation is a method of verifying the integrity of the software on a remote device. At the end of the attestation, the verifier obtains a report signed by a security chip (e.g., Trusted Platform Module, TPM) inside the device

from the prover on whether the hardware and software codes' hashes to be executed meet the verifier's expectations. Later, the research on memory verification and attestation of the hosts [2] and embedded devices [3] is proposed too. However, these memory verification and attestation works can only verify the integrity of the whole memory chunk but cannot find out whether the control flow of specific program running in the memory is hijacked.

In recent years, practical control-flow integrity research has been proposed to ensure runtime program integrity by comparing whether the destination address of the jump instruction is in the branch target sets recorded by pre-analysis [4]. However, the method of comparing address and target sets can only find out the attacks tampering with the control data (the called function pointer, the address pointed to by the jump instruction inside the function, etc.) but cannot detect the hijacking attack on the program control flow by tampering with the noncontrol data (such as branch

and loop variable value) inside the function [5]. Later, the fine-grained control-flow integrity measurement and attestation [6, 7] are proposed to verify runtime program control flow more accurately, since it obtains more context information in the granularity of basic block, but with increased overheads, which is not efficient for providing real-time and reliable services in mission critical systems, for example, industrial control system.

In order to balance the security and efficiency of CFH attacks detection in a resource-constrained embedded environment, this paper proposes NSGA-II (Nondominated Sorting Genetic Algorithm-II) based Granularity-Adaptive Control-Flow Attestation (GACFA), which take functions and basic blocks as different control-flow monitoring granularities. Through genetic algorithm NSGA-II, basic block-level fine-grained control-flow monitoring is performed on core functions that have a greater impact on program security, and function-level control-flow monitoring is performed on a noncore function. In this way, the verifier can verify in runtime whether the program has suffered CFH attacks with balanced security and overhead costs.

The contributions of this paper are as follows:

- (1) A Granularity-Adaptive Control-Flow representation model for CFH detection is proposed, in which the granularity of function and basic block and virtual nodes are introduced to reduce the complexity of programs' control-flow graph and to mitigate verification path explosion problem.
- (2) A NSGA-II-based granularity-adaptive strategy generation method on multiobjective optimization algorithm is proposed, which combines low-overhead and high control-flow security as the optimization goal. With this method, the optimal granularity division strategy is generated, achieving the balance between CFH detection accuracy (e.g., the security goal) and detection overheads (e.g., the performance goal).
- (3) The Granularity-Adaptive Control-Flow Attestation protected by the SGX technology is also proposed to protect the integrity and confidentiality of control-flow measurement data on the program-end environment.
- (4) We also implement a proof of concept system of GACFA. The experiments show that our work can find the best-so-far control-flow granularity with stability and provide secure program attestation for the verifier. The security/performance benefit of adopting our proposal over CGCFA is 13.7, 25.1, and 43.0 times that of adopting FGCFA over ours in different threat scenarios.

2. Related Works

Research on program verification and attestation at the control-flow level is mainly in two fields, including Control-Flow Integrity (CFI) detection/prevention and Control-Flow integrity Attestation (CFA).

The CFI detection/prevention is proposed to detect/prevent the hijacking on program control flow by checking it with the original control-flow graph (CFG) or predefined control-flow policy (e.g., matching set of targets for each indirect control transfer). The GRIFFIN [8] provided online CFI enforcement over unmodified binaries by checking captured control flows with predefined control-flow policies with the aid of the hardware. CFI based on a Lightweight Encryption Architecture with Advanced Encryption Standard (LEA-AES) [9] was proposed to ensure its security through encryption. Burrow et al. [10] introduced and compared a broad range of CFI mechanisms and discussed them from the aspects of precision, security, and performance. In order to improve the accuracy, lots of research was proposed to use more context information of CFI detection. Ding et al. [11] proposed a path-sensitive CFI scheme, which used path-sensitive points at runtime to calculate the legal control transfer target. The value-based constraint CFI (vCFI) [12] was proposed to improve the effectiveness of CFI by monitoring and protecting control and noncontrol data that may be manipulated for Control-Flow Hijacking. The Origin-Sensitive CFI (OS-CFI) [13] was proposed to improve the security of CFI with new contextual information, such as the last branches taken. Khandaker et al. [14] also proposed Control-Flow Integrity with Backtracking (CFI-LB) with adaptive call site sensitivity, which improves program security. Jang et al. [15] proposed an Index-based Bit Vector Control-Flow Integrity (IBV-CFI) scheme to reflect the CFG with accuracy. All in all, the current CFI schemes (including control-flow capture and integrity verification) with more context information are costly and thus expensive for resource-constrained embedded devices.

Remote attestation allows a resource-rich verifier to obtain the program status of the prover with limited resources, which can reduce the CFI verification overhead for embedded devices. The traditional remote attestation schemes are mostly static or coarse, which can only guarantee the integrity of the binary code before the program runs or guarantee the security of the whole embedded devices' memory [3]. Later, C-FLAT (Control-Flow Attestation) [6] was proposed to the verifier with the proof of the application's control-flow path on the prover's device using ARM TrustZone hardware. Then, LO-FAT (Low-Overhead Control-Flow Attestation) [16] was introduced as a hardware version of C-FLAT with lower performance overhead. A runtime remote attestation system ATRIUM [7] was also proposed to attest both the code's binary and its execution behaviour to resist Time of Check Time of Use (TOCTOU) attacks. In addition, ATRIUM provided the method to protect codes' control flow and CFG with the help of the FPGA. Although the hardware-based CFA improves the security of the attestation mechanism, it means extra hardware support and costs.

The SGX (Software Guard Extensions) is an instruction set extension supported by Intel CPU since 2013 [17, 18]. The SGX can provide applications with a trusted execution environment for many scenarios now, such as secure cloud computing [19] and machine learning [20]. SGX is also widely adopted for code and data confidentiality protection for

endpoint systems and devices with relatively cheap performance costs compared to other customized secure hardware because it is a Commercial Off-The-Shelf (COTS) solution. For example, Wang et al. [21] proposed security-enhanced attestation between IoT terminals and devices with SGX-protected attestation codes and keys with about additional 3% time costs. SGX-Log [22] was proposed to protect the system logging program and log data's HMAC key using SGX enclave and sealing primitives with an overhead of 4.84% in log generation and 6.29% in log reading.

3. Threat Model and Assumptions

The existing CFH attacks can be categorized into two groups of attacks, where one is attacks tampering control data and the other is attacks tampering noncontrol data of program control flows. Specifically speaking, the control data includes the return address pointed to by the indirect jump instruction, function pointer, saved return address of functions, and so on. Noncontrol data can be divided into control flow-related variable data, such as branches and loop variables that affect program control flow, and pure data, such as program variables that have nothing to do with control flow. This paper mainly studies control data and control flow-related noncontrol data attacks detection and attestation.

We assume that, in a running program's memory, the code segment can be read and executed, while data and stack/heap segment can only be read and written. This protection is already widely adopted and can ensure that malicious code cannot be executed in the data and stack/heap segment. Figure 1 shows four types of control-flow attacks that the attackers may launch on a program that diverts the original path of the program to an unexpected one. The control-flow graph on the right side represents the control flow of the program running in the memory.

Attack 1 (branch variable attacks): this type of attack achieves the purpose of the attack by tampering with the variables that control the direction of the branch. The attack makes the program control flow going to a branch different from the original one. For example, as the CFG in Figure 1 shows, the attacker tampers with the condition variable in node 1, causing the control flow that should have jumped to node 2 to jump to node 6, making the control flow to an unexpected but legal path.

Attack 2 (loop variable attack): this attack tampers with the loop variable to change the number of loops. For example, the attacker may skip the loop of node 2 by changing the cyclic variable that controls the number of program loops to 0.

Attack 3 (return address attack): this attack alters the return address of functions to make the program control flow to the wrong place. For example, after function calling, node 3 should return to node 2, but if the return address is tampered with, the program returns to node 5 instead.

Attack 4 (function pointer attack): this kind of attack makes the program control flow go to the wrong direction by tampering with the function pointer. For example, after the execution of node 4, the program should call the function going to node 5, but the attacker tampered with the calling function pointer to cause the program to turn to node 7.

We also give the following three assumptions for our method.

- (1) GACFA detects attacks that affect program control flow but does not detect pure data-driven attacks. Because pure data attacks are related to specific applications, they often need to be jointly considered with other methods.
- (2) Attackers cannot physically tamper with the program code while the program is running. We assume it is more difficult for the attackers to tamper with the hardware device than with embedded software. In some cases, embedded devices (e.g., industrial control equipment) usually receive certain physical protection and are difficult for an attacker to tamper with physically.
- (3) The verifier has enough memory space to store the control-flow information of the program to be verified and thus can help ensure the integrity and confidentiality of the verification process. It is assumed that the verifier has more resources and thus more security mechanisms (e.g., could be SGX) than the embedded device to be verified.

4. System Architecture

This paper proposes a Granularity-Adaptive Control-Flow Attestation (GACFA) method based on NSGA-II [23]. In this way, we extend the traditional static binary attestation and the CFA at the control-flow level to the dynamic runtime attestation with adaptive granularity. As shown in Figure 2, our system architecture consists of two roles: the verifier and the prover. The verifier is responsible for offline verification of the program to be verified, including a granularity division module and a control-flow measurement module. The online attestation module is deployed between the verifier and the prover. In addition, the prover is also in charge of collecting control-flow information of the program and prover-end control-flow measurements calculation.

4.1. Granularity Division Module. We regard the function as the high-level unit for program execution, and we mark every function as the core function or noncore function according to the importance of the function to the program. Then, we perform basic block-level control-flow monitoring for core functions and perform function-level control-flow monitoring for noncore functions.

Definition 1 (core function). It refers to the function that has a strong impact on the whole program. The more the function is called, the more important the function is to the program, which implies it is easier to be exploited by an attacker.

Definition 2 (noncore function). It refers to the function that has less impact on the whole program.

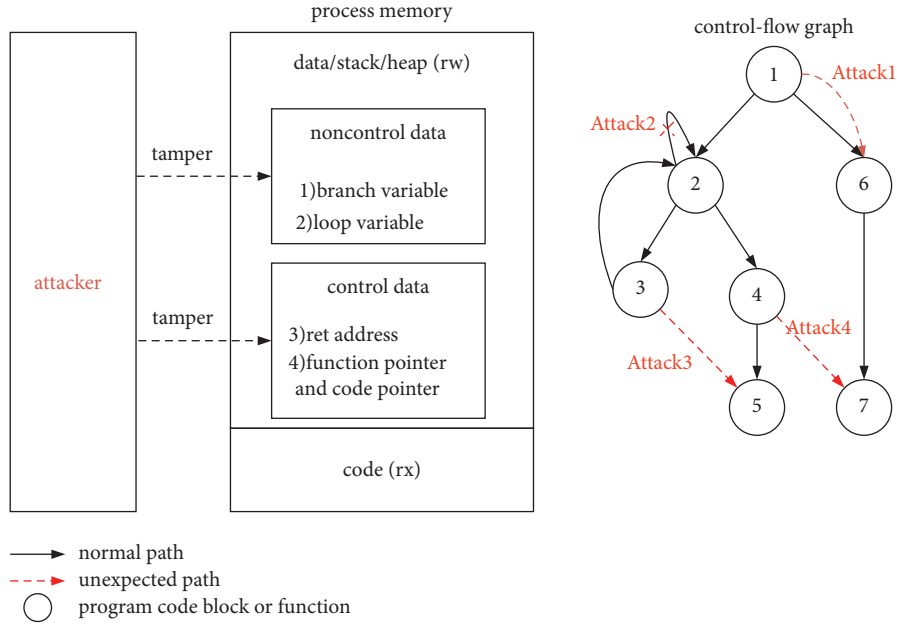


FIGURE 1: Threat model.

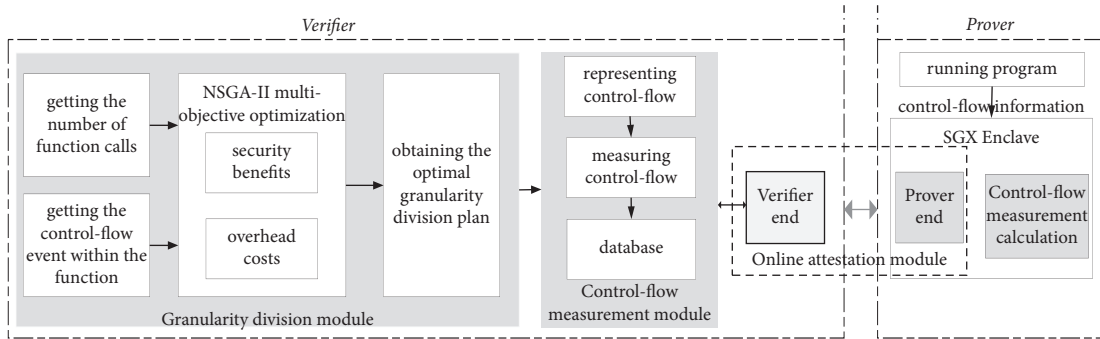


FIGURE 2: System architecture of GACFA.

Granularity division can be regarded as a combined optimization problem, which integrates two goals, that is, reducing overheads and improving control-flow security. The granularity division module includes three parts: input generation, target optimization, and strategy selection.

- (1) **Input generation:** the program A to be verified is usually composed of multiple functions. Firstly, the verifier obtains the function set $F(A)$ contained in program A with static analysis. Then the verifier dynamically analyses the execution process of program A under a specific input (or no input). For function f_i in $F(A)$, count the number of times f_i is called, and the number of control-flow events contained in f_i . Here, the number of control-flow events is the number of the basic blocks in the function. At last, we take the number of f_i calls and the number of control-flow events as the input of the NSGA-II algorithm, where the number of function calls represents the degree of the security impact of the

function on the entire program, and the number of control-flow events represents the overhead.

- (2) **Target optimization:** we use NSGA-II multiobjective optimization method to evaluate the combined target, which is composed of control-flow security and overhead. After determining the control-flow granularity of the program, its security benefits are measured. And the overhead refers to the time required to authenticate the program's control flow.
- (3) **Strategy selection:** we select the optimal core function marking scheme according to the optimization result, in order to perform fine-grained measurement on core functions and coarse-grained measurement on noncore functions later.

4.2. Control-Flow Measurement Module. The control-flow measurement module is responsible for calculating the control-flow hash value of the program to be verified, which

is divided into control-flow representation, control-flow measurement, and the measurement database.

- (1) Control-flow representation: according to the division results, we perform basic block-level instrumentation on the core function and function-level instrumentation on the noncore function. Then, we obtain the control-flow information that combines the coarse and fine granularity of the program.
- (2) Control-flow measurement: after control-flow representation, we can get the granularity-adaptive CFG of the program. The type of the node in the CFG can be function node, basic block node, and so on. Then, we generate the expected measurement value along the path of the CFG with accumulated hashing. The calculation formula is as follows:

$$\begin{cases} H(A_1) = H(H(0), A_1) & i = 1, \\ H(A_i) = H(H(A_{i-1}), A_i) & i > 1 \& \& \text{type}(A_i) = \text{type}(A_{i-1}). \end{cases} \quad (1)$$

Here, A_i is the node i in program's CFG. $H(0)$ is the initial hash value, which is all zero. There are three types of nodes, including function node, basic block node, and virtual node, which is specified in Section 5. For each path in the CFG, the path is divided into several parts when the node type changes. For a part in the path, the hash value of the current node and later node are taken as the inputs of the SHA-256 hash algorithm to get the accumulated hash value until all nodes are processed.

As formula (2) shows, after every parts' hash value is calculated, all parts' hash value can be accumulated again in the way similar to formula (1), and the final hash value of every path is the measurement of current program control flow. In this way, if the final hash value verification failed, we can find out which part is wrong quickly:

$$\begin{cases} H(P_1) = H(H(0), P_1) & i = 1, \\ H(P_i) = H(H(P_{i-1}), P_i) & i > 1. \end{cases} \quad (2)$$

- (3) Database: it stores all the parts and paths (in the CFG of the program to be attested) accumulated hash value corresponding to program A in the measurement database.

4.3. Online Attestation Module. The online attestation module contains the verifier-end and the prover-end. As the prover obtains the program's runtime control-flow information, it generates and submits the signed control-flow report to the verifier for runtime verification. At the prover-end, Intel SGX provides a trusted execution environment for the prover to calculate the final hash value on the program's execution path and ensures the integrity of the report. The remote attestation protocol is as shown in Figure 3 and the following description.

Step 1: the verifier sends a challenge to the prover to indicate the attestation request, including the id of the program to be authenticated, the random number N , and the input i of the program. Among them, the random number is to ensure the freshness of the attestation result and prevent replay attacks.

Step 2: after receiving the attestation request from the verifier, the prover initializes the attestation mechanism and runs program A under input i . The control flow of the program is sent to the measurement module in the SGX enclave with the help of the runtime tracing tool (e.g., Intel Pin). The obtained control-flow information is calculated with formulas (1) and (2), and the final hash value of program's control flow and N are signed by the private key sk of the enclave for attestation report r generation.

Step 3: the prover sends signed r and the measurement h to the verifier.

Step 4: the verifier uses the public key pk to verify the signature after receiving the attestation report. If the verification succeeds, the verifier continues to check whether the control-flow measurement is consistent with the expected value under input i stored in the database. If the result is true, it means that program A has not been attacked by Control-Flow Hijacking.

4.4. SGX Protected Control-Flow Measurement Module. We use the SGX to create a trusted execution environment for control-flow measurement calculation and the prover-end attestation to protect the measurement calculation and attestation report generation.

The prover-end attestation module is already introduced in Section 4.3. And the control-flow measurement calculation module gets control-flow information from the running program and then calculates the hash value of the program execution path.

5. Granularity-Adaptive Control-Flow Representation Model

In order to perform Control-Flow Attestation, the verifier asks the execution path of the program to be measured from the prover's device. Recording and transmitting each executed instruction in the execution path is not feasible because it will lead to very long and nested paths requiring the prover to be calculated and the verifier to traverse, which is costly to attestation. To reduce the complexity of the program's control-flow graph and balance the overheads and the security of Control-Flow Attestation, this paper proposes a granularity-adaptive control flow representation model, which provides basic block-level monitoring for core functions and function-level monitoring for noncore functions. The specific definition of the model is introduced below.

Definition 3. Directed graph of the control flow of a program, denoted as $G = \langle V, E, A \rangle$. The model of granularity-

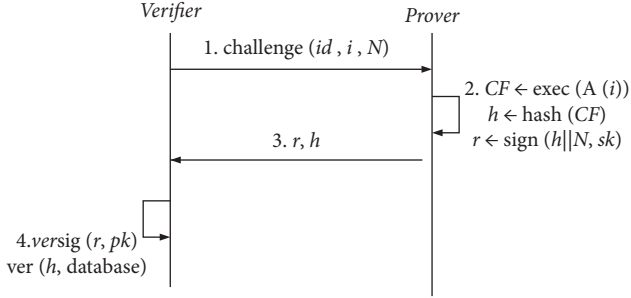


FIGURE 3: Online attestation protocol.

adaptive control flow is a directed graph representing program control-flow information, which is defined as a triplet $\langle V, E, A \rangle$.

Definition 4. Vertices set of G , denoted as V . $V = \{v_i | i \in N\}$. V is the set of vertices in G and each vertex represents a control-flow node, which can be a function, basic block, or virtual node.

Definition 5. Edge set of G , denoted as E . $E = \{v_i \rightarrow v_{i+1} | i \in N\}$. E is the set of edges in G , which is used to represent the call, jump, and return relationship between node v_i and v_{i+1} .

Definition 6. Attribute set of a node i , denoted as A_i . $A_i = \{(\text{type}, \text{cont1}, \text{cont2})\}$. $\forall v \in V$; define A_i as the attribute set of node i , where the type indicates the type of the node i and $\text{type} \in \{00, 01, 10\}$, and cont1 and cont2 specify the context attributes of the node i (e.g., the start address, the next jump address, etc.). As shown in Figure 4, there are three types of nodes, which are function node, basic block node, and virtual node, explained as follows.

- (1) Function node: we monitor noncore functions at the function level and record the calling relationships of functions. The function call relationship can describe the execution path of the program in a coarse-grained manner. For $\forall f \in F(A)$, if there is a direct edge from f_i to f_{i+1} , it means that the function f_i calls the function f_{i+1} . The function node is represented by “00,” and the content includes the entry address of the function and the entry address of the called function. The outdegree of a node represents the number of functions directly called by the function represented by the node, and the indegree indicates the number of times the function is called; if a node has no indegree, the node is generally considered to be an entry function. And if a node has no outdegree, the node is generally considered a function call path.
- (2) Basic block node: we monitor the core functions at the basic block level. The basic block consists of a section of assembly instructions with only entry and exit. The entry event may be the first statement of the program, the target statement of a conditional jump or unconditional jump, and a conditional jump. The

function node	00	function entry address	the entry address of the called function
basic block node	01	basic block first address	jump address
virtual node	10	loop entry address	number of cycles

FIGURE 4: Node structure.

exit event may be a stop statement, a jump statement, or the previous statement of the jump target statement. The basic block node is identified by “01,” and the content includes the first address of the basic block, that is, the address of the first instruction in the basic block and the address of the basic block to jump to. The core function is composed of basic block nodes, including start nodes, internal nodes, and exit nodes. Among them, the start node is the function entry block, and the exit node is the end block of the function, such as the ret instruction. The internal node is a basic block divided by the internal instructions of the function.

- (3) Virtual node: when a program is running, the loop codes may generate lots of basic block jump edges, which are basically the same, causing a large number of repeated measurements and increasing unnecessary performance costs. This paper introduces a virtual node to represent the loop codes in the function, using the “10” mark, and the content includes the loop entry address, that is, the loop ID and the number of loops.

The main function of virtual nodes is to separate common basic blocks and loops to prevent iterative calculations during measurement. The loop structure in the program includes do-while, while, and for. From their assembly code, it can be found that the conditional branch of the backward jump points to the beginning of a loop. This paper uses this common feature as the basis for judging whether to start the loop. When the loop contains jump statements, break, and continue, its control flow will change. For break jump statements, when executed, they jump directly outside the loop, which is equivalent to the exit statement of the second loop, so the normal loop control flow will be recorded, as well as the loop control flow when it encounters a break. When continue is executed, it will jump to the beginning of the next loop, which will also lead to inconsistent control flow, so we will record the number of executions of the continue statement and the control-flow information when the continue is executed. When we encounter a jump statement, we get at least two control-flow metrics, and the repeated codes are only calculated once.

Taking Figure 5 as an example, we explain the control flow measurement and verification as follows.

We can see from Figure 5 that $V = \{F1, F2, B3, B4, B5, B6, B7, B8\}$,

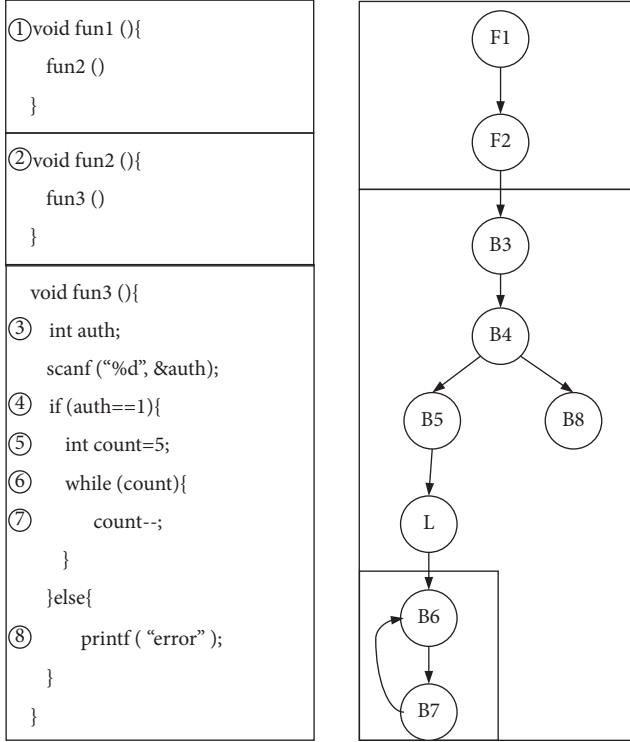


FIGURE 5: Control-flow graph of a program.

$E = \{F1 \rightarrow F2, F2 \rightarrow B3, B3 \rightarrow B4, B4 \rightarrow B5, B5 \rightarrow B6, B6 \rightarrow B7, B7 \rightarrow B6, B4 \rightarrow B8\}$. Suppose $fun1$ and $fun2$ are noncore functions, and $fun3$ is the core function. Node $V_0 \in \{F1, F2\}$ is a function node, and $F1 \rightarrow F2$ represents $fun1$ calling $fun2$. The node $V_1 \in \{B3, B4, B5, B6, B7, B8\}$ is a basic block node. Node L is a virtual node, which points to a loop structure used to store the number of loops. When $auth = 1$, if the virtual node is not introduced, the program execution path is. However, if we introduce the virtual node L , the program execution path will be $P' = \{F1 \rightarrow F2 \rightarrow B3 \rightarrow B4 \rightarrow B5 \rightarrow L \rightarrow B6 \rightarrow B7\}$. And the program execution path length is reduced from 17 (the length of P) to 8 (the length of P').

As shown in Figure 5, the control-flow graph consists of two paths, and the first path is divided into 3 parts. The measurement results are as follows:

$$F1 \rightarrow F2: H_0 = H(H(H(0), F1), F2),$$

$$B3 \rightarrow B5: H_1 = H(H(H(H(0), B3), B4), B5), \quad (3)$$

$$L: H_2 = H(H(H(H(0), B6), B7) \parallel \text{loop_num}).$$

Each part and the whole path's measurements are stored in the measurement database in the form of a key-value pair. The key contains the start node and the end node, and the value is the metric value of the part. The first path is stored as $\{(F1 \rightarrow F2: H_0), (B3 \rightarrow B5: H_1), (L: H_2)\}, H_{\text{path}}$.

Then, we show the four types of attacks (in Figure 1) detection with the example program shown in Figure 5:

- (1) Branch variable attack: the attack tampers with the verifier's input $auth$, since the control flow that should have jumped to $B5$ is changed to jump to $B8$.

The control flow leads to an unexpected but legal path.

- (2) Loop variable attack: the attack tampers with the value of the loop control variable count, causing the number of while loops to change.
- (3) Return address attack: the attack tampers with the return address of the function $fun2$, so that $func2$, which should have returned $fun1$, returns to an illegal address outside of $fun1$, for example, attacker's rogue codes.
- (4) Function pointer attack: the attack the attacker tampered with the code pointer to make node $B5$ turn to $B3$.

Due to the abovementioned four attacks, the original control flow changes, resulting in a change in the final calculated hash value of the execution path, which can be detected by our method.

6. NSGA-II-Based Granularity-Adaptive Strategy Generation

Many embedded devices (e.g., industrial control devices) have high requirements for real-time performance. In this case, fine-grained control-flow monitoring at the basic block level can provide higher security guarantees, but the performance overhead is relatively high. In contrast, control-flow monitoring at the function level cannot detect Control-Flow Hijacking attacks related to control data due to lack of context information, such as loop variable attacks.

This paper marks the function as a core function or a noncore function and performs basic block-level fine-grained control-flow monitoring for core functions and function-level coarse-grained control-flow monitoring for noncore functions. More marked core functions, more functions need fine-grained control-flow monitoring, but the security benefit improvement may be very little and cause high-performance overhead, especially for resource-limited devices (e.g., IoT or industrial control devices).

We define the optimization goals as security benefits and overhead costs. Then, we transform the above problem into a multiobjective optimization problem as follows:

- (1) The goal of optimization:

$$\begin{aligned} & \max\{\text{security}(X)\}, \\ & \min\{\text{overhead}(X)\}. \end{aligned} \quad (4)$$

- (2) The condition of constraint:

$$\begin{aligned} & \text{security}(X) > s_{\text{threshold}}, \\ & \text{overhead}(X) < o_{\text{threshold}}. \end{aligned} \quad (5)$$

- (3) Decision variables:

$$X = \{x_1, x_2, x_3, \dots, x_n\}. \quad (6)$$

- (4) Parameters:

X is the marking vector, representing the marking scheme on all the functions of a program, showing which functions should be core or noncore functions.

$\text{Security}(X)$ refers to the security benefits of the system under the function marking scheme X .

$\text{Overhead}(X)$ refers to the overhead costs under function marking scheme X .

$s_{\text{threshold}}$ refers to the value of security benefits when all noncore functions are used.

$o_{\text{threshold}}$ refers to the overhead when all core functions are used.

Suggest there are n functions in a program, for bit i of marking vector X , referred to as x_i . There is

$$x_i = \begin{cases} 1, & \text{core function,} \\ 0, & \text{noncore function.} \end{cases} \quad (7)$$

Among them, the constraint conditions ensure that our function marking scheme will not be all core functions or all noncore functions, resulting in excessive overhead or low-security benefits.

- (1) Security benefits: for optimization of a granularity division plan, one of the optimization goals is the security benefit of the system, that is, the security of the control flow of the program because the finer the monitoring granularity is, the more the security of the control flow can be guaranteed. So, we hope to perform fine-grained monitoring on core functions and use the importance of the function and the attacks that can be detected under different granularity monitoring to quantify the security of the control flow as follows:

$$\text{Security} = \sum_{i=1}^n \text{dep}_i \times \text{att_num}_i, \quad (8)$$

$$\text{dep}_i = \frac{\text{fcall}_i}{\text{fcall}_{\text{total}}}. \quad (9)$$

Among them, n is the number of functions in the program and i corresponds to the number of the function.

This paper introduces the concept of function dependency (dep) as an indicator to determine whether a function is a core function. Function dependency refers to the proportion of the number of times a function is called to all calling events during the running of the program. The greater the function dependency, the greater the impact on the entire program after the function is destroyed. So, dep_i represents the probability that the i -th function is marked as a core function. $fcall_i$ is the number of calls of the i -th function, and $fcall_{\text{total}}$ is the sum of the number of calls of all functions.

att_num is the number of attack types that can be detected by different granularity monitoring methods and represents the ability of the granularity monitoring method to provide security protection. As shown in the threat model, this paper proposes four control flow-related attacks. The tampering of control data such as code or function pointer and return address tampering can be detected by coarse-grained monitoring methods, and fine-grained monitoring methods can detect all attacks. Therefore, $att_num_i \in \{2, 4\}$, and when the i -th function performs coarse-grained monitoring, the value is 2. Otherwise, the value is 4.

- (2) Overhead: some resource-limited devices (e.g., industrial control devices) have extremely high requirements for availability. From this perspective, reducing overhead is an important optimization goal. We use the control-flow event to represent a node in the control-flow graph. Especially with fine-grained monitoring, the number of control-flow events is the number of basic block nodes. The total overhead depends on the number of control-flow events.

Our experiments show the relationship between overhead and control-flow events in Figure 6. The abscissa represents the number of control-flow events, and the ordinate represents the overhead. $R^2 = 0.9978 > 0.99$ indicates that it is linearly related. The linear relationship is shown as follows:

$$\text{Overhead} = \sum_{i=1}^n \text{overhead} = 0.0044 * \text{cfe_num}_i + 0.0833. \quad (10)$$

Among them, cfe_num_i is the number of control-flow events of the i -th function. When the function is a core function, the number of control-flow events included is the number of basic block nodes within the function. When the function is a noncore function, the number of control-flow events is 1. The granularity division based on the NSGA-II is described in detail as follows:

Step 1: first, the random population P_t is initialized. P_t is composed of N individuals. Each individual represents a function marking scheme, and each function is represented by a gene. Assuming that there are n functions in the program to be verified, each individual is composed of n genes, and the individual $p = (f_1, f_2, f_3, \dots, f_n)$, where $f_i \in \{0, 1\}$, $1 \leq i \leq n$. When $f_i = 0$, it means that the i -th function is marked as a noncore function. And when $f_i = 1$, it means that the i -th function is marked as a core function.

Step 2: the parent P_t generates a child Q_t of size N through selection, crossover, and mutation operators and merges the parent and child into a R_t of size $2N$.

Step 3: we perform nondominated sorting on R_t to obtain the nondominated sequence of each function

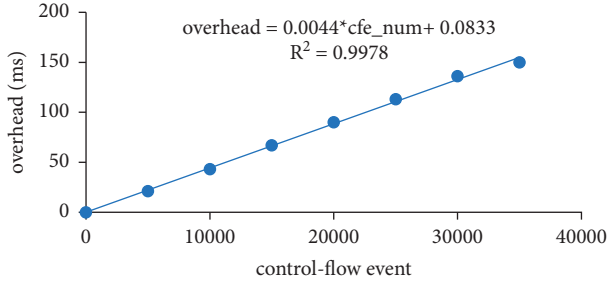


FIGURE 6: Relationship between overhead and control-flow events.

marking scheme. In this paper, p_{rank} is used to represent the nondominated order of individual p . p_{rank} is determined by the nondominated order of the individual in the entire population. The dominating rule is $\widehat{\text{security}}_p \geq \widehat{\text{security}}_q$ and $\widehat{\text{overhead}}_p \leq \widehat{\text{overhead}}_q$; that is, if the security of plan p is not less than plan q and the cost of plan p is not higher than plan q , it means plan p dominates plan q . We use the min-max standardization method to standardize security and overhead as follows:

$$\widehat{\text{Security}}_p = \frac{\text{security}_p - \min_{\forall p \in P} \text{security}}{\max_{\forall p \in P} \text{security} - \min_{\forall p \in P} \text{security}}, \quad (11)$$

$$\widehat{\text{Overhead}}_p = \frac{\text{overhead}_p - \min_{\forall p \in P} \text{overhead}}{\max_{\forall p \in P} \text{overhead} - \min_{\forall p \in P} \text{overhead}}. \quad (12)$$

Count the dominance set and the number of dominated plans for each plan according to the dominance rules. If the number of dominated plans for plan p is 0, then $p_{\text{rank}} = 0$, and add this plan to the first dominance level Rank_0 , while continuing to dominate ranking the remaining plans until the entire population is stratified.

Step 4: we sort the congestion degree of Rt and use p_{distance} to represent the congestion degree of plan p .

Step 5: when the sorting rule ($p_{\text{rank}} < q_{\text{rank}}$) or ($(p_{\text{rank}} = q_{\text{rank}})$ and ($p_{\text{distance}} < q_{\text{distance}}$)) is satisfied, it means that plan p is better than plan q . We select the first N better individuals to form a new generation population P_{t+1} for the next iteration.

Step 6: when judging whether the predetermined number of iterations G is reached, output the first-ranked optimal function marking scheme if it is reached; otherwise, proceed to Step 2.

Table 1 describes the optimization process.

7. Runtime Control-Flow Attestation Protection for Control-Flow Measurements Based on SGX

Intel SGX [17, 18] (Intel Software Guard Extensions) is an extension of the Intel CPU, which encapsulates the programs

that need to be protected in the enclave. All privileged or nonprivileged software cannot access the content in the enclave and can be used to protect applications. The key codes and data are not tampered with by malicious software or high-level system management software (such as OS, VMM, and BIOS). The root of trust of SGX only includes the CPU, which greatly improves the system's security. Therefore, this paper uses Intel SGX to provide a trusted execution environment for the measurement of the program.

As shown in Figure 7, the prover is divided into safe area and unsafe area. The runtime tracking part of the program is placed in the unsafe area, and the measurement and attestation part are placed in the safe area (SGX enclave), ensuring the secure storage of the attestation report and secret key information.

When the prover receives the verification request from the verifier, it runs the program to be verified according to the specified input. When the function call is executed, it will jump to the coarse-grained interceptor; when the jump instruction inside the core function is executed, it will jump to the fine-grained interceptor and then send the intercepted address information to the node inspection module. Nodes are distinguished and then sent to the measurement module to generate the accumulated hash value of each block. The attestation module uses enclave's private key to sign the measurement result and random number, generates an attestation report, and sends it to the verifier.

All key modules are introduced in the following.

7.1. Runtime Tracing. This part is used to track the verified application. This paper rewrites the pin tool in Intel Pin-3.15 and makes it the tool for detecting control-flow events when the program is running. In order to obtain the function call relationship in the program, the program needs to be instrumented at the function level. The instrumentation points are mainly in the call and ret instructions. In order to obtain the internal control-flow information of the core function, it is necessary to perform basic block-level instrumentation to monitor indirect jump statements.

7.2. Coarse-Grained Interceptor. During program execution, when a function call instruction is encountered, GACFA will transfer the program control flow to the coarse-grained interceptor. The interceptor obtains the address of the function at this time, the address of the called function, and the node type of what we insert. Then, through the SGX *ECALL* instruction, it switches to the SGX enclave, passes the acquired address information to the node inspection module in the safe area, and then returns the program control flow to the program execution.

7.3. Fine-Grained Interceptor. When the core function of the program is executed, the fine-grained interceptor intercepts branch information, obtains the first address of the basic block and the node ID we inserted, and sends the information to the node inspection module in the security zone.

TABLE 1: Granularity division algorithm based on NSGA-II.

```

Input:  $F(A)$ ,  $A.exe$ ,  $A\_inputs[ ]$ 
//The function collection of the program, binary file, and input list
Output: Res
//Output optimal granularity division plan
{
  // $F(A)$  is the set of all functions of the program
   $F(A).count = 0$ ; //the number of functions of the program  $A$ 
   $F(A).size[ ] = 0$ ; //the number of control-flow events of each function
  static_func ( $F(A)$ ,  $A.exe$ ,  $inputs[ ]$ ,  $F(A).count$ ,  $F(A).size[ ]$ );
  //Count the number of function calls and control-flow events in each function
  GranularityDivide ( $F(A).count$ ,  $F(A).size[ ]$ , Res);
  //Granularity division, return the optimal granularity division plan
}
GranularityDivide (count, size[ ], best_scheme)
{
   $g = 0$ ;
  InitPopulation ( $Pt$ :  $N$ );
  //The population  $Pt$  is initialized, containing  $N$  individuals representing a granularity division plan each
  while  $g \leq G$  do:
    CreateOffspring ( $Pt$ ,  $Qt$ );
    //The crossover and mutation on  $Pt$  and  $Qt$  produce offspring
     $Rt = Merge (Pt, Qt)$ ;
    for  $p$  in  $Rt$  do:
      //Objective calculation
      security =  $f\_security (count, size[ ], p)$ ;
      //Calculate the security benefits of each individual  $p$  according to formulas (8) and (9)
      overhead =  $f\_overhead (count, size[ ], p)$ ;
      //Calculate the overhead costs of each individual  $p$  according to formula (10)
    end for
    NonDominatedSorting( $Rt$ );
    //Non-dominated sorting based on objective function according to formulas (11) and (12)
    CrowdingDistanceSorting( $Rt$ );
    //Sort the congestion degree of  $Rt$ 
     $Pt = Rt[0, N]$ ;
    //Select the top  $N$  better solutions
     $g = g + 1$ ; //go to the next generation, trying to get better granularity division plan
  end while
  best_scheme =  $Pt[0]$ ;
  //Return to the optimal partition plan
}

```

7.4. Node Check Module. This module judges the node type at this time. If the type changes, it initializes the measurement result and restarts accumulating hash. If a virtual node is detected, the number of cycles needs to be recorded and sent to the measurement module.

7.5. Measurement Module. In order to ensure the security of the measurement process, we perform the SHA256 hash calculation in the enclave container and send the metric value to the attestation module.

7.6. Attestation Module. Use the SGX function to create public and private key pair required for signing the random number and measurement to generate the attestation report and send the attestation report to the verifier.

8. Evaluation

The following is an evaluation and analysis of our proposal GACFA from three aspects: functionality, overhead, and security.

8.1. Functional Evaluation. We use the SNU real-time benchmark [24] to test the effectiveness of GACFA, which is dedicated to the performance evaluation of C/C++ programs. GACFA aims to achieve a combination of coarse- and fine-grained attestation methods, so we use multi-objective optimization algorithms for function marking to determine the Control-Flow Attestation scheme.

8.1.1. Multiobjective Optimization of NSGA-II. In order to prove the effect of NSGA-II optimization, we selected the program “*adpcm-test*” for the analysis, which contains the

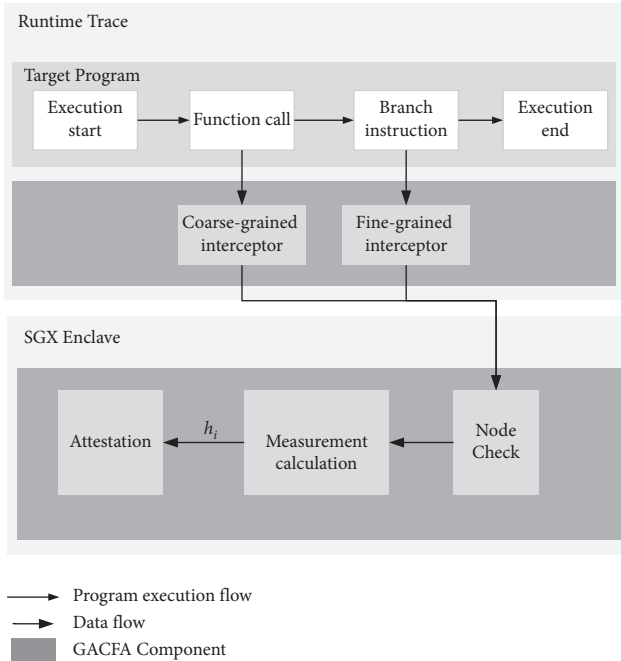


FIGURE 7: Runtime protection on program-end (the prover) based on the SGX.

most functions (15 functions) in the SNU test set. The program’s information and test parameters are listed in Table 2. We get the optimal function marking scheme according to the algorithm described in Section 6. The result is shown in the last column of Table 2, in which functions marked as “1” are chosen to be measured in the granularity of the basic block, while functions marked as “0” are measured as one piece.

Figure 8 shows the optimizing marking process of program “*adpcm-test*.” The blue/red line shows the change trend of the security/overhead goal of each generation (with different granularity division schemes). As we can see, generation 67 is the optimal solution after convergence with “010100110000111” as its gene (shown as the last column in Table 2).

8.1.2. Control-Flow Attestation. The verifier sends an attestation request to the prover (in our case, an embedded industrial control computer) to verify the security of the industrial control program. Take the program simulating user verification process, described in Figure 5 in Section 5, as an example. The program includes two paths. When the input is 1, it is an authorized path, and with the other input, it is an unauthorized path.

The verifier can perform measurement on both paths in advance to get the expected measurement values shown in Figure 9.

Figure 10 shows a successful attestation process. As Figure 10(a) shows, after receiving the attestation request, the control-flow measurements are gotten from the running program and the report and signature are generated and sent to the verifier-end. As Figure 10(b)

TABLE 2: Parameters and optimal function marking solution of program “*adpcm-test*.”

Function name	Number of calls	Number of dangerous function calls	Number of control-flow events	Optimized marking solution
Abs	0	0	5	0
logsch	2000	4	7	1
invqah	0	0	2	0
Uppol1	4000	2	10	1
Uppol2	4000	2	13	0
upzero	4000	3	11	0
scalel	4000	2	2	1
logsc1	2000	5	7	1
invqx1	0	0	2	0
encode	1000	2	37	0
decode	1000	1	24	0
reset	1	0	10	0
fabs	15729	4	5	1
filtez	4000	4	4	1
filtep	4000	5	2	1

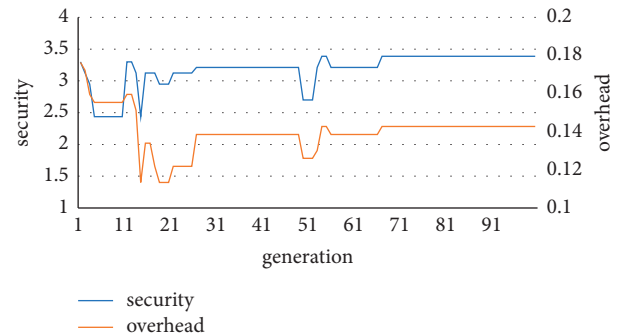


FIGURE 8: The optimizing marking process of program “*adpcm-test*”.

indicates, the verifier verifies the data sent by the program (the prover-end) with the measurement database. Since there is no Control-Flow Hijacking attack, the attestation is successful.

8.1.3. Attack Detection. We launch all 4 kinds of CFH attacks (Attacks 1–4) described in Section 3 to the program “*adpcm-test*” (containing 15 functions) and test three different detection methods for comparative attack detection experiments, namely, Fine-Grained Control-Flow Attestation [6] (noted as FGCFA, with the granularity of basic block), GACFA (our proposal, with the optimal function marking solution shown in Table 1), and Coarse-Grained Control-Flow Attestation [25] (noted as CGCFA, with the granularity of the function).

In the real world, the vulnerability may be found in any function. Therefore, we use the potential threat ratio of 1 : 1 (7 functions without attack and 8 functions with attacks), 1 : 2 (5 functions without attack and 10 functions with attacks), and 1 : 4 (3 functions without attack and 12 functions with attacks) to represent the ratio of the functions without

```

-----
path1:
F1-F2 : 799e4183ddcb2b2a91e6589474892d092021f573a041064adb95a85119244d8
B3-B5 : c0f220e500ef58ba20370510f57889d8ea166b3f0459bf0481a8bf79c054e6e1
L : 8fea18e0b61af4b0cbe3eca30f2de14e32e21669db0c0d0bde558b040858927a
-----
path2:
F1-F2 : 799e4183ddcb2b2a91e6589474892d092021f573a041064adb95a85119244d8
B3-B8 : 2e29502cd2efa1d106b2d0ccf2c29b2a33ecd39368b0035f95700f3c467de621

```

FIGURE 9: Expected measurements of two execution paths.

vulnerability to the functions with vulnerability. That is, the function potential threat ratio is 53%, 67%, and 80%. We conducted 3 rounds of random attack tests on the program, one round for each ratio. The average detection rate results are shown in Figure 11.

In Figure 11, the FGCFA can detect all attacks, so the detection rate is 1 under different ratios. The CGCFA detection capability is related to the simulated attack category, but it does not consider the risk function is more vulnerable to attack, so the detection rate under different proportions is low. With the increase in the number of attacks on threat functions, the detection capability of the GACFA solution increases. At the potential threat ratio of 53.3%, the detection rate is 0.7, which is 30% lower than that of the FGCFA solution and is 2.5 times the CGCFA solution. The detection rate of this scheme is 24% lower than that of the FGCFA solution and 2.69 times the CGCFA solution. Therefore, as the probability of being attacked by threat functions increases, the detection capability of the scheme in this paper gradually approaches the FGCFA solution, but the performance overhead remains unchanged.

8.1.4. Runtime Control-Flow Attestation Protection with the SGX. In this section, we discuss the security of GACFA. The security requirement of GACFA is to verify the integrity of the control flow of the program to be verified on the prover to ensure that the measurement results are not tampered with.

The online attestation module includes the program’s runtime tracking component and measurement attestation component. The runtime tracking component is a binary file after the pin tool has been instrumented. Like C-FLAT [6], we believe that the static measurement includes the target program and runtime tracking, and the integrity of this part will not be tampered with. GACFA places the measurement and authentication components of control-flow information in the SGX enclave and uses the program isolation mechanism of the SGX enclave to ensure the validity of the authentication report.

In order to verify the security of the SGX enclave protected measurement and attestation at the program-end, this paper attempts to tamper with the control-flow hash value not protected and protected by the SGX enclave. The measured program is still the one shown in Figure 5. Figure 12(a) shows the accumulated hash calculation process of path 1 of the program. Then, we shut down the SGX and

tamper measurement code from memory. Figure 12(b) shows the 3rd part’s hash value has changed, and the tampering is successful. However, after placing the measurement program in the SGX enclave, trying to access measurement code outside the enclave is prohibited, and the tampering on final measurement failed, which is shown in Figure 12(c).

According to Figure 12, we can find that by introducing SGX, GACFA can reliably verify the integrity of the control flow of the program and ensure that the measurement results and attestation report from being tampered with.

8.2. Performance Analysis. Offline program analysis is performed by the verifier and does not affect the execution process of the program, so we only consider the performance overhead of the online attestation phase. The runtime program attestation time includes the control-flow information collection time t_{col} , the measurement time t_{mea} , and the signing time t_{sig} , so the total runtime attestation time is shown as follows:

$$t_{total} = t_{col} + t_{mea} + t_{sig}. \quad (13)$$

We have implemented the FGCFA, GACFA, and CGCFA for the *adpcm-test* program. In particular, for our proposal GACFA, we perform instrumentation according to the optimal function marking scheme 010100110000111 given in Section 8.1, of which 8 functions perform coarse-grained instrumentation and 7 functions perform fine-grained instrumentation. Other procedures, such as measurement and attestation, in all three methods are the same.

Figure 13 shows the time of online attestation for the three methods. Although our time cost is increased by 31.79% compared with the CGCFA, which cannot detect any basic block-level CFH attacks, our time cost is reduced by 56.99% compared with FGCFA. This is because the necessary time of control-flow collection and measurement is greatly reduced compared to the FGCFA.

Figure 14 shows the comparison of the measurement time for several different programs in the SNU data set. Although the average size of SNU programs is small, we can still see that, for different programs, the measurement time cost of our proposal is much lower than the FGCFA.

8.3. Balance of Security and Overhead. In order to evaluate the security and overhead balance capability of different Control-Flow Attestation schemes, we propose the concept of security/performance benefit ratio for different schemes as $spbr_{x/y}$ (x, y are different schemes), that is, the detection rate change ratio/detection time change ratio. Apparently, when $spbr_{x/y}$ is bigger, scheme x is better than y . The calculation method of $spbr_{x/y}$ is shown as follows:

$$spbr_{x/y} = \frac{\text{detection}_{ratex} - \text{detection}_{ratey} / \text{detection}_{ratey}}{\text{detection}_{timex} - \text{detection}_{timey} / \text{detection}_{timey}}. \quad (14)$$

$spbr_{FGCFA/GACFA}$ and $spbr_{GACFA/CGCFA}$ are shown in Table 3. It can be seen from the table that $spbr_{GACFA/CGCFA}$ is

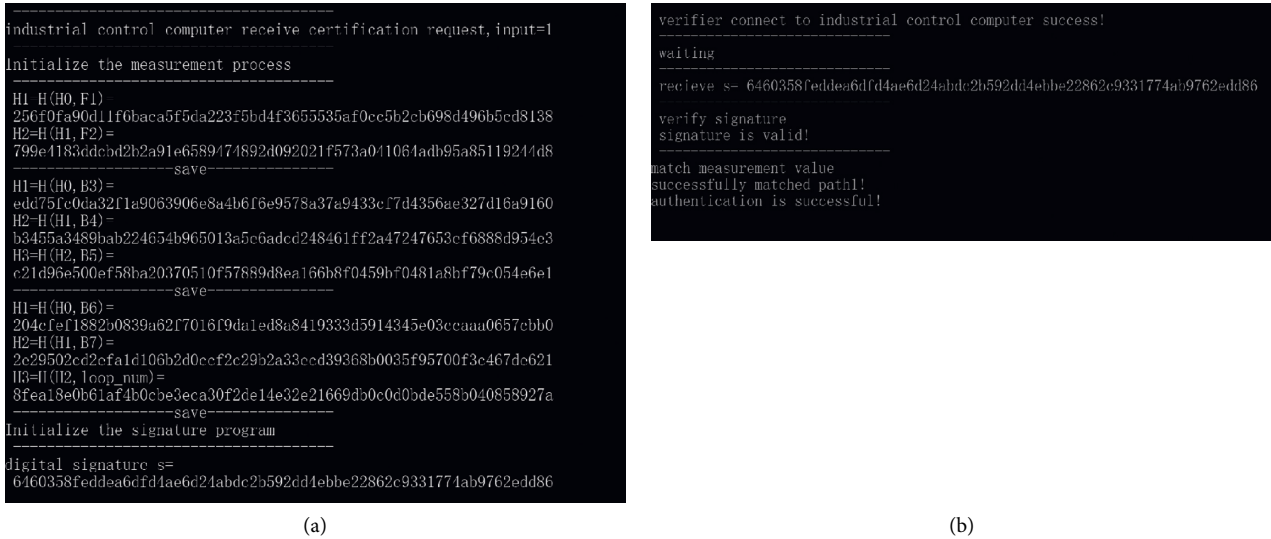


FIGURE 10: The prover-end and the verifier-end results in attestation: (a) the prover-end results and (b) the verifier-end results with no control-flow tampering attack.

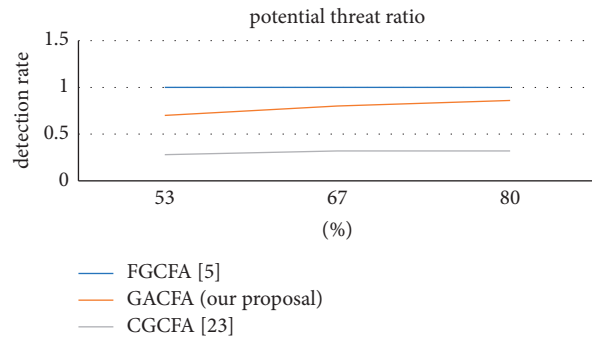


FIGURE 11: Comparison of attack detection capabilities under different attack scenarios.

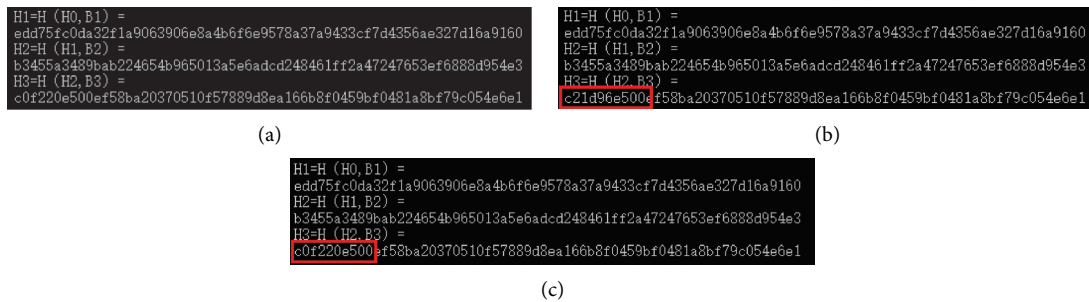


FIGURE 12: SGX antitampering results. (a) Expected path and final hash result (H3). (b) Tampering (H3) succeeded. (c) Tampering (H3) failed.

the highest when the CGCFA is changed to our scheme (GACFA), and $spbr_{GACFA/CGCFA}$ is 13.7, 25.1, and 43.0 times that of $spbr_{FGCFA/GACFA}$ in the case of three different potential threat ratios (specified in Section 8.1). This means changing from CGCFA to our scheme can achieve a much better balance of security and overhead. That is, higher security can be obtained with less performance overhead in our scheme.

9. Discussion

We propose GACFA for CFH detection of embedded programs to detect control data and noncontrol data attacks that indirectly affect control flow. To ensure the security of attestation and measurement, we put the modules into the SGX enclave, so that measurement calculation and attestation report signing and generation will not be tampered

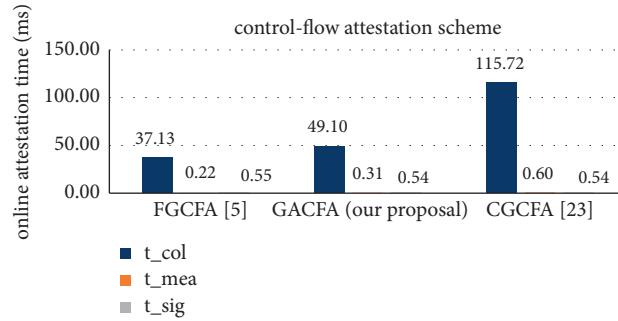


FIGURE 13: Comparison of online attestation time on different schemes.

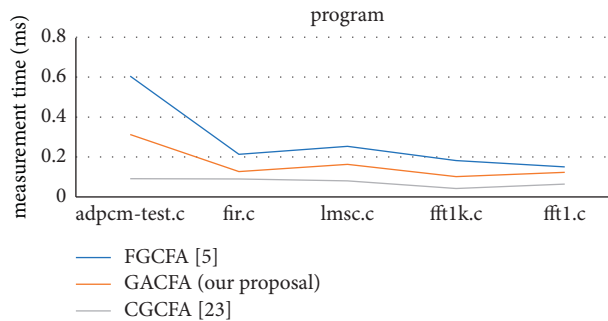


FIGURE 14: Comparison of measurement time on different schemes with different programs.

TABLE 3: Security/performance balance capability comparison in different threat scenarios.

Potential threat ratio (%)	$spbr_{FGCFE/GACFA}$	$spbr_{GACFA/CGCFE}$
53	0.32	4.69
67	0.18	4.69
80	0.12	5.27

with by privileged attackers. However, the SGX is proposed to protect applications. But the Intel pin tools are used to take care of the process of monitoring and extracting control flows of the running program, which cannot be protected by the SGX because the tools rely on the kernel. This means if the pin tools are tampered with, the control-flow information we get and use in the SGX enclave cannot be trusted. Hardware-based method, such as LO-FAT [16], ATRIUM [7], seems to be promising to solve this problem. LO-FAT [16] extends branch tracking of processor pipeline with additional logic to achieve efficient tracing of control-flow information, along with measurement calculation and attestation, and implement the proof-of-concept system based on a RISC-V SoC. ATRIUM [7] is tightly integrated with a processor and can extract the executed instructions and memory addresses and generate the final measurement and attestation report. Of course, a hardware-based method means extra hardware support and costs.

Besides, our proposal can only detect CFH attacks based on program execution-related data (e.g., jump address, branch, and loop variables), not pure data attacks (data-oriented programming attacks, DOP attacks) [26]. The DOP attacks manipulate data in the program and

tamper with key variables but not control data and noncontrol data mentioned above, which means the current control-flow detection method may be bypassed. Hu et al. [27] proposed several new solutions, such as data flow integrity protection and fine data flow randomization. However, these solutions on data flow cost much more than CFI methods and need more research before they are used in practice.

10. Conclusions

Control-Flow Hijacking attacks have become the main method of exploiting vulnerabilities, seriously threatening the security of industrial control systems. This paper proposes a NSGA-II based Granularity-Adaptive Control-Flow Attestation (GACFA) for CFH detection of embedded programs, which can be used to detect control data attacks and noncontrol data attacks that indirectly affect control flow. A control-flow representation model is used to reduce the complexity of programs' control-flow graph, while the NSGA-II algorithm is used in offline analysis to optimize the granularity division strategy to maximize system security benefits and minimize overhead. Besides, runtime

protection for the GACFA at the program-end with SGX is proposed to protect the integrity and confidentiality of control-flow measurement data. Experiments have shown that the algorithm can obtain a relatively balanced coarse and fine granularity between control-flow security and runtime efficiency. In addition, we evaluated the performance and security of the runtime module and found that compared with the current fine-grained Control-Flow Attestation method, the security/performance benefit of adopting our proposal over CGCFA is 13.7, 25.1, and 43.0 times that of adopting FGCFA over ours in different threat scenarios.

Data Availability

All data are available from the corresponding author upon request.

Conflicts of Interest

The authors have no conflicts of interest.

Acknowledgments

This work was supported by grants from the National Key Research and Development Program of China (Grant no. 2016YFB0800204) and the Open Research Fund of Beijing Key Laboratory of Trusted Computing.

References

- [1] H. Ke, H. Wu, and Y. Dongmei, "Towards evolving security requirements of industrial internet: a layered security architecture solution based on data transfer techniques," pp. 504–511, Association for Computing Machinery, Beijing China, December 2020.
- [2] N. Jr, T. Fraser, J. Molina, and W. A. Arbaugh, "Copilot - a coprocessor-based kernel runtime integrity monitor," in *Proceedings of the 13th USENIX Security Symposium*, pp. 179–194, USENIX Association, San Jose, CA, USA, August 2004.
- [3] A. Seshadri, A. Perrig, L. Doorn, and P. Khosla, "SWATT: Software-based ATtestation for embedded devices," in *Proceedings of the IEEE Symposium On Security And Privacy*, pp. 272–282, IEEE, Berkeley, CA, USA, May 2004.
- [4] Z. Chao, W. Tao, and Z. Chen, "Practical control flow integrity and randomization for binary executables," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP)*, pp. 559–573, IEEE, Berkeley, CA, USA, May 2013.
- [5] N. Carlini, A. Barresi, M. Payer, D. Wagner, and T. R. Gross, "Control-flow bending: on the effectiveness of control-flow integrity," in *Proceedings of the 24th USENIX Security Symposium*, pp. 161–176, USENIX, Berkeley, CA, USA, August 2015.
- [6] T. Abera, N. Asokan, and L. Davi, "C-flat: control-flow attestation for embedded systems software," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security ACM*, pp. 743–754, ACM, Darmstadt, Germany, October 2016.
- [7] S. Zeitouni, G. Dessouky, O. Arias, D. Sullivan, and A. R. Sadeghi, "Atrium: runtime attestation resilient under memory attacks," in *Proceedings of the 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 384–391, IEEE, Irvine, CA, USA, November 2017.
- [8] X. Ge, W. Cui, and T. Jaeger, "Guarding control flows using intel processor trace," in *Proceedings of the the Twenty-Second International Conference*, pp. 585–598, ACM, Redmond, WA, USA, Apr 2017.
- [9] P. F. Qiu, Y. Q. Lyu, J. Zhang, D. Wang, and G. Qu, "Control flow integrity based on lightweight encryption architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 1358–1369, 2017.
- [10] N. Burow, S. A. Carr, J. Nash et al., "Control-flow integrity," *ACM Computing Surveys*, vol. 50, no. 1, pp. 1–33, 2017.
- [11] R. Ding, C. Qian, C. Song, B. Harris, and W. Lee, "Efficient protection of path-sensitive control security," in *Proceedings of the 26th USENIX Security Symposium*, pp. 131–148, USENIX, Vancouver, BC, Canada, August 2017.
- [12] D. Jung, M. Kim, J. Jang, and B. B. Kang, "Value-based constraint control flow integrity," *IEEE Access*, vol. 8, Article ID 50542, 2020.
- [13] MR. Khandaker, W. Liu, A. Naser, Z. Wang, and J. Yang, "Origin-sensitive control flow integrity," in *Proceedings of the 28th USENIX Security Symposium*, pp. 195–211, USENIX, Santa Clara, CA, USA, August 2019.
- [14] M. Khandaker, A. Naser, W. Liu, Z. Wang, Y. Zhou, and Y. Cheng, "Adaptive call-site sensitive control flow integrity," in *Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroSP)*, pp. 95–110, IEEE, Stockholm, Sweden, June 2019.
- [15] H. Jang, M. Park, and H. Dong, "IBV-CFI: efficient fine-grained control-flow integrity preserving CFG precision," *Computers & Security*, vol. 94, 2020.
- [16] G. Dessouky, S. Zeitouni, T. Nyman et al., "LO-FAT: low-overhead control flow ATtestation in hardware," in *Proceedings of the 54th Annual Design Automation Conference 2017*, pp. 1–6, IEEE, Austin, TX, USA, June 2017.
- [17] I. Corporation, *Intelsoftware Guard Extensions Programming Reference*, Intel, San Jose, CA, USA, 2014.
- [18] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU based attestation and sealing," in *Proceedings of the Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*, pp. 1–6, Tel-Aviv, Israel, June 2013.
- [19] F. Schuster, M. Costa, C. Fournet et al., "VC3: trustworthy data analytics in the cloud using SGX," in *Proceedings of the IEEE Symposium on Security and Privacy SP*, pp. 38–54, San Jose, CA, USA, 2015.
- [20] F. Shaon, M. Kantarcioglu, Z. Lin, and K. Latifur, "SGX-BigMatrix: a practical encrypted data analytic framework with trusted processors," in *Proceedings of the 2017 ACM SIGSAC Conference*, pp. 1211–1228, ACM, Dallas, TX, USA, October 2017.
- [21] J. Wang, H. Zhi, and Y. Zhang, "Enabling security-enhanced attestation with intel SGX for remote terminal and IoT," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 88–96, 2017.
- [22] V. Karande, E. Bauman, Z. Lin, and K. Latifur, "SGX-log: securing system logs with SGX," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS)*, pp. 19–30, ACM, New York, NY, United States, April 2017.
- [23] C. Khammassi and S. Krichen, "A nsga2-lr wrapper approach for feature selection in network intrusion detection," *Computer Networks*, vol. 172, 2020.

- [24] “SNU real-time benchmarks,” http://www.cprover.org/satabs/examples/SNU_Real_Time_Benchmarks/.
- [25] Y. Yu, P. Wang, Y. Zhang, H. Zhang, and H. Zhang, “Detection of control flow attacks based on return address signature,” *Journal of East China University of Science and Technology*, vol. 46, pp. 800–806, 2020.
- [26] S. Chen, J. Xu, E. C. Sezer, P. Gauriar, and R. K. Iyer, “Non-control-data attacks are realistic threats,” in *Proceedings of the 14th Unix Security Symposium*, Baltimore, MD, USA, July 2005.
- [27] H. Hu, S. Shinde, S. Adrian, Z. L. Chua, P. Saxena, and Z. Liang, “Data-oriented programming: on the expressiveness of non-control data attacks,” in *Proceedings of the 37th IEEE Symposium on Security and Privacy*, pp. 969–986, IEEE, San Jose, CA, USA, May 2016.

Research Article

A Blockchain-Based CP-ABE Scheme with Partially Hidden Access Structures

Yang Ba,¹ Xuexian Hu ,¹ Yue Chen,¹ Zenghang Hao,¹ Xuewei Li,² and Xincheng Yan³

¹State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China

²Kunming Audit Center, Kunming 650001, China

³State Key Laboratory of Space Medicine Fundamentals and Application, Beijing 100094, China

Correspondence should be addressed to Xuexian Hu; xuexian_hu@hotmail.com

Received 27 August 2021; Accepted 12 October 2021; Published 8 November 2021

Academic Editor: Qi Jiang

Copyright © 2021 Yang Ba et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Data sharing has become a key technology to break down data silos in the big data era. Ciphertext-policy attribute-based encryption (CP-ABE) is widely used in secure data-sharing schemes to realize flexible and fine-grained access control. However, in traditional CP-ABE schemes, the access structure is directly shared along with the ciphertext, potentially leading to users' private information leakage. Outsourcing data to a centralized third party can easily result in privacy leakage and single-point bottlenecks, and the lack of transparency in data storage and sharing casts doubts whether users' data are safe. To address these issues, we propose a blockchain-based CP-ABE scheme with partially hidden access structures (BCP-ABE-PHAS) to achieve fine-grained access control while ensuring user privacy. First, we propose an efficient CP-ABE scheme with partially hidden access structures, where the ciphertext size is constant. To assist data decryption, we design a garbled Bloom filter to help users quickly locate the position of wildcards in the access structure. Then, to improve storage efficiency and system scalability, we propose a data storage scheme that combines blockchain technology and the interplanetary file system, ensuring data integrity. Finally, we employ smart contracts for a transparent data storage and sharing process without third-party participation. Security analysis and performance evaluation show that the proposed BCP-ABE-PHAS scheme can preserve policy privacy with efficient storage and low computational overhead.

1. Introduction

Cloud computing promotes the aggregation of storage and computational resources and has a tremendous market value. However, when data owners outsource data to cloud services, they lose control of their data, and their private information is at risk of leakage [1]. Recently, data security incidents have occurred frequently, and such events undermine users' confidence in data security and raise concerns regarding cloud storage.

In 2005, Sahai and Waters [2] proposed attribute-based encryption (ABE) to achieve fine-grained access control. The ABE scheme is mainly categorized into ciphertext-policy ABE (CP-ABE) [3] and key-policy ABE (KP-ABE) [4]. In the KP-ABE scheme, the secret key and ciphertext are associated with the access structure (or access policy) and attribute set,

respectively. In this case, the ciphertext can only be decrypted when the attribute set satisfies the access policy. Contrarily, in the CP-ABE scheme, the ciphertext and secret key are associated with the access policy and attribute set, respectively.

The CP-ABE scheme features fine-grained access control and one-to-many secure data sharing. However, in the traditional CP-ABE scheme, the access policy is directly shared along with the ciphertext. Consequently, anyone can get this access policy while obtaining the ciphertext; however, the access policy may contain the user's sensitive information.

Consider a scenario in which a patient with a social security number (SSN) 123-456-789 wants to outsource his (or her) health data to the cloud and establish an access policy, as shown in Figure 1(a). This patient designs an

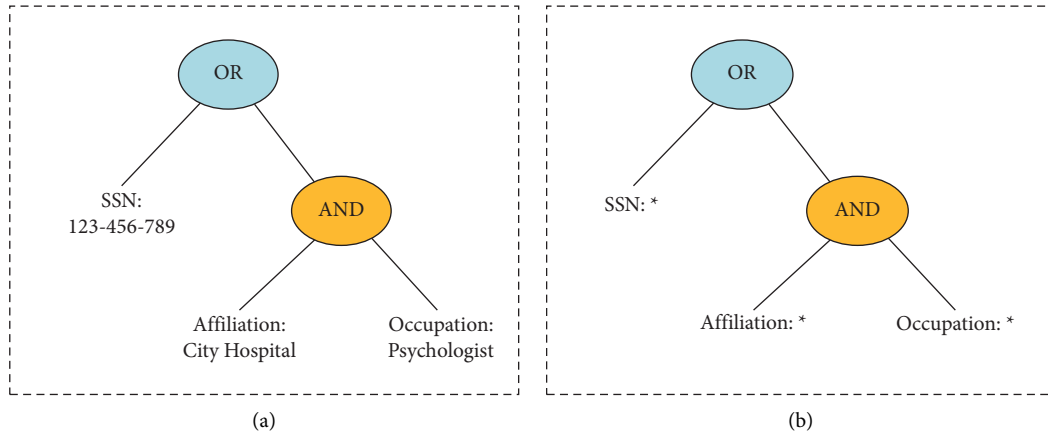


FIGURE 1: (a) Access structure. (b) Partially hidden access structure.

access policy based on which only this patient or the psychologist at the city hospital can access the data. If the patient uses the traditional CP-ABE scheme to send the encrypted data and access policy to the cloud, anyone with access to this cloud can obtain the patient's access policy. Thus, the data security of this patient, who is suffering from psychological problems, is undoubtedly threatened.

The most effective way to protect a user's access policy information is to hide the attribute information. Policy hiding involves fully and partially hiding. In the CP-ABE scheme, fully hidden access policies imply that no attribute information in the access policy is revealed, and partially hidden access policies imply that only sensitive attribute values are hidden. As shown in Figure 1(b), the partially hidden access policy is expressed as (SSN: * OR (Affiliation: * AND Occupation: *)), where the attribute values that may expose a user's information are hidden. A tradeoff is obtained between the efficiency of the CP-ABE scheme and the fully hidden access structure using a partially hidden access structure embedded in the CP-ABE scheme to reduce computational costs [5].

Additionally, centralized storage architectures are vulnerable to various network attacks such as single point of attack, man-in-the-middle attack, and distributed denial-of-service attack [6, 7]. Owing to such attacks, data owners may lose control of their data. Because blockchain technology is transparent, decentralized, and unforgeable, blockchain-based data storage and sharing schemes have been proposed to resist such attacks. Blockchain is an append-only distributed database, so large-scale data can quickly bloat the blockchain and make it expensive and inefficient to scale. To alleviate the storage pressure of the blockchain, we propose a storage scheme that combines blockchain technology and the interplanetary file system (IPFS) [8].

Therefore, we propose a blockchain-based CP-ABE scheme with a partially hidden access structure (BCP-ABE-PHAS) to realize secure data storage and sharing. Our main contributions are summarized as follows:

- (1) We propose a CP-ABE scheme with partially hidden access structures to achieve fine-grained access control and ensure user privacy. Moreover, to assist

data decryption, we design a garbled Bloom filter (GBF) to locate the position of wildcards in the access policy.

- (2) To ensure data integrity and improve system scalability, we adopt a storage scheme that combines blockchain technology and the IPFS, in which the real ciphertext is stored in the IPFS, and meanwhile, the access policy is stored on the blockchain.
- (3) We employ smart contracts to achieve automated and trusted access control, where the entire data storage and sharing process is transparent without third-party participation.
- (4) Security analysis and performance evaluation show that the proposed scheme can achieve effective privacy preservation without incurring considerable overhead.

The remainder of this paper is organized as follows. In Section 2, we introduce the related work. In Section 3, preliminaries are described. We then present the system architecture and security model in Section 4, followed by the detailed construction of the proposed scheme in Section 5. The security analysis and the performance evaluation are performed in Section 6, and the conclusions are presented in Section 7.

2. Related Work

Bethencourt et al. [3] proposed the first CP-ABE scheme. This scheme allows data owners to specify a fine-grained access policy for their data to realize secure data sharing. However, an access policy may contain a user's sensitive information which is attached to the ciphertext as a plaintext, causing privacy leakage [9].

To address this problem, some schemes that hide the access policy have been proposed. For example, Nishide et al. [10] proposed a CP-ABE scheme with hidden access policies. They proposed two schemes in which only attribute values are hidden using AND gates on multivalued attributes with wildcards. Based on this scheme [10], Li et al. [11] implemented user accountability while hiding the access policy.

Phuong et al. [12] proposed two CP-ABE schemes with a hidden access policy. In this case, the access structure employs AND gates on positive and negative attributes with wildcards, and the ciphertext length is constant. Although these schemes are secure and efficient, AND-based access policies are limited in terms of expressiveness.

Thus, to facilitate a more expressive access policy, Lai et al. [13] proposed a partially hidden CP-ABE scheme that supports linear secret-sharing scheme-based access policy. Based on this scheme [13], Zhang et al. [14] proposed a scheme that can support a large attribute universe. However, these schemes are built using composite-order bilinear groups; thus, their efficiency is low. Katz et al. [15] first proposed the inner-product predicate encryption. However, the “superpolynomial blowup” problem makes the CP-ABE schemes that use the attribute-hiding IPE to construct a fully hidden access policy very inefficient [16]. Hur [17] proposed a CP-ABE scheme that can support any monotonous access policy. In this case, the access policy is hidden by attribute remapping, and most decryption operations are delegated to the cloud storage center to considerably reduce the requester’s computational overhead.

Because blockchain technology is decentralized, tamperproof, and transparent, it is widely used in secure data sharing and access control schemes. Based on inner-product predicate encryption [15], Gao et al. [18] proposed a trustworthy secure CP-ABE scheme with a fully hidden access policy based on blockchain technology. This scheme combines inner-product encryption and homomorphic encryption to hide access policies and uses smart contracts to store the generated proof on the blockchain permanently. Zhang et al. [19] proposed an access control for the Internet of Things (IoT) based on the smart contract which consists of the judge contract, access control contract, and register contract to achieve intelligent and efficient access control. Additionally, Xu et al. [20] proposed a blockchain-based smart healthcare system for large-scale health data privacy preservation. This system uses digital envelope technology to verify the confidentiality of information; however, it can only support one-to-one secure transmission, which does not satisfy the requirements of users who simultaneously employ multiple third parties to provide services. In the IoT environment, Xu et al. [21] and Novo [22] adopted the blockchain technology to realize secure data sharing and access control; however, these schemes do not satisfy large-scale storage and privacy protection requirements.

3. Preliminaries

In this section, we introduce some basic knowledge associated with our BCP-ABE-PHAS.

3.1. Bilinear Map. Let \mathbb{G} and \mathbb{G}_T be multiplicative cyclic groups of prime order q . A bilinear mapping is a function $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ which has the following properties:

- (1) Bilinearity: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_q^*$, there exists $e(u^a, v^b) = e(u, v)^{ab}$

- (2) Nondegeneracy: there exists $g \in \mathbb{G}$ such that $e(g, g) \neq 1$
- (3) Computability: $\forall u, v \in \mathbb{G}$, $e(u, v)$ can be effectively computed

3.2. Blockchain. Blockchain is an append-only data structure in a peer-to-peer network environment, where data blocks are connected chronologically in a chain and the data in a blockchain are assured to be tamperproof, unforgeable, and traceable using cryptography [23]. As shown in Figure 2, a block comprises the block header and block body. The block header consists of four components: (1) PreBkHash, which is the digest of the previous block; (2) TS, which is the timestamp of the block creation; (3) nonce, which is the consensus proof computed by miners and guarantees the consensus of the block; (4) Merkle root, which is the root hash of the Merkle hash tree. The block body stores transaction details.

The concept of smart contracts was first proposed by Szabo [24]. A smart contract is a program that contains code (its function) and data (its state). Smart contracts are used in Ethereum blockchain [25]. The contract address is usually given when a contract is deployed to the blockchain. Contract address is the address to a collection of codes on the blockchain that executes functions. These functions of a contract address are executed when a transaction is made to the contract address. Once a smart contract is deployed in the network, it can run as programmed without human intervention.

3.3. Bloom Filter. The Bloom filter is a space-efficient probabilistic data structure used to determine whether an element is contained in a specific set [26]. The Bloom filter is an m -bit array that can represent a set S of maximum n elements. The Bloom filter has k independent hash functions $H = (h_1, \dots, h_k)$, where $h_i: \{0, 1\}^* \mapsto [1, m]$ and $1 \leq i \leq k$ indicates that the value generated by the hash function is uniformly distributed in $[1, m]$. Herein, a Bloom filter with parameters (m, n, k, H) is represented as $(m, n, k, H)0BF$, a Bloom filter encoding the set S is represented as BF_S , and the value at index i in BF_S is represented as $BF_S[i]$.

First, all bits in the Bloom filter are set to 0. As shown in Figure 3, when we add the element x in the set $S = \{x, y\}$ to the Bloom filter, we set $BF_S[h_i(x)] = 1$ for $1 \leq i \leq 3$. When we verify the existence of an element y in set S , if $BF_S[h_i(y)] = 0$ exists for $1 \leq i \leq 3$, this proves that $y \notin S$; otherwise, $y \in S$ with a high probability.

The Bloom filter yields false positives; in other words, it yields an element that does not belong to the set S , but the corresponding position values are all 1. As shown in Figure 3, the element z does not belong to the set $\{x, y\}$; however, $BF_S[h_i(z)] = 1$ for $1 \leq i \leq 3$. According to Bose et al. [27], the false positive probability is negligible if we select the optimal k and m values.

3.4. Secret Sharing. Secret sharing technology is an important aspect of cryptography research. For example, Shamir [28] proposed a (k, n) -threshold secret-sharing scheme. The

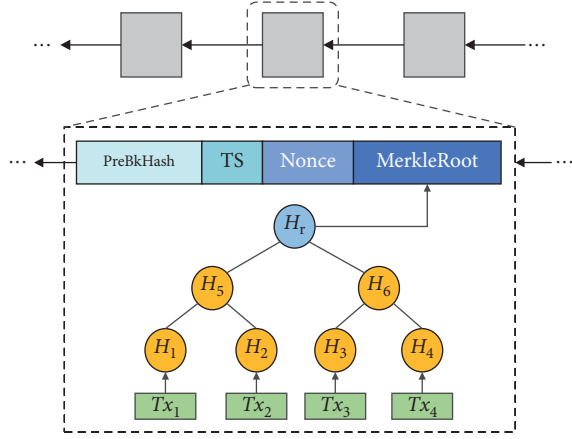


FIGURE 2: Blockchain structure.

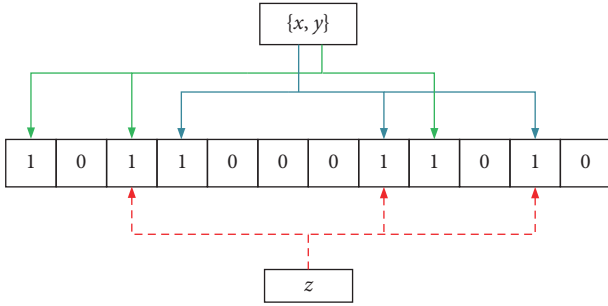


FIGURE 3: Bloom filter.

basic concept of this scheme is that the secret s to be shared is divided and distributed to n participants. The secret can be recovered in the case of minimum k participants; otherwise, the secret cannot be recovered. When $k = n$, secret sharing can be obtained using the \oplus (XOR) operation. Randomly generate $n - 1$ bit strings r_1, \dots, r_{n-1} with the same length as the secret s , and calculate $r_n = r_1 \oplus \dots \oplus r_{n-1} \oplus s$; each of r_i is a part of the secret s . Finally, the secret s can be obtained by computing $r_1 \oplus \dots \oplus r_n$.

3.5. Attribute Vector. As shown in Figure 4, we define two attribute vectors $\bar{W} = (\bar{w}_1, \dots, \bar{w}_L) \in \Sigma_*^L$ and $W = (w_1, \dots, w_L) \in \Sigma^L$, where $\Sigma \subset \mathbb{Z}_q^*$ and $\Sigma_* = \Sigma \cup \{*\}$. Here, \bar{W} contains the wildcard $*$, and $J = \{j_1, \dots, j_n\} \subset \{1, \dots, L\}$ represents the set of wildcard positions in \bar{W} .

The decryption algorithm discussed in this paper employs the following polynomial identity, where w_i is the attribute value at position i in the attribute vector.

$$\sum_{i=1}^L \prod_{j \in J} (i - j) w_i = \sum_{i=1, i \notin J}^L \prod_{j \in J} (i - j) w_i. \quad (1)$$

We use Viète's formulas [29] to construct the polynomial $\prod_{j \in J} (i - j) = x^n + a_{n-1} x^{n-1} + \dots + a_0$ in equation (1), and the coefficients are calculated as follows:

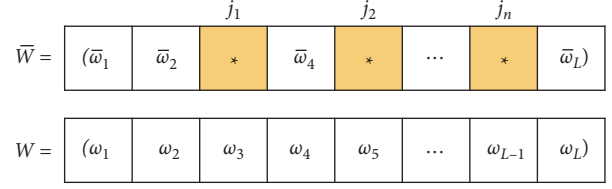


FIGURE 4: Attribute vector.

$$a_{n-k} = (-1)^{i-n} \prod_{1 \leq i_1 < i_2 < \dots < i_k \leq n} j_{i_1} j_{i_2} \dots j_{i_k}, 0 \leq k \leq n, \quad (2)$$

where $n = |J|$. Here, if J is clear, we can calculate the polynomial coefficients a_i . For example, when $J = \{j_1, j_2, j_3\}$, we can construct polynomial $(x - j_1)(x - j_2)(x - j_3)$ and calculate the coefficients:

$$\begin{aligned} a_0 &= -j_1 j_2 j_3, \\ a_1 &= j_1 j_2 + j_1 j_3 + j_2 j_3, \\ a_2 &= -(j_1 + j_2 + j_3), \\ a_3 &= 1. \end{aligned} \quad (3)$$

3.6. Decision Linear Assumption. Let \mathbb{G} be a bilinear group of prime order q with a generator g . For any probabilistic polynomial-time (PPT) adversary \mathcal{A} , its advantage $Adv_{\mathcal{A}}(\lambda)$ in solving the decision linear (DLIN) problem [30] in \mathbb{G} is

$$\begin{aligned} Adv_{\mathcal{A}}(\lambda) &= \left| \Pr \left[\mathcal{A}(g, g^a, g^b, g^{ac}, g^d, g^{b(c+d)} = 1) \right] \right. \\ &\quad \left. - \Pr \left[\mathcal{A}(g, g^a, g^b, g^{ac}, g^d, g^r = 1) \right] \right|, \end{aligned} \quad (4)$$

where the probability is taken over all possible choices of $a, b, c, d, r \in \mathbb{Z}_q^*$. We say that the DLIN assumption holds in \mathbb{G} if there exists a negligible function $\varepsilon(\lambda)$ such that $Adv_{\mathcal{A}}(\lambda) < \varepsilon$ for any PPT algorithm \mathcal{A} .

4. System Architecture and Security Model

4.1. System Architecture. The system architecture of the proposed scheme is shown in Figure 5. As illustrated, the system architecture involves five entities, i.e., attribute authority (AA), IPFS, data owner (DO), data user (DU), and blockchain.

AA: the AA manages all attributes in the system and assigns attributes to users. It is also responsible for generating public parameters and issuing secret keys based on the users' attributes. In this paper, the AA is fully trusted.

IPFS: the IPFS is a distributed file storage system based on content addressing. Note that there is no central server node in the IPFS; thus, it can avoid the risk of a single point of failure. The IPFS uses an encryption algorithm to calculate the hash value $hash_{ipfs}$ of a file, and this $hash_{ipfs}$ is used as the file's address. This approach reduces the repeated storage of files and ensures the integrity of files.

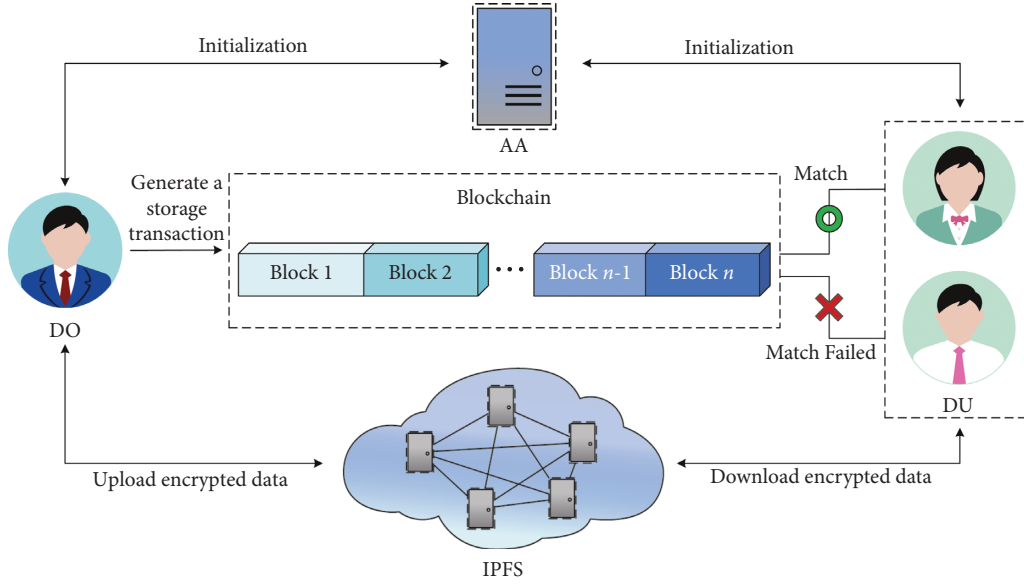


FIGURE 5: System architecture.

DO: the DO selects the *file* to be shared and creates a corresponding access policy. First, the DO encrypts the *file* using the symmetric key $aeskey$ and stores the ciphertext $encfile$ in the IPFS. Then, the proposed CP-ABE scheme is used to encrypt $aeskey$ and generate the ciphertext CT . Finally, $hashipfs$ and CT are stored on the blockchain using a smart contract.

DU: the DU sends a request to the AA, and the AA generates a secret key based on the attribute set of the DU. The DU obtains CT stored on the blockchain using the smart contract and decrypts CT based on its secret key. Here, if the attribute set satisfies the access structure set by the DO, then the DU can obtain the *file* from the IPFS using $aeskey$ and $hashipfs$.

Blockchain: the blockchain is an append-only distributed database, where data are stored permanently and are tamperproof. To ensure secure data sharing and fine-grained access control, the DO only stores $hashipfs$ and CT on the blockchain using smart contracts.

4.2. The Definition of the BCP-ABE-PHAS Scheme. Here, we present the definition of our scheme. This scheme mainly involves the following four algorithms:

- (i) $Setup(1^\lambda)$: the Setup algorithm is executed by the AA. This algorithm takes security parameter 1^λ as the input and outputs the public parameters PK and master secret key MSK .
- (ii) $KeyGen(PK, MSK, W)$: the KeyGen algorithm is executed by the AA. Here, PK , MSK , and W of the DU are taken as inputs, and the secret key SK_W associated with W is the output.
- (iii) $Encrypt(PK, M, (\overline{W}, J))$: the Encrypt algorithm is executed by the DO. This algorithm comprises the BuildGBF and GenCT functions.

$BuildGBF(\overline{W}, J)$: this function takes an access policy \overline{W} and the wildcard position set J as inputs and outputs GBF

$GenCT(PK, M, (\overline{W}, J))$: this function takes PK , a message M , an access policy \overline{W} , and the wildcard position set J as inputs and outputs ciphertext CT

- (iv) $Decrypt(PK, SK_W, GBF, CT)$: the Decrypt algorithm is executed by the DU and comprises the QueryGBF and Dec functions.

$QueryGBF(W, GBF)$: this function takes an attribute vector W of the DU as the input and queries GBF to obtain the wildcard position set J

$Dec(PK, SK_W, CT, J)$: this function takes PK , SK_W , J , and CT as inputs and outputs M

Here, let $(PK, MSK) \leftarrow Setup(1^\lambda)$, $SK_W \leftarrow KeyGen(PK, MSK, W)$, and $(CT, GBF) \leftarrow Encrypt(PK, M, (\overline{W}, J))$. For correctness, we require the following conditions to hold:

- (1) If the attribute vector W of the DU satisfies the access policy \overline{W} , then $M \leftarrow Decrypt(PK, SK_W, GBF, CT)$
- (2) Otherwise, $Decrypt(PK, SK_W, GBF, CT)$ outputs a random message

4.3. Security Model

Definition 1. A CP-ABE scheme with the hidden access policy is semantically secure in the selective model if for all PPT adversaries \mathcal{A} ,

$$\left| \Pr[Exp_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| < \epsilon(\lambda), \quad (5)$$

for some negligible function $\epsilon(\lambda)$. Based on [29], the security game $Exp_{\mathcal{A}}(\lambda)$ is described as follows:

Init: \mathcal{A} selects two different challenge attribute vectors $\overline{W}_0, \overline{W}_1 \in \Sigma_*^L$, for at least one $\overline{w}_i \neq *$.

Setup: the challenger \mathcal{B} runs $\text{Setup}(1^\lambda)$ algorithm, which outputs PK and MSK . It sends PK to \mathcal{A} and keeps MSK to itself.

Query phase 1: \mathcal{A} adaptively issues key queries for the attribute vector $W \in \Sigma^L$, under the restriction that $w_i \neq \overline{w}_{0i}$ and $w_i \neq \overline{w}_{1i}$. \mathcal{B} runs $\text{KeyGen}(PK, MSK, W)$ algorithm to obtain SK_W and sends SK_W to \mathcal{A} .

Challenge: \mathcal{A} submits two messages M_0, M_1 ($|M_0| = |M_1|$) and sends them to \mathcal{B} . Given \overline{W}_0 and \overline{W}_1 , \mathcal{B} randomly selects $\beta \in \{0, 1\}$ and encrypts M_β under \overline{W}_β . Finally, \mathcal{B} sends CT_β to \mathcal{A} .

Query phase 2: query phase 2 is the same as query phase 1.

Guess: finally, \mathcal{A} outputs its guess $\beta' \in \{0, 1\}$ for β . If $\beta' = \beta$, then return 1; else, return 0.

5. Construction of the BCP-ABE-PHAS

5.1. *Setup Phase.* In this paper, we use $U = \{att_1, \dots, att_L\}$ to represent the attribute universe in the system. Here, $V_i =$

$\{v_{i,1}, \dots, v_{i,n_i}\}$ is the set of possible values of the i^{th} category attribute, where $n_i = |V_i|$. Thus, the user's attribute vector is $W = (w_1, \dots, w_L)$, where $w_i \in V_i, 1 \leq i \leq L$. The access structure of the proposed scheme is $\overline{W} = (\overline{w}_1, \dots, \overline{w}_L) = \wedge_{i \in I_W} \overline{w}_i$, where $I_W = \{i | 1 \leq i \leq L, \overline{w}_i \neq *\}$. If $\overline{w}_i = w_i$ or $\overline{w}_i = *$, we use $W \models \overline{W}$ to denote that the user's attribute vector W satisfies the access policy \overline{W} ; otherwise, we use $W \not\models \overline{W}$ to denote that the user's attribute vector W does not satisfy the access policy \overline{W} . Here, the wildcard $*$ in the access structure means "do not care." The upper bound of the wildcard in the access structure is defined as N , where $N \ll L$.

The setup phase is run by the AA. Here, \mathbb{G} and \mathbb{G}_T are the multiplicative cyclic groups of a large prime order q , g is a generator of \mathbb{G} , and $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map. The AA randomly chooses $\alpha, t_1, t_2, (x_1, \dots, x_N) \in \mathbb{Z}_q$ and $V_0, U_1, \dots, U_L \in \mathbb{G}$ and sets $\Omega_1 = e(g, V_0)^{att_1}$ and $\Omega_2 = e(g, V_0)^{att_2}$. Let $V_j = V_0^{x_j}$ for $j = 1, \dots, N$. The AA also generates k independent hash functions $H = (h_1, \dots, h_k)$. Therefore, the public parameters are expressed as follows:

$$PK = (e, \mathbb{G}, \mathbb{G}_T, g, q, \Omega_1, \Omega_2, g^\alpha, V_0, (x_1, \dots, x_N), H, (U_1, \dots, U_L)). \quad (6)$$

Additionally, the master secret key is expressed as follows:

$$MSK = (\alpha, t_1, t_2, (V_1, \dots, V_N)). \quad (7)$$

5.2. *Data Encryption Phase.* The encryption phase is executed by the DO and involves three main parts, which are described in the following section.

5.2.1. *IPFS Storage.* The DO selects the *file* to be shared, generates the symmetric key *aeskey* using the Advanced Encryption Standard (AES), and encrypts the *file* using *aeskey* to generate the ciphertext *encfile*.

To relieve the pressure on blockchain storage, the proposed scheme stores *encfile* in the IPFS using the *ipfs add encfile* command, and the IPFS returns unique hash value *ipfshash* to retrieve *encfile*. Note that anyone can obtain the ciphertext *encfile* stored in the IPFS using the *ipfs get ipfshash* command.

5.2.2. *Hidden Access Policy.* The blockchain is public, all participants can obtain the data on the blockchain, so we need to hide the attribute information of the access policy. The access policy developed by the DO is $\overline{W} = (\overline{w}_1, \dots, \overline{w}_L)$. Assume that the access policy \overline{W} contains $n \leq N$ wildcards that occur at positions $J = \{j_1, \dots, j_n\}$.

When data are decrypted, determining the position of the wildcard symbols is essential; however, directly sending the set J may reveal the user's private information.

Thus, to solve this problem, we adopt an efficient positioning algorithm based on the GBF. The GBF is a combination of a Bloom filter and secret sharing technology. Differing from traditional Bloom filters that use a bit array, the GBF uses an array of λ bits. The GBF can verify whether an attribute exists in the specified set and locate the position index of the attribute to realize the hidden set J and protect the user's private information. In addition to the probability of hash function collisions, the probability of string matching must be verified. Therefore, the false positive probability of the GBF is less than that of the traditional Bloom filter.

When the DO adds an element $att_j, j \in J$, to the GBF, the algorithm first uses the (k, k) -secret-sharing scheme to randomly generate $k - 1$ λ -bit strings $r_{1,j}, \dots, r_{k-1,j}$ and sets $r_{k,j} = r_{1,j} \oplus \dots \oplus r_{k-1,j} \oplus j$. Then, it hashes att_j with k independent hash functions $H = (h_1, \dots, h_k)$ and obtains $h_1(att_j), \dots, h_k(att_j)$, where $h_i(att_j)$ is uniformly distributed in $[1, m]$. Finally, generated $r_{i,j}$ is stored in the GBF based on the position index generated by $h_i(att_j)$.

When elements are further added to the GBF, if a certain position is already occupied by previously added elements, we reuse the share already stored in the GBF. As shown in Figure 6, when we add j_2 to the GBF, the hash value of $h_d(att_{j_2})$ is the same as the hash value of $h_i(att_{j_1})$. If we modify r_{i,j_1} , the previously added element j_1 cannot be restored; thus, we set $r_{d,j_2} = r_{i,j_1}$. The construction of the GBF is presented in Algorithm 1.

The DO constructs a GBF to hide the set J of wildcard positions based on Algorithm 1 and then uses Viète's

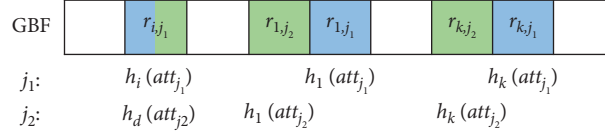


FIGURE 6: The garbled Bloom filter.

Input: set J , security parameter λ , m , and k hash functions $H = \{h_1, \dots, h_k\}$
Output: garbled Bloom filter GBF

- (1) $GBF = \text{new } m\text{-element array of bit strings};$
- (2) **for** $i = 0$ to $m - 1$ **do**
- (3) $GBF[i] = \text{NULL};$ //initialize the GBF with “NULL”
- (4) **end for**
- (5) **for** each $x \in J$ **do**
- (6) $\text{emptySolt} = -1, \text{finalShare} = x;$
- (7) **for** $i = 0$ to $k - 1$ **do**
- (8) $j = h_{i+1}(att_x);$ //get the index of the position
- (9) **if** $GBF[j] == \text{NULL}$ **then**
- (10) **if** $\text{emptySolt} == -1$ **then**
- (11) $\text{emptySolt} = j;$
- (12) **else**
- (13) $GBF[j] \leftarrow \{0, 1\}^\lambda;$ //get a new share
- (14) $\text{finalShare} = \text{finalShare} \oplus GBF[j];$
- (15) **end if**
- (16) **else**
- (17) $\text{finalShare} = \text{finalShare} \oplus GBF[j];$ //reuse an existing share
- (18) **end if**
- (19) **end for**
- (20) $GBF[\text{emptySolt}] = \text{finalShare};$ //store the last share
- (21) **end for**
- (22) **for** $i = 0$ to $m - 1$ **do**
- (23) **if** $GBF[i] == \text{NULL}$ **then**
- (24) $GBF[i] \leftarrow \{0, 1\}^\lambda;$ //fill the position with random strings
- (25) **end if**
- (26) **end for**

ALGORITHM 1: Build GBF.

formulas to compute $\{a_i\}_{1 \leq i \leq n}$. Here, $m = (\sum_{k=0}^n x_k a_k)^{-1}$, where $x_0 = 1$. It randomly chooses $r_1, r_2 \in \mathbb{Z}_q^*$. The DO then creates CT as

$$\begin{aligned}
 C_0 &= M \Omega_1^{r_1} \Omega_2^{r_2}, \\
 C_1 &= g^{amr_1}, \\
 C_2 &= g^{mr_2}, \\
 C_3 &= \left(V_0 \prod_{i=1}^L U_i \prod_{k=1}^n (i - j_k)^{m \bar{w}_i} \right)^{r_1 + r_2},
 \end{aligned} \tag{8}$$

where M is *aeskey*. Therefore, the ciphertext is $CT = (C_0, C_1, C_2, C_3, GBF)$.

5.2.3. Blockchain Storage. A blockchain is an append-only distributed database that stores data on the blockchain permanently, which ensures that the data can be tamper-proof but increases the storage pressure on the blockchain. Therefore, in this paper, the ciphertext *encfile* is stored in the IPFS, and only *ipfshash* and CT are stored on the blockchain. To achieve secure data sharing and fine-grained access

control, we employ smart contracts to ensure that the data storage and sharing process is open and transparent without third-party participation. Here, the public and private keys of the DO in the blockchain are represented by BPK_{DO} and BSK_{DO} , respectively.

Generally, the DO is the creator of the access control contract (ACC) who wants to share data with DUs. The ACC provides application binary interfaces (ABIs) to manage and implement access control. The ABIs of the ACC are presented in Table 1.

The DO creates and deploys the ACC and then obtains the contract address and ABIs. Furthermore, the DO sends a transaction to execute the *uploadfile* ABI of the ACC to upload the data on the blockchain (Algorithm 2).

5.3. Key Generation Phase. The key generation phase is executed by the AA. In this phase, the secret key is generated for the DU, the attribute vector of which is $W = (w_1, \dots, w_L) \in \Sigma^L$. The AA randomly chooses $s \in \mathbb{Z}_q$ and sets $s_1 = t_1 + s$ and $s_2 = t_2 + s$. Then, the following algorithm is executed:

TABLE 1: ABIs of the ACC.

ABI	Permissions	Description
<i>uploadfile()</i>	Contract creator	The ABI uploads data on the blockchain
<i>getfile()</i>	Public	The ABI obtains the data stored on the blockchain
<i>kill()</i>	Contract creator	The ABI performs the <i>selfdestruct</i> operation to delete the ACC

Input: CT , $ipfshash$, $time$, BSK_{DO} , ACC address $addr$, and $uploadfile$ ABI

Output: storage transaction $Tx_{storage}$

- (1) Compute the message digest $MD = H(time, ipfshash, CT)$;
- (2) Generate the signature $sign = sign_{BSK_{DO}}(MD)$;
- (3) Generate a storage transaction $Tx_{storage} = \{time, ipfshash, CT, sign\}$;
- (4) Send $Tx_{storage}$ according to $addr$ and $uploadfile$ ABI;
- (5) **return** $Tx_{storage}$;

ALGORITHM 2: Generate a storage transaction.

$$\begin{aligned}
K &= g^{\alpha s}, \\
K_1 &= \{K_{1,0}, K_{1,1}, \dots, K_{1,N}\} \\
&= \left\{ V_0^{s_1} \prod_{i=1}^L U_i^{sw_i}, V_1^{s_1} \prod_{i=1}^L U_i^{siw_i}, \dots, V_N^{s_1} \prod_{i=1}^L U_i^{si^N w_i} \right\}, \\
K'_1 &= \{K'_{1,0}, K'_{1,1}, \dots, K'_{1,N}\} \\
&= \left\{ V_0^{\alpha s_2} \prod_{i=1}^L U_i^{\alpha s w_i}, V_1^{\alpha s_2} \prod_{i=1}^L U_i^{\alpha s i w_i}, \dots, V_N^{\alpha s_2} \prod_{i=1}^L U_i^{\alpha s i^N w_i} \right\}.
\end{aligned} \tag{9}$$

Therefore, the secret key of the DU is $SK = (K, K_1, K'_1)$.

5.4. Data Decryption Phase. Data decryption is performed by the DU and mainly comprises the following three phases.

5.4.1. Obtain Data on the Blockchain. The DU sends a transaction to execute the *getfile* ABI of the ACC to obtain the data stored on the blockchain (Algorithm 3).

Therefore, the DU obtains *ipfshash* and CT stored on the blockchain.

5.4.2. QueryGBF. The DU obtains the data stored on the blockchain, where $CT = (C_0, C_1, C_2, C_3, GBF)$. As observed from the data encryption phase, obtaining J is the key to decrypting CT . Here, the DU obtains J according to Algorithm 4.

First, we determine whether the hash value of the attribute exists in the GBF. If the corresponding position of the

attribute in the GBF is 0, the attribute must not be in J . When all GBF positions corresponding to the k hash values of the attribute are not empty, the DU must calculate the position index of the wildcard in the access policy using \oplus ; if the calculated value is the same as the position index corresponding to the attribute vector of the DU, the attribute is present in J ; otherwise, this attribute is not in J .

5.4.3. Data Decryption Phase. In this phase, the DU obtains the set of wildcard positions J and then uses Viète's formulas to compute $\{a_i\}_{1 \leq i \leq n}$, where $m = (\sum_{k=0}^n x_k a_k)^{-1}$. The ciphertext can only be decrypted when the attributes of the DU can satisfy the access policy. Then, M can be computed as follows:

$$M = C_0 \frac{e(K, C_3)}{e(C_1, \prod_{k=0}^n K_{1,k}^{a_k}) e(C_2, \prod_{k=0}^n \prod_{k=0}^n (K'_{1,K})^{a_k})}. \tag{10}$$

When the DU successfully decrypts the data, it obtains *aeskey*. Then, the DU obtains *encfile* stored in the IPFS using *ipfshash*. Subsequently, *aeskey* is used to decrypt *encfile* to obtain the *file* shared by the DO.

6. Security Analysis and Performance Evaluation

6.1. Correctness. In this section, we verify the correctness of the proposed scheme. When we use a decryption key that satisfies the given access policy, the Decrypt algorithm indeed returns the correct message.

$$\begin{aligned}
e(K, C_3) &= e\left(g^{\alpha s}, \left(V_0 \prod_{i=1}^L U_i^{\prod_{k=1}^n (i-j_k) \bar{w}_i / \sum_{m=0}^n x_m a_m}\right)^{r_1+r_2}\right) \\
&= e(g, V_0)^{\alpha s (r_1+r_2)} \prod_{i=1}^L e(g, U_i)^{\alpha s \bar{w}_i (r_1+r_2) \sum_{k=1}^n (i-j_k) / \sum_{m=0}^n x_m a_m},
\end{aligned} \tag{11}$$

Input: ACC address $addr$, $getfile$ ABI, and BPK_{DO}
Output: verified result

- (1) Obtain the data according to $addr$ and $getfile$ ABI;
- (2) Compute the message digest $MD' = H(time, ipfhash, CT)$;
- (3) Verify the signature $MD = Verify_{BPK_{DO}}(sign)$;
- (4) **if** $MD' == MD$ **then**
- (5) **return** True;
- (6) **else**
- (7) **return** False;
- (8) **end if**

ALGORITHM 3: Obtain the data stored on the blockchain.

Input: garbled Bloom filter GBF, n , the number of attributes L , security parameter λ , m , and k hash functions $H = \{h_1, \dots, h_k\}$
Output: set J

- (1) J = new set of length m ;
- (2) **for** $x = 1$ to L **do**
- (3) $recovered = \{0\}^L$;
- (4) **for** $i = 1$ to k **do**
- (5) $j = h_i(att_x)$;
- (6) **if** $GBF[j] == NULL$ **then**
- (7) **break**;
- (8) **else**
- (9) $recovered = recovered \oplus GBF[j]$;
- (10) **end if**
- (11) **end for**
- (12) **if** $recovered == x$ **then**
- (13) $J.add(x)$;
- (14) **end if**
- (15) **end for**

ALGORITHM 4: QueryGBF.

where for equation (11), we use that $\sum_{k=0}^n i^k a_k = \prod_{k=1}^n (i - j_k)$,

$$\begin{aligned}
e\left(C_1, \prod_{k=0}^n K_{1,k}^{a_k}\right) &= e\left(g^{\alpha r_1 / \sum_{m=0}^n x_m a_m}, \prod_{k=0}^n \left(V_k^{s_1} \prod_{i=1}^L U_i^{s_1^k w_i}\right)^{a_k}\right) \\
&= e(g, V_0)^{\frac{\alpha r_1 s_1 \prod_{k=0}^n x_k a_k}{\sum_{m=0}^n x_m a_m}} \prod_{i=1}^L e(g, U_i)^{\alpha s r_1 w_i \sum_{k=0}^n i^k a_k / \sum_{m=0}^n x_m a_m} \\
&= \Omega_1^{r_1} e(g, V_0)^{\alpha s r_1} \prod_{i=1}^L e(g, U_i)^{\alpha s r_1 w_i \sum_{k=0}^n i^k a_k / \sum_{m=0}^n x_m a_m} \\
e\left(C_2, \prod_{k=0}^n (K'_{1,k})^{a_k}\right) &= e\left(g^{\frac{r_2}{\sum_{m=0}^n x_m a_m}}, \prod_{k=0}^n \left(V_k^{\alpha s_2} \prod_{i=1}^L U_i^{\alpha s_2^k w_i}\right)^{a_k}\right) \\
&= e(g, V_0)^{\frac{\alpha r_2 s_2 \prod_{k=0}^n x_k a_k}{\sum_{m=0}^n x_m a_m}} \prod_{i=1}^L e(g, U_i)^{\alpha s r_2 w_i \sum_{k=0}^n i^k a_k / \sum_{m=0}^n x_m a_m} \\
&= \Omega_2^{r_2} e(g, V_0)^{\alpha s r_2} \prod_{i=1}^L e(g, U_i)^{\alpha s r_2 w_i \sum_{k=0}^n i^k a_k / \sum_{m=0}^n x_m a_m}
\end{aligned} \tag{12}$$

Then, we have

$$\begin{aligned} & e\left(C_1, \prod_{k=0}^n K_{1,k}^{a_k}\right) e\left(C_2, \prod_{k=0}^n (K_{1,k})^{a_k}\right) \\ &= \Omega_1^{r_1} \Omega_2^{r_2} e(g, V_0)^{\alpha s (r_1 + r_2)} \prod_{i=1}^L e(g, U_i)^{\frac{\alpha s w_i (r_1 + r_2) \sum_{k=0}^n t^k a_k}{\sum_{m=0}^n x_m a_m}}. \end{aligned} \quad (13)$$

If the secret key of the DU is valid, then $w_i = \bar{w}_i$, $i \notin \{j_1, \dots, j_n\}$. Thus,

$$C_0 \frac{e(K, C_3)}{e\left(C_1, \prod_{k=0}^n K_{1,k}^{a_k}\right) \cdot e\left(C_2, \prod_{k=0}^n (K'_{1,k})^{a_k}\right)} = \frac{M \Omega_1^{r_1} \Omega_2^{r_2}}{\Omega_1^{r_1} \Omega_2^{r_2}} = M. \quad (14)$$

6.2. Security Analysis

Theorem 1. *The proposed BCP-ABE-PHAS scheme is semantically secure in the selective model assuming that the DLIN assumption holds in group \mathbb{G} .*

Proof. Assume there exists a PPT adversary \mathcal{A} that can break the selective semantic security. We then build an algorithm \mathcal{B} that uses \mathcal{A} to solve the DLIN problem in \mathbb{G} .

Here, the challenger selects a bilinear group \mathbb{G} of prime order q and a generator $g \in \mathbb{G}$, as well as the group \mathbb{G}_T and a bilinear map $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Then, the challenger randomly chooses five values $a, b, c, d, r \in \mathbb{Z}_q^*$ and computes $Z_0 = g^{b(c+d)}$ and $Z_1 = g^r$. The challenger randomly chooses $\beta \in \{0, 1\}$ and sends the tuple $(g, g^a, g^b, g^{ac}, g^d, Z_\beta)$ to \mathcal{B} . Note that the goal of \mathcal{B} is to guess β with a probability greater than $1/2$. To generate a guess, \mathcal{B} interacts with \mathcal{A} in the following selective semantic security experiment.

Init: \mathcal{A} chooses two challenge attribute vectors $\bar{W}_0 \in \Sigma_*^L$ and $\bar{W}_1 \in \Sigma_*^L$. Here, the wildcard position sets are

denoted as J_0 and J_1 , respectively. Then, \mathcal{B} randomly chooses $\gamma = \{0, 1\}$, where $\bar{W}_\gamma = (\bar{w}_1, \dots, \bar{w}_L)$.

Setup: \mathcal{B} chooses $N \leq L$, which is the upper bound number of wildcards. Then, \mathcal{B} chooses $v_0, u_1, \dots, u_L \in \mathbb{Z}_q$ uniformly at random and sets the following:

$$\begin{aligned} x_j &= \frac{\sum_{i=1}^L t^{ju_i}}{\sum_{i=1}^L u_i} \quad \text{for } j = 0, \dots, N, \\ V_j &= (g^b)^{x_j v_0} g^{-\sum_{i=1}^L i j u_i} \quad \text{for } j = 0, \dots, N, \\ U_i &= g \frac{u_i}{\bar{w}_i} \quad \text{for } i = 1, \dots, L. \end{aligned} \quad (15)$$

Here, \mathcal{B} randomly chooses $\sigma_1, \sigma_2, \sigma_3 \in \mathbb{Z}_q$ and computes $\Omega_1 = e(g^a, V_0)^{\sigma_1 - \sigma_2}$ and $\Omega_2 = e(g^{\sigma_3} \cdot (g^a)^{-\sigma_2}, V_0)$.

The public key is expressed as follows:

$$PK = (e, \mathbb{G}, \mathbb{G}_T, g, q, \Omega_1, \Omega_2, g^a, V_0, (x_1, \dots, x_N), H, (U_1, \dots, U_L)). \quad (16)$$

Additionally, the master secret key is expressed as follows:

$$MSK = \left(\alpha = a, t_1 = \sigma_1 - \sigma_2, t_2 = \frac{\sigma_3}{a} - \sigma_2, (V_1, \dots, V_N) \right). \quad (17)$$

Query phase 1: in this phase, \mathcal{B} will respond to the key query of \mathcal{A} . Each time, \mathcal{A} will commit an attribute vector $W = (w_1, \dots, w_L)$ and set $s = \sigma_2$, $s_1 = t_1 + \sigma_2$,

and $s_2 = \sigma_2 + \sigma_3/a - \sigma_2 = \sigma_3/a$. Then, \mathcal{B} responds by computing

$$\begin{aligned}
K &= g^{a\sigma_2}, \\
K_1 &= \{K_{1,0}, K_{1,1}, \dots, K_{1,N}\} \\
&= \left\{ g^{(bv_0 - \sum_{i=1}^L u_i)\sigma_1} \prod_{i=1}^L g^{\sigma_2 u_i / \bar{w}_i} w_i, g^{(bv_0 - \sum_{i=1}^L u_i)x_1 \sigma_1} \prod_{i=1}^L g^{\sigma_2 i u_i / \bar{w}_i} w_i \right. \\
&\quad \left. \dots, g^{(bv_0 - \sum_{i=1}^L u_i)x_N \sigma_1} \prod_{i=1}^L g^{\sigma_2 i^N u_i / \bar{w}_i} w_i \right\} \\
K'_1 &= \{K'_{1,0}, K'_{1,1}, \dots, K'_{1,N}\} \\
&= \left\{ g^{(bv_0 - \sum_{i=1}^L u_i)\sigma_3} \prod_{i=1}^L (g^a)^{\sigma_2 u_i / \bar{w}_i} w_i, g^{(bv_0 - \sum_{i=1}^L u_i)x_1 \sigma_3} \prod_{i=1}^L (g^a)^{\sigma_2 i u_i / \bar{w}_i} w_i \right. \\
&\quad \left. \dots, g^{(bv_0 - \sum_{i=1}^L u_i)x_N \sigma_3} \prod_{i=1}^L (g^a)^{\sigma_2 i^N u_i / \bar{w}_i} w_i \right\}.
\end{aligned} \tag{18}$$

Finally, \mathcal{B} sends $SK = (K, K_1, K'_1)$ to \mathcal{A} .

Challenge: when query phase 1 is over, \mathcal{A} sends two messages $M_0, M_1 \in \mathbb{G}_T$, ($|M_0| = |M_1|$), to \mathcal{B} . Then, \mathcal{B}

randomly chooses and outputs its guess and selects a message M_γ to encrypt under \bar{W}_γ . Then, \mathcal{B} creates

$$\begin{aligned}
C_0 &= M_\gamma \cdot e(g^{ac}, g^{bv_0})^{\sigma_1 - \sigma_2} \cdot e(g^{ac}, g)^{\sum_{i=1}^L u_i (\sigma_1 - \sigma_2)} \\
&\quad \cdot e(g^d, g^b)^{v_0 \sigma_3} \cdot e(g^d, g^{-a\sigma_2})^{-\sum_{i=1}^L u_i} \cdot e(g^d, g^{\sigma_3})^{-\sum_{i=1}^L u_i}, \\
C_1 &= g^{ac / \sum_{m=0}^n x_m a_m}, \\
C_2 &= g^{d / \sum_{m=0}^n x_m a_m}, \\
C_3 &= Z_\beta^{v_0},
\end{aligned} \tag{19}$$

\mathcal{B} sends the challenge ciphertext $CT = (C_0, C_1, C_2, C_3, GBF)$ to \mathcal{A} .

Query phase 2: query phase 2 is the same as query phase 1.

Guess: \mathcal{A} outputs its guess $\gamma' \in \{0, 1\}$ for γ .

Finally, if $\gamma' = \gamma$, \mathcal{B} outputs 1; otherwise, \mathcal{B} outputs 0. In the following, we analyze the probability of success for \mathcal{B} . Here, if $\beta = 0$, then \mathcal{B} will behave correctly as a

challenger to \mathcal{A} . Furthermore, \mathcal{A} will have the probability of $1/2 + \epsilon$ of guessing γ . If $\beta = 1$, \mathcal{A} will have the probability of $1/2$ of guessing γ .

To conclude this proof, we obtain the following:

$$\begin{aligned}
& \left| \Pr[\mathcal{B}(g, g^a, g^b, g^{ac}, g^d, g^{b(c+d)}) = 1] - \Pr[\mathcal{B}(g, g^a, g^b, g^{ac}, g^d, g^r) = 1] \right| \\
& \geq |\Pr[\beta = 0 \wedge \gamma' = \gamma]| - |\Pr[\beta = 1 \wedge \gamma' = \gamma]| \\
& = \left| \frac{1}{2} \Pr[\gamma' = \gamma \mid \beta = 0] - \frac{1}{2} \Pr[\gamma' = \gamma \mid \beta = 1] \right| \\
& = \frac{1}{2} \left| \Pr[\text{Exp}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| \\
& \geq \frac{1}{2} \epsilon,
\end{aligned} \tag{20}$$

which is nonnegligible, thereby contradicting the DLIN assumption. \square

3.5 ms. Therefore, introducing the GBF does not increase the computational overhead of the system.

6.3. Performance Evaluation

6.3.1. Data Storage Efficiency. We conduct an experiment to verify the efficiency of two storage schemes, i.e., cloud-based server storage and IPFS-based distributed storage. Here, we set up a local server and an IPFS cluster in the same local area network. The cluster comprises five local devices, all of which are Ubuntu 18.04 systems with an Intel Core i5 CPU at 2.4 GHz with 4 GB RAM. We compare the performance of the two storage schemes based on the time required for uploading and downloading files. Figure 7 shows the transmission time required by the two schemes. Note that all experimental results are the average of 30 trials. As shown in Figure 7, the upload and download time of files in the IPFS-based scheme are less than those of the files in the cloud-based scheme. Moreover, the time required for uploading and downloading the files in the cloud-based scheme exhibits a faster growth trend than the IPFS-based scheme when the file size increased. Therefore, the IPFS-based scheme can improve storage efficiency and system scalability.

6.3.2. GBF Efficiency. We conduct another experiment to verify the storage and query efficiency of the GBF. Here, we use double hashing technology, and the k hash functions of the GBF are constructed using the 128-bit SpookyHash and MurmurHash. The length of the GBF is set to $m = 1024$, and $\lambda = 8$ in this experiment. Furthermore, the number of attributes in the access policy is 5–35, the number of wildcards is 2–14, and the number of hash functions is $k = 6, 8, \text{ and } 10$. Note that all reported experimental results are the average values obtained over 30 trials. As shown in Figure 8, for $k = 8$, the time required to add 10 wildcards to the GBF is approximately 4.5 ms, and the query time is approximately

6.3.3. Performance Evaluation. We also analyze the performance of our scheme and five existing CP-ABE schemes with AND gates in terms of the ciphertext size, decryption consumption, whether access policies are hidden, and so on. The results are presented in Table 2, where p represents the pairing operation, e represents the exponentiation operation, l represents the number of attributes in the access structure, m represents the number of possible values for an attribute, and n represents the number of wildcards in the access structure. From Table 2, among all schemes that support wildcards and the hidden access policy, our scheme exhibits the smallest ciphertext size. Furthermore, the ciphertext size of our scheme is constant. Note that the decryption consumption of our scheme is related to the number of wildcards n , where $n \ll l$; thus, our scheme has the advantage in terms of decryption consumption.

To evaluate the actual performance of our scheme, we compare it to schemes proposed in [10, 31]. Here, we implement our scheme on a desktop PC (3.4 GHz Intel Core i7 CPU with 16 GB RAM) based on Ubuntu 18.04 LTS and Java Pairing-Based Cryptography Library (JPBC) 2.0.0. This implementation uses a 160-bit elliptic curve group based on the supersingular curve $y^2 = x^3 + x$ over a 512-bit finite field. The number of attributes in the access policy is 5–35, and the number of wildcards is 2–14. To ensure accuracy in our experiments, all reported experimental results are the averages obtained over 30 trials.

Figure 9(a) shows the size of the public parameters in the setup phase. The size of public parameters in all schemes increases linearly with the increase in the number of attributes. Nishide et al. [10] defined all possible values for each attribute in the public parameter; thus, the size of the public parameters is larger than other schemes. Figure 9(b) presents the execution time of the key generation phase. In our scheme, the key is generated

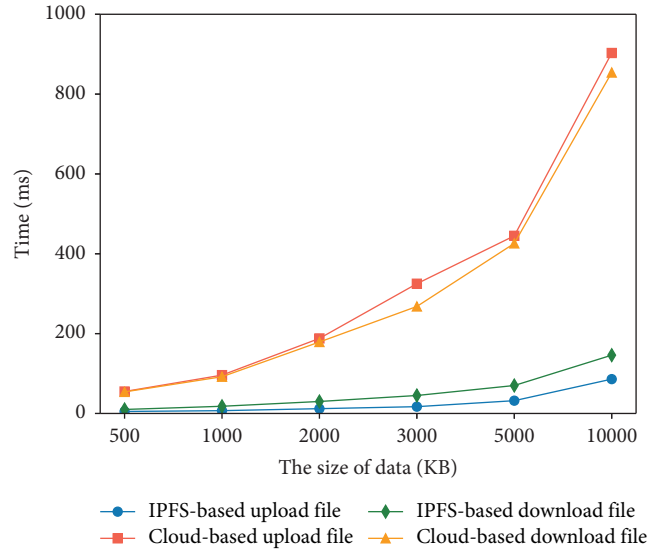


FIGURE 7: Transmission time of the two schemes.

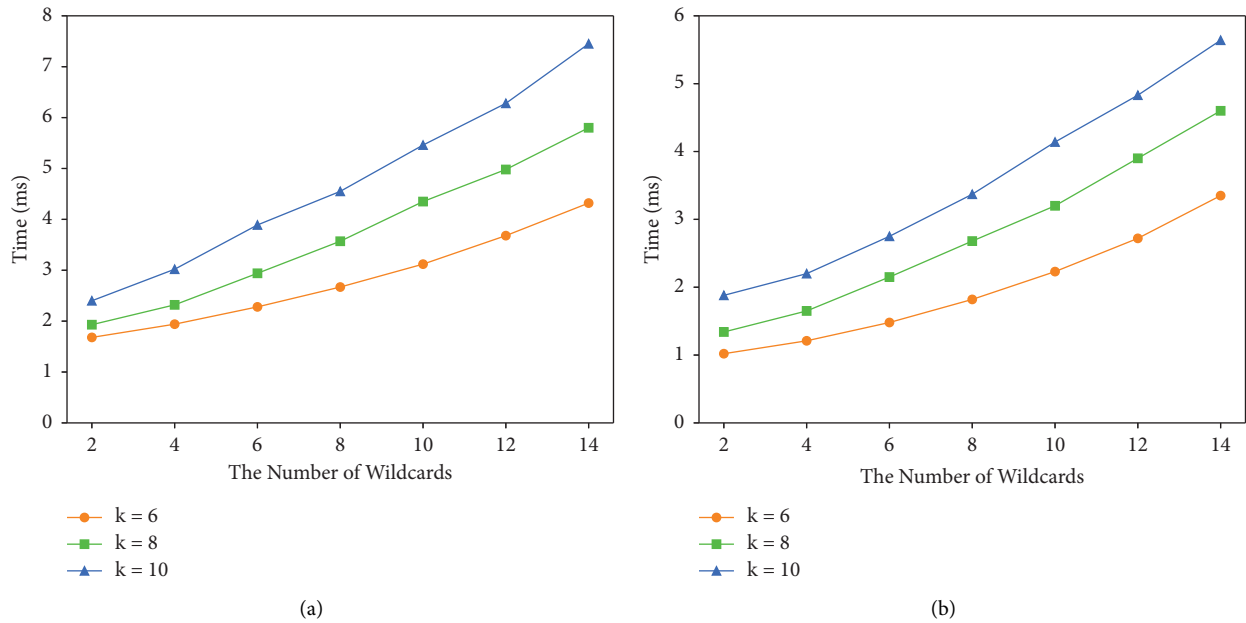


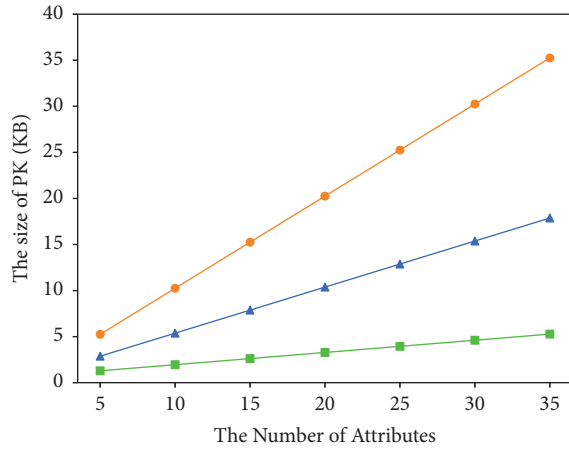
FIGURE 8: Computational time for (a) BuildGBF and (b) QueryGBF.

by the AA with high computational power; therefore, although the time required in our scheme is greater than other schemes, it will not affect the efficiency of our scheme. Figure 9(c) presents the execution time of the encryption operation. Here, the number of exponentiation operations in the encryption algorithm is related to the attribute; thus, the required time increases with the increase in the number of attributes. The scheme in [10]

exhibits more exponentiation operations in the encryption phase than our scheme and the scheme in [31]; thus, the required time is greater than the other two schemes. Figure 9(d) shows the execution time of the decryption operation. Note that the existing scheme [31] does not hide the access policy; thus, its decryption time is fixed. The decryption time of our scheme is less than that of the scheme in [10].

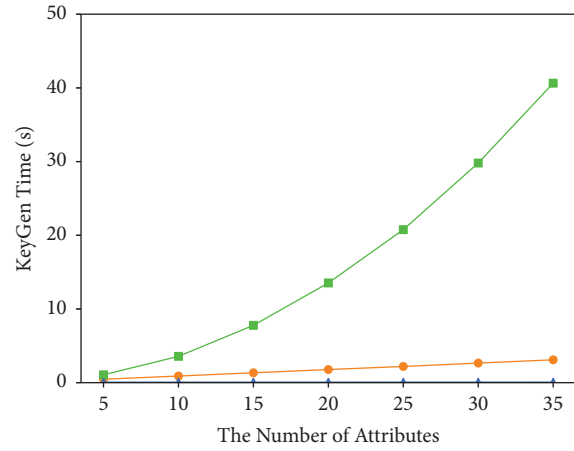
TABLE 2: Performance comparison of CP-ABE schemes.

Scheme	Group order	Access structures	Ciphertext size	Decryption cost	Wildcard	Hidden policy
[10]	Prime	AND gates on multivalued attributes	$ \mathbb{G}_T + (2ml + 1) \mathbb{G} $	$(3l + 1)p$	\checkmark	\checkmark
[12]	Prime	AND gates on \pm	$ \mathbb{G}_T + (4n + 2) \mathbb{G} $	$(4n + 2)p$	\checkmark	\checkmark
[31]	Prime	AND gates on multivalued attributes	$ \mathbb{G}_T + 2 \mathbb{G} $	$2p$	\times	\times
[32]	Composite	AND gates on multivalued attributes	$ \mathbb{G}_T + (2ml + 2) \mathbb{G} $	$(l + 1)p$	\checkmark	\checkmark
[33]	Prime	AND gates on \pm	$ \mathbb{G}_T + (l + 1) \mathbb{G} $	$(l + 1)p$	\checkmark	\times
Ours	Prime	AND gates on multivalued attributes	$ \mathbb{G}_T + 3 \mathbb{G} $	$3p + 2ne$	\checkmark	\checkmark



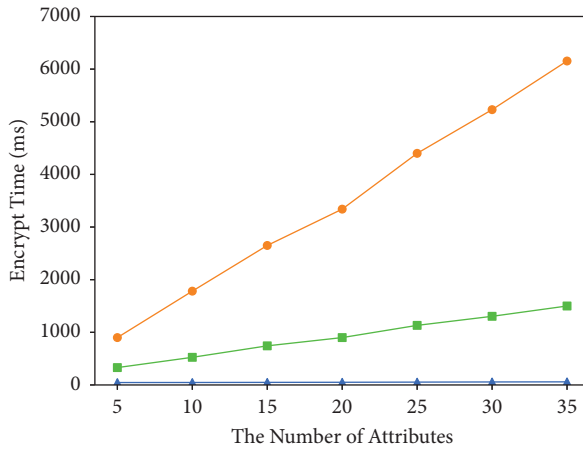
—○— Scheme [10]
—■— BCP-ABE-PHAS
—▲— Scheme [31]

(a)



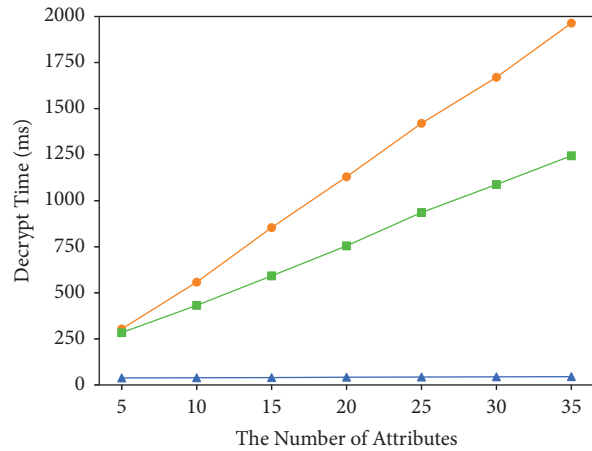
—○— Scheme [10]
—■— BCP-ABE-PHAS
—▲— Scheme [31]

(b)



—○— Scheme [10]
—■— BCP-ABE-PHAS
—▲— Scheme [31]

(c)



—○— Scheme [10]
—■— BCP-ABE-PHAS
—▲— Scheme [31]

(d)

FIGURE 9: Performance analysis and comparisons. (a) Public key. (b) Key generation. (c) Encryption data. (d) Decryption data.

In summary, our scheme has advantages in terms of data encryption and decryption when implementing the hidden access policy.

7. Conclusion

In this paper, we propose a BCP-ABE-PHAS scheme to achieve trustworthy access while ensuring user privacy. Traditional centralized storage architectures are vulnerable to various network attacks, e.g., single point of attack, man-in-the-middle attack, and distributed denial-of-service attack. Therefore, we adopt a data storage scheme that combines blockchain technology and the IPFS; this approach relieves the storage pressure on the blockchain and guarantees data integrity. The experimental results demonstrate that our scheme is efficient and maintains a constant ciphertext size. Furthermore, to assist data decryption, we design a GBF to help users quickly locate the position of wildcards in the access policy. The proposed scheme uses smart contracts to guarantee that the entire data storage and sharing process is transparent, dynamic, and automated. The results of security analysis and performance evaluations demonstrate that our scheme is secure and efficient.

Data Availability

As part of the data in the paper is confidential, the code cannot be published for the time being. If data needed, send the corresponding author an email.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62172433, 62172434, 61862011, and 61872449).

References

- [1] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," *IEEE Network*, vol. 28, no. 4, pp. 46–50, 2014.
- [2] A. Sahai and B. Waters, "Fuzzy identity-based encryption, lecture notes in computer science," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Springer, Aarhus, Denmark, May 2005.
- [3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 321–334, IEEE, CA, USA, May 2007.
- [4] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 89–98, Alexandria, VA, USA, November 3 2006.
- [5] H. Cui, R. H. Deng, J. Lai, X. Yi, and S. Nepal, "An efficient and expressive ciphertext-policy attribute-based encryption scheme with partially hidden access structures, revisited," *Computer Networks*, vol. 133, pp. 157–165, 2018.
- [6] Z. Li, D. Wang, and E. Morais, "Quantum-safe round-optimal password authentication for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2020.
- [7] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2020.
- [8] N. Nizamuddin, H. R. Hasan, and K. Salah, "IPFS-blockchain-based authenticity of online publications," in *Proceedings of the International Conference on Blockchain*, pp. 199–212, Springer, Xi'an, China, December 22 2018.
- [9] C. Wang, D. Wang, G. Xu, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy for industry 4.0," *Science China Information Sciences*, vol. 65, no. 1, pp. 1–15, 2021.
- [10] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 111–129, Springer, New York, NY, USA, June 2008.
- [11] J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability," in *Proceedings of the International Conference on Information Security*, pp. 347–362, Springer, Pisa, Italy, September 2009.
- [12] T. V. X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 35–45, 2015.
- [13] J. Lai, R. H. Deng, and Y. Li, "Expressive CP-ABE with partially hidden access structures," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pp. 18–19, ACM, Seoul, Republic of Korea, May 2 2012.
- [14] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: efficient policy-hiding attribute-based access control," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2130–2145, 2018.
- [15] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 146–162, Springer, Istanbul, Turkey, April 2008.
- [16] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 62–91, Springer, French Riviera, May 30 2010.
- [17] J. Hur, "Attribute-based secure data sharing with hidden policies in smart grid," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 11, pp. 2171–2180, 2013.
- [18] S. Gao, G. Piao, J. Zhu, X. Ma, and J. Ma, "TrustAccess: a trustworthy secure ciphertext-policy and attribute hiding access control scheme based on blockchain," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5784–5798, 2020.
- [19] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan, "Smart contract-based access control for the Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1594–1605, 2018.

- [20] J. Xu, K. Xue, S. Li et al., "Healthchain: a blockchain-based privacy preserving scheme for large-scale health data," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8770–8781, 2019.
- [21] R. Xu, Y. Chen, E. Blasch, G. Chen, and C. A. C. Blend, "A blockchain-enabled decentralized capability-based access control for IoTs," in *Proceedings of the 2018 IEEE International Conference on Internet of Things*, pp. 1027–1034, Halifax NS Canada, August 3 2018.
- [22] O. Novo, "Blockchain meets IoT: an architecture for scalable access management in IoT," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1184–1195, 2018.
- [23] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008, <https://bitcoin.org/bitcoin.pdf>.
- [24] N. Szabo, "Smart contracts," 1994, <http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html>.
- [25] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2014, <https://www.gavwood.com/paper.pdf>.
- [26] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.
- [27] P. Bose, H. Guo, E. Kranakis et al., "On the false-positive rate of Bloom filters," *Information Processing Letters*, vol. 108, no. 4, pp. 210–213, 2008.
- [28] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [29] S. Sedghi, P. van Liesdonk, S. Nikova, P. Hartel, and W. Jonker, "Searching keywords with wildcards on encrypted data," in *Proceedings of the International Conference on Security and Cryptography for Networks*, pp. 138–153, Springer, Amalfi, Italy, September 16 2020.
- [30] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proceedings of the Annual International Cryptology Conference*, pp. 41–55, Springer, Santa Barbara, California, USA, August 22 2002.
- [31] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi, "A ciphertext-policy attribute-based encryption scheme with constant ciphertext length," in *Proceedings of the International Conference on Information Security Practice and Experience*, pp. 13–23, Springer, Xi'an, China, April 2009.
- [32] J. Lai, R. H. Deng, and Y. Li, "Fully secure ciphertext-policy hiding CP-ABE," in *Proceedings of the International Conference on Information Security Practice and Experience*, pp. 24–39, Springer, Guangzhou, China, May 2011.
- [33] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 456–465, ACM, NY, USA, October 31 2007.

Research Article

An Improved Coercion-Resistant E-Voting Scheme

Yuanjing Hao , Zhixin Zeng , and Liang Chang 

Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China

Correspondence should be addressed to Liang Chang; changl@guet.edu.cn

Received 16 August 2021; Revised 20 September 2021; Accepted 22 September 2021; Published 18 October 2021

Academic Editor: Chunhua Su

Copyright © 2021 Yuanjing Hao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

E-voting has gradually replaced the traditional voting methods to make it easier for people to conduct an election. Recently, Liu et al. propose an unconditional secure e-voting scheme using secret sharing and k -anonymity. Their scheme achieves correct tallying results without revealing raw voting information. However, in this paper, we observe that Liu et al.'s scheme cannot achieve coercion resistance in e-voting since the voter can prove the content of his ballot to the colluded candidates. Then, we propose an improved e-voting scheme to cover up the ballot of the voter with masked values. In this way, even if the voter colludes with corresponding candidates, he cannot prove which candidate he has voted for. Moreover, comparing with Liu et al.'s scheme, the security analysis shows that our proposed e-voting scheme achieves these security requirements like the coercion resistance, integrity of ballots, privacy of ballots, multiple-voting detection, and fairness. Through performance analysis, the experimental results show that our proposed e-voting scheme has higher time efficiency. Compared with other schemes, our scheme achieves a complete voting process and obtains the correct tallying result without complex computation and intricate communication process.

1. Introduction

At present, election is regarded as an indispensable democratic activity in real life. Over time, it has been divided into two categories: traditional election and electronic election. The traditional election consumes a lot of time and resources and has low tallying efficiency. Thus, Chaum [1] first proposed the electronic voting scheme in 1981, which eliminates these disadvantages. Subsequently, various e-voting schemes continue to spring up. Recently, cloud technology and blockchain technology are widely used in e-voting to achieve secure electronic election. For the cloud technology, Shankar et al. [2] proposed a secure e-voting protocol, which realizes secure data transfer with cloud effectively. Anjima and Hari [3] proposed a secure cloud e-voting system using fully homomorphic elliptical curve cryptography, which greatly ensures the privacy of ballots and minimizes the risk of the vote being tampered or leaked. Although the cloud technology presents advantages in the application of e-voting, it is less used than blockchain technology. Currently, some e-voting researches related to blockchain have a tendency to increase gradually, such as reducing voter fraud [4], preventing ballot content attacks [5], and achieving the

universal verification of ballot [6]. However, the previously mentioned description mainly illustrates the recent development of e-voting, rather than the scope of our study.

In a real election, due to multiparty participation, the voter may sell the content of his ballot to others to obtain profit. Out of curiosity, candidates may collude with other participants to infer the content of the original ballot. Therefore, in practical application, the secure e-voting scheme needs to satisfy some necessary security requirements to protect the privacy of ballot, such as freedom (no one can be forced to cast a certain vote); fairness (no one can use more than his own votes to influence the tallying result); confidentiality (no one can know other voter's content except himself); coercion resistance [7–10] (the voter proves nothing about the content of his ballot to others); verifiability [11, 12] (every voter can verify if his ballot is correctly tallied, and any participant can verify the correctness of tallying result).

For meeting the previously mentioned security requirements, some encryption techniques have been popularly applied to e-voting to achieve secure election, such as mix-net [13–16], blind signature [17–23], homomorphic encryption [24–28], and secret sharing [29–31]. Meanwhile,

due to the high overhead, mix-net is hard to be applied for the actual election. Although blind signature has better practicality in e-voting, it does not satisfy the security requirement of receipt-freeness and verifiability to some extent. At present, in order to achieve secure and feasible e-voting, homomorphic encryption is popular in conjunction with other encryption techniques, such as zero-knowledge proof [32] and partial knowledge proof [33]. However, the computation burden becomes a problem that needs to be solved. Thus, avoiding these disadvantages of above encryption techniques, secret sharing is applied to e-voting because of its better completeness and feasibility and also achieves running time in linear [34]. Based on such advantages, Liu and Zhao [35] recently proposed an e-voting scheme using secret sharing and k-anonymity, which achieves the correct tallying result and satisfies the necessary security requirements. Concretely, the content of one ballot is used to sever as the coefficients of the shared polynomial. The calculated shares are regarded as the encrypted form of one ballot, respectively held by voter, voting system, and all candidates. Then, according to the additive homomorphism of secret sharing, the total number of ballots can be correctly obtained for each candidate. Liu and Zhao [35] declare their e-voting scheme achieves the security requirement of coercion resistance, which means the voter cannot prove the content of his ballot to others.

However, in this paper, we observe that Liu et al.'s scheme cannot achieve the security requirement of coercion resistance. The voter can collude with candidates to prove the content of his ballot. Thus, we propose an improved coercion-resistant e-voting scheme to cover up the content of ballot with masked values, which achieves the security requirement of coercion resistance. Meantime, we also solve the abstention from voting. Additionally, through theoretical analysis and data simulation, we indicate that our proposed e-voting scheme can solve the shortcoming in [35] and has higher time efficiency compared with [32, 33]. Our contributions are shown in details as follows:

- (1) In Liu et al.'s scheme [35], all voters and candidates hold shares. The shared polynomial constructed from the original ballot information can be recovered if the voter colludes with corresponding candidates. The voter can prove which candidate he has voted for. Liu et al.'s scheme does not satisfy the coercion-resistant security requirement. Thus, we propose an improved e-voting scheme, which replaces the content of the original ballot with masked values to achieve the construction of the share polynomial. In this way, even if the voter colludes with corresponding candidates to recover the shared polynomial, he cannot prove the content of his ballot to others. The improved e-voting scheme achieves coercion-resistant security requirement.
- (2) The improved e-voting scheme considers abstention which Liu et al.'s scheme [35] does not accomplish. In the improved e-voting scheme, voting system constructs the shared polynomial of abstainer only using masked values, and then performs subsequent

voting process. Finally, all participants can obtain the correct tallying result.

- (3) The security analysis shows that the improved e-voting scheme not only inherits some security requirements in Liu et al.'s scheme [35] but also achieves coercion-resistant security requirement which Liu et al.'s scheme does not satisfy. Moreover, the scheme also achieves additional security properties which consist of the integrity of ballots, the privacy of ballots, multiple-voting detection, and fairness. The performance analysis shows that the proposed e-voting scheme has higher time efficiency when candidates and voters are, respectively, specific number.

The rest of this paper is organized as follows. In Section 2, the steps of Liu et al.'s e-voting scheme are reviewed. The system model and threat model are presented in Section 3. Section 4 proposes our improved e-voting scheme in detail. In Section 5, we state the differences between Liu et al.'s scheme and the improved e-voting scheme. Finally, the system analysis and conclusion are, respectively, presented in Section 6 and Section 7.

2. Review

In this section, Liu et al.'s scheme [35] is reviewed. It mainly consists of three phases: prevoting phase, voting phase, and postvoting phase. And the process of communication is shown in Figure 1.

2.1. Prevoting Phase. Assume that there are n voters and m candidates, respectively. The voters are divided into several sets, and each set contains k voters.

2.2. Voting Phase

Step 1. Each voter V_i ($i = 1, \dots, n$) registers to a trusted authority centre (AC). Then, the voting system (VS) issues a temporary identity ID_i for each voter, and no one knows the relationship between voter and his temporary identity ID_i .

Step 2. If the candidate C_j ($j = 1, \dots, m$) is selected by the voter V_i , $a_{i,j} = 1$; otherwise, $a_{i,j} = 0$. Then, VS generates the shared polynomial $f_i(x) = a_{i,0} + a_{i,1}x + a_{i,2}x^2 + \dots + a_{i,m}x^m \bmod p$ and computes $m + 2$ shares $(x_j, y_{i,j})$, ($j = 1, 2, \dots, m + 2$) by using the IDs of voter V_i , candidate C_j ($j = 1, \dots, m$) and VS.

Step 3. VS stores the share $(x_{m+1}, y_{i,m+1})$, and then sends shares $(x_j, y_{i,j})$, ($j = 1, 2, \dots, m$) to corresponding candidate C_j and credential $CR_i = \{a_{i,0}, x_{m+2}, y_{i,m+2}\}$ to the voter V_i .

2.3. Postvoting Phase. In this phase, first, voters are randomly divided into some sets, and each set contains k voters. The process can be briefly described in the following steps:

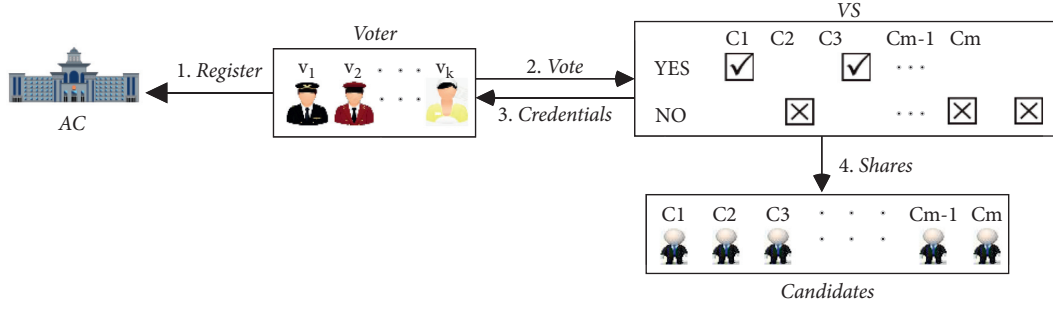


FIGURE 1: The process of communication.

Step 1. The voters who are located in a set publish their $a_{i,0}$, ($i = 1, \dots, k$) on the bulletin board.

Step 2. VS and C_j compute $y_j = \sum_{i=1}^k y_{i,j}$ and publish the points (x_j, y_j) , ($j = 1, 2, \dots, m+1$) on the bulletin board. According to these points, each participant can recover the polynomial $F(x) = a_0 + a_1x + a_2x^2 + \dots + a_mx^m \bmod p$, where $a_j = \sum_{i=1}^k a_{i,j}$, ($j = 0, 1, \dots, m$).

Step 3. VS publishes the aggregated ballots $\{a_0, a_1, a_2, \dots, a_m\}$ of k voters on the bulletin board.

Step 4. The participant verifies the correctness of the aggregated ballots by the equation $a_0 = \sum_{i=1}^k a_{i,0}$. If the verification is true, everyone can compute the result of C_j , $\text{vote}_j = \sum a_j$, ($j = 1, 2, \dots, m$). Otherwise, all candidates and VS are asked to check their publishing information and reconstruct the polynomial again.

3. The System Model and Threat Model

In this section, the system model, threat model, and design goals are introduced, respectively.

3.1. The System Model. In e-voting system, the participants involved are, respectively, a voter (V), candidate (C), trusted authority centre (AC), voting system (VS), and bulletin board, which is used to publish information in voting process. These participants mainly achieve the following functions:

V: the voter votes for his favourite candidate and obtains the credential from VS

C: the candidate and VS collaborate to get the tallying result together

AC: AC authorizes the legal voter to cast the ballot and takes charge of arbitrating the disputes and granting the digital certificate for each participant

VS: VS generates shares for candidate and credential for voter, respectively; moreover, VS leaks nothing about the intention of the voter

In this paper, the communication model is the same as Liu et al.'s scheme and presented in Figure 1. In the communication model, a legal voter casts his favourite candidate using VS. VS generates corresponding credential for this voter and divides the masked voting intention of this voter

into m shares. Then, VS sends credential to this voter and m shares to each candidate C_j ($j = 1, \dots, m$).

3.2. The Threat Model. In this section, the threat model is described, mainly embodied in the security issue of Liu et al.'s scheme. Many secure e-voting schemes can ensure the correctness of the tallying result. Meanwhile, Liu et al.'s scheme uses the homomorphic additivity of polynomial to get the correct tallying result. However, in this scheme, the voter can prove the content of his ballot to others. Their scheme cannot resist coercive attack from the internal voter, which cannot achieve coercion-resistant security requirement in e-voting. For the shortcoming in Liu et al.'s scheme, we give a brief example to describe the attack model. Assume that there are four candidates, and their IDs are separately 2, 3, 4, and 5.

3.2.1. Adversary. Assume that V_1 is an internal attacker who launches the coercive attack to prove the content of his ballot to others. Meantime, assume that V_i 's ID is 1 and he casts candidates C_1, C_3, C_4 , $a_{1,1} = 1, a_{1,2} = 0, a_{1,3} = 1, a_{1,4} = 1$.

3.2.2. Attack Process

- (1) First, for simplifying computation, we choose $p = 29$. In voting phase, according to the selection of the voter V_1 , VS generates the shared polynomial $f_1(x) = 3 + x + x^3 + x^4 \bmod p$. Then, VS computes four shares (2,0), (3,27), (4,8), and (5,4), which are separately sent to C_1, C_2, C_3, C_4 and one credential $\{3, 1, 6\}$, which is sent to V_1 .
- (2) After receiving the credential, V_1 colludes with the candidates C_1, C_3, C_4 since he casts the supporting ballot for them. In this way, V_1 can recover the shared polynomial $f_1(x)$ by owned random number 3 and known three shares of candidates C_1, C_3, C_4 .
- (3) In order to prove the correctness of the recovered polynomial $f_1(x)$, V_1 verifies the recovered polynomial $f_1(x)$ using his share (1,6) and published random number $a_{1,0} = 3$. If the verification is true, the candidates C_1, C_3, C_4 believe that the recovered polynomial $f_1(x)$ is true. In fact, the verification is usually true as long as the calculation is correct.

3.2.3. Attack Result. Through construction of the shared polynomial in Liu et al.'s scheme, we know that the coefficient of the shared polynomial can show whether corresponding candidate obtains the ballot or not. If the coefficient is 1, it shows that the voter casts the candidate whose identity is the same as power of the shared polynomial. Thus, according to the description in attack process, the candidates C_1, C_3, C_4 believe the correctness of the recovered polynomial $f_1(x)$. Naturally, V_1 can prove he cast the supporting ballot for the candidates C_1, C_3, C_4 . Liu et al.'s scheme cannot achieve the security requirement of coercion resistance in e-voting.

Therefore, to solve the shortcoming of Liu et al.'s scheme, we propose the improved coercion-resistant e-voting scheme. The scheme not only satisfies the coercion-resistant security requirement but also solves the issue of abstainers. The specific process is presented in section 4.

3.3. Design Goals. For the design goals, we mainly introduce some necessary security requirements, which needs to be satisfied in next proposed e-voting scheme. These security requirements include coercion resistance, which Liu et al.'s scheme does not achieve and other additional security requirements, which are integrity of ballots, privacy of ballots, multiple-voting detection, and fairness.

- (i) Coercion resistance: no one voter can prove to others which candidate he has voted for.
- (ii) Integrity of ballots: in order to obtain correct tallying result, the ballots of all voters involved in the voting process should be counted validly.
- (iii) Privacy of ballots: no one participant can leak any voting information.
- (iv) Multiple-voting detection: a legal voter only can cast ballot once. If a voter votes for more than once, the superfluous ballots should be detected.
- (v) Fairness: no one candidate can know his own ballot in advance.

4. The Improved Voting Scheme

In this section, we present the improved e-voting scheme in detail. We first suppose that the trust assumptions are the same as [35]. The improved e-voting scheme consists of four phases: prevoting phase, voting phase, postvoting phase, and abstention from voting. Meanwhile, all computations are over a finite field F_p , where p is a secure prime, which is published by AC before the prevoting phase. The list of symbols used in the improved e-voting scheme is shown in Table 1, and Figure 2 shows the specific voting process in voting phase, postvoting phase, and abstention from voting.

4.1. Prevoting Phase. Assume that there are n voters V_1, \dots, V_n and m candidates C_1, \dots, C_m .

4.2. Voting Phase. In this phase, the session keys are negotiated freely among voters, and VS computes the masked

values. Then, each voter casts his favourite candidates and gets the credential in a face-to-face way. The process is described as follows:

Step 1. Every voter V_i , ($i = 1, 2, \dots, n$) registers to AC. VS generates a temporary identity ID_i , ($i = 1, 2, \dots, n$) for V_i , and no one knows the relationship between V_i and ID_i .

Step 2. V_i negotiates session key k_{ij} , ($i, j \in \{1, 2, \dots, n\}, i \neq j$) with other $\beta_i \in \{1, 2, \dots, n-1\}$ voters in a set and then obtains his session key list $\{k_{i1}, k_{i2}, \dots, k_{i\beta_i}\}$. For example, there are three voters $\{V_1, V_2, V_3\}$ in a set, V_1 and V_2 share the session key k_{12} , and V_1 and V_3 share the session key k_{13} . Then, V_1, V_2 , and V_3 can, respectively, obtain their session key lists $\{k_{12}, k_{13}\}$, $\{k_{21}\}$, and $\{k_{31}\}$, where $k_{12} = k_{21}$, $k_{13} = k_{31}$.

Step 3. V_i sends his session key list to VS via a secure channel. VS computes $\lambda_{i,l} = \sum_{j=1, j \neq i}^{\beta_i} (ID_i - ID_j)H(k_{ij} | l)$, ($i \in \{1, 2, \dots, n\}, l \in \{1, 2, \dots, m\}$) and then divides $\lambda_{i,l} = \sum_{j=1}^n b_{l,j}$, where $H: Z_p^* \rightarrow Z_p^*$ is a secure cryptographic hash function, which is published and $b_{l,1}, b_{l,2}, \dots, b_{l,n}$ are n random numbers. Afterwards, VS computes masked values $\sum_{j=1}^n b_{1,i} = B_{1,i}, \sum_{j=1}^n b_{2,i} = B_{2,i}, \dots, \sum_{j=1}^n b_{m,i} = B_{m,i}$, ($i = 1, 2, \dots, n$). Meanwhile, the masked values of the voter V_i can be represented as the list $\{B_{1,i}, B_{2,i}, \dots, B_{m,i}\}$.

Step 4. V_i casts his favourite candidates. If the candidate C_j ($j = 1, \dots, m$) is selected, $a_{i,j} = 1$; otherwise, $a_{i,j} = 0$. According to the computation of masked values for the voter V_i , VS constructs the shared polynomial $f_i(x) = a_{i,0} + (B_{1,i} + a_{i,1})x + (B_{2,i} + a_{i,2})x^2 + \dots + (B_{m,i} + a_{i,m})x^m \bmod p$, where $a_{i,0} \neq 0$ is a random number.

Step 5. Assume that the IDs of candidates C_j ($j = 1, \dots, m$), the voter V_i , and VS are, respectively, x_j, x_{m+1} and x_{m+2} . VS computes $m+2$ shares $(x_j, y_{i,j})$, ($j = 1, 2, \dots, m+2$) by the polynomial in Step 4. Then, VS stores the share $(x_{m+1}, y_{i,m+1})$ and sends shares $(x_j, y_{i,j})$, ($j = 1, 2, \dots, m$) to corresponding candidates C_j ($j = 1, \dots, m$) and credential $CR_i = \{a_{i,0}, x_{m+2}, y_{i,m+2}\}$ to the voter V_i .

4.3. Postvoting Phase. In this phase, VS and all candidates reconstruct polynomial together by the sum of shares, and each participant can obtain the correct tallying result by verifying the constant coefficient of the reconstructed polynomial. The steps are given as follows:

Step 1. All voters are published, and each voter V_i , ($i = 1, 2, \dots, n$) publishes their $a_{i,0}$, ($i = 1, 2, \dots, n$) on the bulletin board.

Step 2. VS and C_j compute $y_j = \sum_{i=1}^n y_{i,j}$ and publish the points (x_j, y_j) , ($j = 1, 2, \dots, m+1$). Then, each participant can recover the polynomial $F(x) = a_0 + a_1x + a_2x^2 + \dots + a_mx^m \bmod p$, where $a_j = \sum_{i=1}^n a_{i,j}$, ($j = 0, 1, \dots, m$). Finally, VS publishes the aggregated ballots $\{a_1, a_2, \dots, a_m\}$ of n voters on the bulletin board.

TABLE 1: List of notations.

Symbol	Significance
n	The number of voters
m	The number of candidates
V_i	i -th voter
C_j	j -th candidate
ID_i	Temporary identification of V_i , ($i \in \{1, 2, \dots, n\}$)
k_{ij}	Session key of V_i , ($i \in \{1, 2, \dots, n\}$)
β_i	The number of session keys negotiated with V_i , ($i \in \{1, 2, \dots, n\}$)
$\lambda_{i,l}$	A value can be used to compute masked values
$B_{j,i}$	j -th masked value of the i -th shared polynomial
$a_{i,j}$	A ballot from V_i , ($i \in \{1, 2, \dots, n\}$) for C_j , ($j \in \{1, 2, \dots, m\}$)
$(x_j, y_{i,j})$	The shares computed by the i -th shared polynomial
CR_i	The credential of V_i , ($i \in \{1, 2, \dots, n\}$)
y_j	The sum of shares computed by C_j , ($j \in \{1, 2, \dots, m\}$)
a_j	The total number of ballots for C_j , ($j \in \{1, 2, \dots, m\}$)

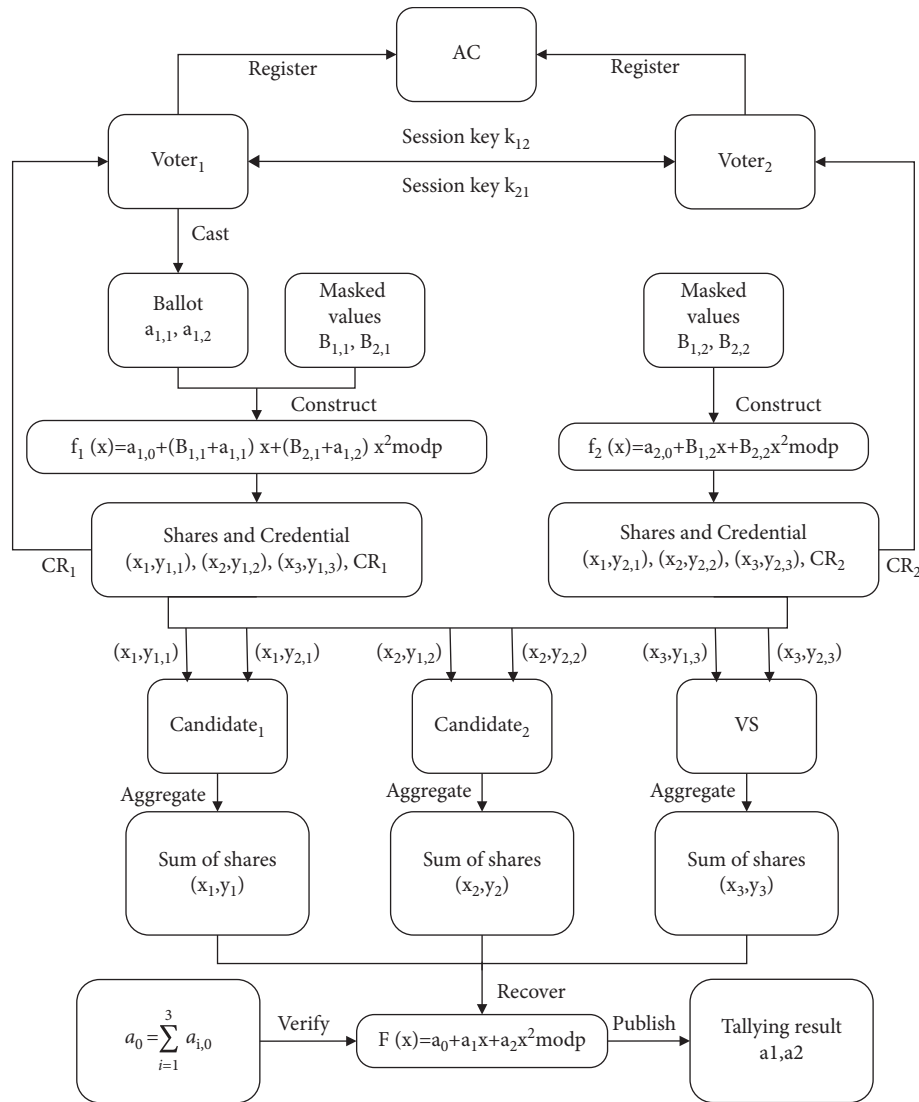


FIGURE 2: The voting phase, postvoting phase, and abstention from voting of two voters and two candidates in the improved e-voting system. Assume the Voter₂ abstains from voting.

Step 3. Each participant verifies the correctness of the aggregated ballots by the equation $a_0 = \sum_{i=1}^n a_{i,0}$. If the verification is not true, VS and all candidates are asked to check their publishing information and reconstruct the polynomial again.

4.4. Abstention from Voting. In the real-life case, abstention from voting is widespread. For obtaining the correct tallying result, when there are abstainers in voting phase, VS sets their ballots as 0 and then constructs the shared polynomials only using the masked values. In proposed e-voting scheme, if the candidate C_j , ($j \in \{1, \dots, m\}$) obtains the supporting ballot of the voter V_i , ($i \in \{1, 2, \dots, n\}$), $a_{i,j} = 1$. For one abstainer, it can consider that all candidates do not get the supporting ballot from the abstainer. VS sets $a_{i,j} = 0$, ($i \in \{1, 2, \dots, n\}$, $j = 1, \dots, m$). In this way, the voting system not only ensures the normal voting process but also does not make a difference for the voting result. Therefore, the improved e-voting scheme solves the issue of abstainers and guarantees the correctness of the tallying result.

5. Differences of Two Schemes

In this section, the differences are expounded between Liu et al.'s scheme [35] and the improved e-voting scheme. We describe these differences in terms of the following five security requirements:

Coercion resistance: coercion resistance means that no one voter can prove the content of his ballot to others. The requirement is described from the aspect of the voter. When the voter acts as an internal attacker, he cannot disclose the content of the original ballot. In Liu et al.'s scheme, VS constructs the shared polynomial with the ballot of one voter. However, if this voter colludes with at most m candidates, he can recover the shared polynomial to disclose the content of the original ballot and prove which candidate he has voted for. Therefore, Liu et al.'s scheme cannot resist the coercive attack from the internal voter. In the improved e-voting scheme, the masked values are computed and added to the coefficients of the original shared polynomial. In this way, even if the voter recovers the shared polynomial, he also cannot prove the content of his ballot to others. The improved e-voting scheme achieves the security requirement of coercion resistance.

Abstention from voting: in a real election, there are voters who may abstain from voting. In Liu et al.'s scheme, all voters take part in the election. Liu et al.'s scheme does not consider the issue of the abstainer. However, in the improved e-voting scheme, for an abstainer, we set his ballot as 0. In this way, all candidates cannot obtain the supporting ballot from the abstainer, which ensures the correctness of tallying result. Therefore, comparing Liu et al.'s scheme, the improved e-voting scheme considers the issue of abstainer and can obtain the correct tallying result.

Privacy: in Liu et al.'s scheme, the voter can reveal the content of his ballot by colluding with some candidates. The scheme does not meet the privacy requirement in e-voting. However, in the improved e-voting scheme, the computation of the masked values prevents the voter from revealing the content of his ballot. Moreover, the participant can only obtain the tallying result by recovering the polynomial in postvoting phase. Thus, the improved e-voting scheme satisfies the privacy requirement of ballot.

Security: security means that the voting scheme does not rely on the hard problem, such as discrete logarithm and integer factorization, and still can achieve the confidentiality of the voting process. This requirement is described from the aspect of the voting scheme. Without additionally complex conditions, the voting scheme can resist attacks from external and internal adversaries, respectively, and achieve the confidentiality of the original ballot. In Liu et al.'s scheme, it shows that the malicious adversary cannot obtain any voting information from some shares. In fact, according to the executive process of secret sharing, Liu et al.'s scheme does not achieve the confidentiality of the original ballot. First, as described in the security requirement of the coercion resistance, Liu et al.'s scheme cannot resist the coercive attack from an internal adversary. Second, since the number of shareholders exceeds the threshold m of secret sharing in Liu et al.'s scheme. Likewise, Liu et al.'s scheme is also hard to prevent external adversary from obtaining the content of the original ballot. Thus, Liu et al.'s scheme cannot guarantee the requirement of security. However, in the improved e-voting scheme, the shares cannot be served as an advantage for adversary to reveal the content of the original ballot. The improved e-voting scheme guarantees the confidentiality of original ballot forever, which achieves the requirement of security.

Fairness: fairness means that no one can know the ballot information in advance. In Liu et al.'s scheme, the voter can prove the content of his ballot to corresponding some candidates. In this way, these candidates can know partial ballots in advance. Liu et al.'s scheme does not satisfy the fairness requirement in e-voting. However, in the improved e-voting scheme, no one can know any ballot information in advance, and the correct tallying result can be obtained simultaneously by each participant. The improved e-voting scheme achieves the fairness requirement.

6. System Analysis

The system analysis includes the security analysis and the performance analysis.

6.1. Security Analysis. The improved e-voting scheme inherits these security requirements, which are correctness, anonymity, confidentiality, efficiency, noncheating, and

universal verifiability in Liu et al.'s scheme [35]. Moreover, the improved e-voting scheme also achieves coercion-resistant security requirement, which Liu et al.'s scheme does not accomplish by adding masked values to the coefficients of the original shared polynomial. In this way, the content of one ballot can be protected, and corresponding privacy requirement can also be satisfied. Meantime, the improved e-voting scheme considers the abstention from voting, which not only ensures the correct tallying result but also achieves integrity of ballots.

In the following part of this section, we give theoretical analysis and proof of the integrity of ballots, privacy of ballots, multiple-voting detection, fairness, and coercion resistance of the improved e-voting scheme.

6.1.1. Integrity of Ballots

Theorem 1. For voters who submit the session key list, VS should hold their voting information to ensure the integrity of ballots and then achieve the correct tallying result.

Proof. In a real election, the registered voter may give up casting his ballot. In the proposed e-voting system, assume that a voter V_t , ($t \in \{1, 2, \dots, n\}$) gives up voting for a ballot after submitting the session key list to VS.

For simplifying analysis, assume that separately negotiates the session key with the remaining voters in voting phase. First, VS computes and divides λ as follows:

$$\left\{ \begin{array}{l} \lambda_{1,1} = (-H(k_{12}|1)) + \dots + (1-t)H(k_{1t}|1) + \dots + (1-n)H(k_{1n}|1) = b_{1,1} + \dots + b_{1,t} + \dots + b_{1,n}, \\ \dots, \\ \lambda_{1,m} = (-H(k_{12}|m)) + \dots + (1-t)H(k_{1t}|m) + \dots + (1-n)H(k_{1n}|m) = b_{m,1} + \dots + b_{m,t} + \dots + b_{m,n}, \\ \vdots, \\ \vdots, \\ \lambda_{n,1} = (n-1)H(k_{n1}|1) = b_{1,1} + \dots + b_{1,t} + \dots + b_{1,n}, \\ \dots, \\ \lambda_{n,m} = (n-1)H(k_{n1}|1) = b_{m,1} + \dots + b_{m,t} + \dots + b_{m,n}. \end{array} \right. \quad (1)$$

Equation (1) shows $\sum_{i=1}^n \lambda_{i,l} = 0$, ($l \in \{1, 2, \dots, m\}$). Then, VS computes the masked values $B_{1,i} = \sum_{j=1}^n b_{1,i}$, \dots , $B_{m,i} = \sum_{j=1}^n b_{m,i}$, ($i = 1, 2, \dots, n$).

For the abstainer V_t , ($t \in \{1, 2, \dots, n\}$), if VS does not construct his shared polynomial, and the sum of shares can be computed as

$$\left\{ \begin{array}{l} \sum_{i=1, i \neq t}^n f_i(x) = \sum_{i=1, i \neq t}^n a_{0,i} + \left(\sum_{i=1}^n \lambda_{i,1} - \sum_{j=1}^n b_{1,t} + \sum_{j=1}^n a_{i,1} \right) x + \dots + \left(\sum_{i=1}^n \lambda_{i,m} - \sum_{i=1}^n b_{m,t} + \sum_{i=1, i \neq t}^n a_{i,m} \right) x^m \bmod p \\ = \sum_{i=1, i \neq t}^n a_{0,i} + \left(\sum_{j=1}^n b_{1,t} + \sum_{i=1, i \neq t}^n a_{i,1} \right) x + \dots + \left(-\sum_{j=1}^n b_{m,t} + \sum_{i=1, i \neq t}^n a_{i,m} \right) x^m \bmod p. \end{array} \right. \quad (2)$$

According to secret sharing homomorphism, the tallying result is $-\sum_{j=1}^n b_{1,t} + \sum_{i=1, i \neq t}^n a_{i,1}, \dots, -\sum_{j=1}^n b_{m,t} + \sum_{i=1, i \neq t}^n a_{i,m}$, instead of $\sum_{i=1, i \neq t}^n a_{i,1}, \dots, \sum_{i=1, i \neq t}^n a_{i,m}$.

Thus, in order to obtain the correct tallying result, VS should ensure the integrity of ballots. In our voting system, VS sets the ballot of abstainer V_t as 0 and constructs the shared polynomial $f_t(x) = a_{t,0} + \sum_{j=1}^n b_{1,t}x + \dots + \sum_{j=1}^n b_{m,t}x^m \bmod p$. In this way, the sum of shares is $\sum_{i=1}^n f_i(x) = \sum_{i=1}^n a_{0,i} + \sum_{i=1}^n a_{i,1}x + \dots + \sum_{i=1}^n a_{i,m}x^m \bmod p$. The ballot of V_t is 0. Therefore, the tallying result is $\sum_{i=1}^n a_{i,1} = \sum_{i=1, i \neq t}^n a_{i,1}, \dots, \sum_{i=1}^n a_{i,m} = \sum_{i=1, i \neq t}^n a_{i,m}$.

To sum up, in a real election, abstention from voting needs to be considered to ensure the integrity of ballots and obtain the correct tallying result. \square

6.1.2. Privacy of Ballots

Theorem 2. No one voter or candidate can reveal any voting information.

Proof. In proposed voting scheme, assume the voter V_k , ($k \in \{1, 2, \dots, n\}$) colludes with m candidates after receiving credential and then recovers the shared polynomial $f_k(x) = a_{k,0} + (\sum_{j=1}^n b_{1,k} + a_{k,1})x + \dots + (\sum_{j=1}^n b_{m,k} + a_{k,m})x^m \bmod p$. However, the voting information $a_{k,1}, \dots, a_{k,m}$ cannot be revealed because of the masked values $\sum_{j=1}^n b_{1,k}, \dots, \sum_{j=1}^n b_{m,k}$. Therefore, in the proposed e-voting scheme, even if a voter or candidate has enough computing power, he also cannot reveal any voting

information. The scheme achieves the privacy requirement of ballots. \square

6.1.3. Multiple-Voting Detection

Theorem 3. *A legal voter only can submit ballot to VS once.*

Proof. In the proposed e-voting system, assume the legal voter V_h , ($h \in \{1, 2, \dots, n\}$) submits ballot to VS twice. In this way, VS can construct two shared polynomials $f_{h1}(x)$ and $f_{h2}(x)$ for V_h according to twice different submissions. Then, the voter V_h can receive two credentials, and each candidate can receive $n+1$ shares. However, according to the computation of the sum of shares in postvoting phase, each candidate only can hold n shares. Moreover, when all voters are published, each candidate can discover that the number of received shares is not equal to n . Thus, the multiple-voting is detected for the voter V_h . The improved e-voting scheme satisfies the requirement that a legal voter casts a ballot once. \square

6.1.4. Fairness

Theorem 4. *No one candidate can know his own ballot in advance.*

Proof. For knowing the voting information about himself, the candidate is willing to collude with one voter to reveal the content of one ballot after receiving shares. However, in the proof of Theorem 2, it shows that no one voter can leak any privacy of ballots. Thus, it is impossible for one candidate to know the content of the single ballot in advance. Moreover, in postvoting phase, the sums of shares are published on the bulletin board, and any participant can obtain the tallying result. Therefore, the improved e-voting scheme achieves the requirement of fairness in e-voting. \square

6.1.5. Coercion Resistance

Theorem 5. *No one voter can prove the content of his ballot to others.*

Proof. In voting phase, assume that VS constructs the shared polynomial $f_e(x) = a_{e,0} + (B_{1,e} + a_{e,1})x + (B_{2,e} + a_{e,2})x^2 + \dots + (B_{m,e} + a_{e,m})x^m \pmod{p}$ for the voter V_e , ($e \in \{1, 2, \dots, n\}$), and computes shares $(ID_{C_1}, f_e(ID_{C_1})), \dots, (ID_{C_m}, f_e(ID_{C_m})), (ID_{VS}, f_e(ID_{VS}))$ and credential $\{a_{e,0}, ID_e, f_e(ID_e)\}$. The shares $(ID_{C_1}, f_e(ID_{C_1})), \dots, (ID_{C_m}, f_e(ID_{C_m}))$ are sent to corresponding candidates C_1, \dots, C_m , and the credential $\{a_{e,0}, ID_e, f_e(ID_e)\}$ is sent to the voter V_e . For proving the content of the ballot, it is most likely for V_e to recover the shared polynomial $f_e(x)$ by colluding with m candidates. However, different from the Liu et al.'s scheme [35], the improved e-voting scheme masks the relationship between coefficient of shared polynomial and voting information. Even if the voter V_e recovers the shared

polynomial $f_e(x)$, he cannot prove his voting intention $a_{e,1}, \dots, a_{e,m}$ to others. Thus, the scheme achieves the security requirement of coercion resistance. \square

6.2. Performance Analysis. In this section, the performance of our proposed e-voting system is analysed. The prevoting phase has no computation, which only distributes the numbers of the voters and candidates. Thus, we mainly analyse the computation cost of the voting phase and postvoting phase. In addition, we compare our scheme with [32, 33, 35] for partial security requirement and computation complexity, which is shown in Table 2. Moreover, we also, respectively, test the total time cost for the different numbers of voters and candidates by using the 1024-bit session key and the 512-bit shared secret on a laptop with Intel i5 3.1 GHz CPU and 8.00 GB memory, and the result is shown in Figures 3 and 4.

6.2.1. Performance Analysis of the Voting Phase. In the voting phase, the five steps are as follows: registering identification for voters, negotiating session keys among voters, generating masked values, constructing shared polynomials, and computing shares. Meanwhile, the computation cost mainly concentrates upon generating masked values and computing shares. Assume that the computation costs of one masked value and one share are separately expressed as $\text{cost}_{\text{mask}}$ and $\text{cost}_{\text{share}}$. The total computation cost in this phase can be expressed as $\text{cost}_{\text{voting_phase}} = n \cdot m \cdot \text{cost}_{\text{mask}} + n \cdot (m + 2) \cdot \text{cost}_{\text{share}}$.

6.2.2. Performance Analysis of the Postvoting Phase. In postvoting phase, VS and candidates are responsible for computing the sum of shares and then publish. Each participant reconstructs a polynomial to obtain the tallying result and then verifies it. Meanwhile, computing the sum of shares, recovering polynomial, and verifying the tallying result are the main computation cost in this phase. Assume that they are separately expressed as $\text{cost}_{\text{share_sum}}$, $\text{cost}_{\text{recover}}$, and $\text{cost}_{\text{verify}}$. The total computation cost in this phase is $\text{cost}_{\text{post_voting}} = (m + 1) \cdot \text{cost}_{\text{share_sum}} + \text{cost}_{\text{recover}} + \text{cost}_{\text{verify}}$.

From the previously mentioned analysis of two phases, the total computation cost is $\text{cost}_{\text{total}} = \text{cost}_{\text{voting_phase}} + \text{cost}_{\text{post_voting}}$ in the improved e-voting scheme and increases with the numbers of voters and candidates. Thus, the computation complexity is $O(nm)$ in the improved e-voting scheme. In Table 2, it shows that our e-voting scheme simultaneously achieves privacy, verifiability, and coercion resistance, and the computation complexity is less than that mentioned in [32] due to $m \ll n$.

In Figure 3, we select $m = 5$ candidates to test the total time cost with different numbers of voters. As shown in Figure 3, the total time cost of the improved e-voting scheme is slightly higher than the scheme [35]. Liu et al.'s scheme [35] is unaffected by the numbers of voters. Therefore, it has an advantage in total time cost. However, it cannot achieve the security requirement of coercion resistance in e-voting. Although the improved e-voting scheme increases the total

TABLE 2: Comparison of partial security requirement and computation complexity.

Performance	[35]	[32]	[33]	Our scheme
Privacy	No	Yes	Yes	Yes
Coercion resistance	No	No	No	Yes
Verifiability	Yes	Yes	Yes	Yes
Computation complexity	$O(nm)$	$O(n \log n)$	$O(nm)$	$O(nm)$

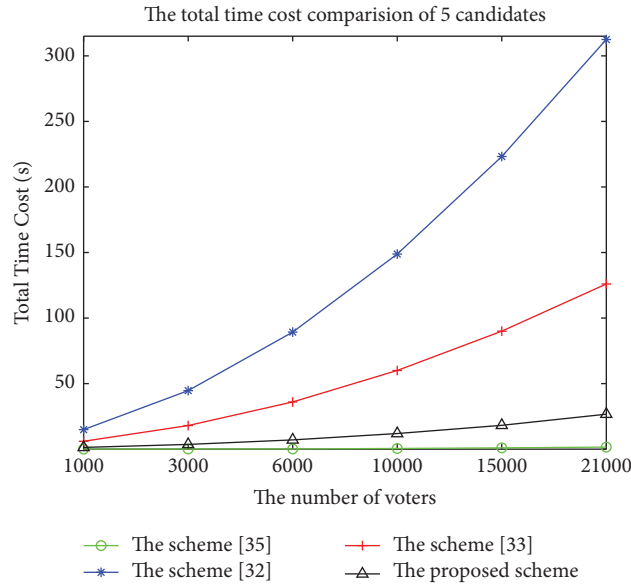


FIGURE 3: The total time cost comparison for four schemes in the case of five candidates: the number of voters is, respectively, 1000, 3000, 6000, 10000, 15000, and 21000.

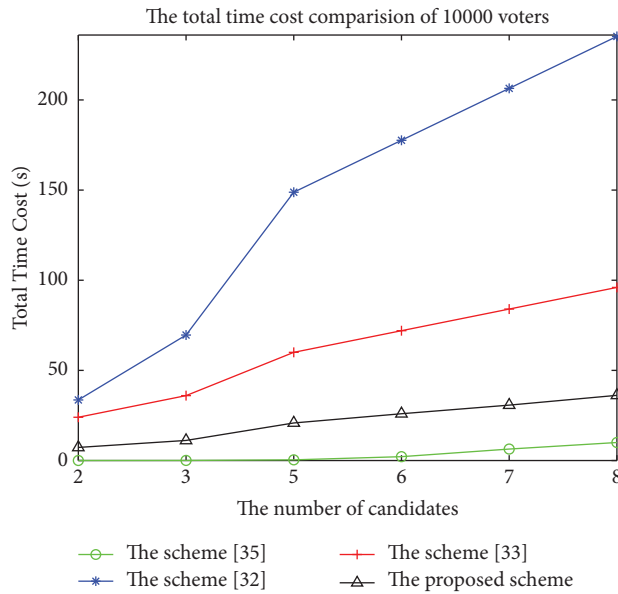


FIGURE 4: The total time cost comparison for four schemes in the case of 10000 voters: the number of candidates is, respectively, 2, 3, 5, 6, 7, and 8.

time cost, it achieves the coercion-resistant security requirement. Moreover, comparing [32, 33], the improved e-voting scheme takes less time overall.

In Figure 4, we select $n = 10000$ voters to test the total time cost with different numbers of candidates. As shown in Figure 4, both the improved e-voting scheme and the scheme [35] increase with the numbers of candidates in the total time cost. Similarly, the total time cost of the improved e-voting scheme is slightly higher than the scheme [35]. In order to achieve the security requirement of coercion resistance, the improved e-voting scheme adds masked values on the basis of the scheme [35]. Thus, the computation cost is higher than Liu et al.'s scheme [35]. However, the total time consumption is still smaller than those in [32, 33].

7. Conclusions

In this paper, we first show that the e-voting scheme recently proposed by Liu et al. in [35] suffers from the coercive attack by internal voter. Then, we proposed an improved e-voting scheme to achieve coercion resistance and solve abstention from voting. Different from the scheme proposed by Liu et al., the improved e-voting scheme uses the sum of the masked value and the value of the original ballot to achieve the construction of shared polynomial. The scheme prevents malicious voter proving the content of his ballot to others, which achieves the security requirement of coercion resistance. Besides, VS sets the ballot of the abstainer as 0, and correct tallying result can be obtained in subsequent voting process. Moreover, theoretical analysis shows that our proposed scheme not only inherits all security requirements of Liu et al.'s scheme but also achieves the integrity of ballots, privacy of ballots, multiple-voting detection, and fairness. The performance evaluation results also indicate our scheme can achieve higher efficiency than other e-voting schemes in terms of total time consumption. In future, with the increasing application of blockchain, we are going to combine blockchain technology to propose novel and secure e-voting scheme.

Data Availability

All data generated or analysed during this study are included in this published article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this study.

Acknowledgments

This work was supported by the Natural Science Foundation of China (grant nos. 61966009, U1811264, U1711263, and 62072133) and Natural Science Foundation of Guangxi Province (grant nos. 2018GXNSFDA281045 and 2018GXNSFD A281040). The authors would like to thank everyone for the guidance of paper writing.



References

- [1] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.
- [2] A. Shankar, P. Pandiaraja, K. Sumathi, T. Stephan, and P. Sharma, "Privacy preserving E-voting cloud system based on ID based encryption," *Peer-to-Peer Networking and Applications*, vol. 14, no. 4, pp. 2399–2409, 2021.
- [3] V. S. Anjima and N. N. Hari, "Secure cloud e-voting system using fully homomorphic elliptical curve cryptography," in *Proceedings of the 2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pp. 858–864, Secunderabad, India, June 2019.
- [4] N. Kshetri and J. Voas, "Blockchain-Enabled E-voting," *IEEE Software*, vol. 35, no. 4, pp. 95–99, 2018.
- [5] S. Panja and B. Roy, "A secure end-to-end verifiable e-voting system using blockchain and cloud server," *Journal of Information Security and Applications*, vol. 59, Article ID 102815, 2021.
- [6] S. Zhang, L. Wang, and H. Xiong, "Chaintegrity: blockchain-enabled large-scale e-voting system with robustness and universal verifiability," *International Journal of Information Security*, vol. 19, no. 3, pp. 323–341, 2020.
- [7] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Towards Trustworthy Elections: New Directions in Electronic Voting*, D. Chaum, M. Jakobsson, R. L. Rivest et al., Eds., Springer Berlin Heidelberg, Berlin, Germany, 2010.
- [8] O. Spycher, R. Koenig, R. Haenni, and M. Schl pfer, "A new approach towards coercion-resistant remote E-voting in linear time," in *Financial Cryptography and Data Security*, G. Danezis, Ed., pp. 182–189, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [9] P. Grontas, A. Pagourtzis, A. Zacharakis, and B. Zhang, "Towards everlasting privacy and efficient coercion resistance in remote electronic voting," in *Financial Cryptography and Data Security*, A. Zohar, I. Eyal, V. Teague et al., Eds., pp. 210–231, Springer Berlin Heidelberg, Berlin, Germany, 2019.
- [10] P. Grontas, A. Pagourtzis, and A. Zacharakis, "Coercion resistance in a practical secret voting scheme for large scale elections," in *Proceedings of the 2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC)*, pp. 514–519, Exeter, United Kingdom, June 2017.
- [11] A. Kiayias, T. Zacharias, and B. Zhang, "An efficient E2E verifiable E-voting system without setup assumptions," *IEEE Security & Privacy*, vol. 15, no. 3, pp. 14–23, 2017.
- [12] R. K sters, J. Liedtke, J. M ller, D. Rausch, and A. Vogt, "Ordinos: a verifiable tally-hiding E-voting system," in *Proceedings of the 2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 216–235, Genoa, Italy, September 2020.
- [13] C. A. Neff, "A verifiable secret shuffle and its application to E-voting," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 116–125, Association for Computing Machinery, Philadelphia, PA, USA, November 2001.
- [14] X. Boyen, T. Haines, and J. M ller, "A verifiable and practical lattice-based decryption mix net with external auditing," in *Computer Security – ESORICS 2020*, L. Chen, N. Li, K. Liang,

- and S. Schneider, Eds., Springer International Publishing, New York, NY, USA, pp. 336–356, 2020.
- [15] N. Islam, K. M. R. Alam, S. Tamura, and Y. Morimoto, “A new e-voting scheme based on revised simplified verifiable re-encryption mixnet,” in *Proceedings of the 2017 International Conference on Networking, Systems and Security (NSysS)*, pp. 12–20, Dhaka, Bangladesh, December 2017.
- [16] C. Culnane, A. Essex, S. J. Lewis, O. Pereira, and V. Teague, “Knights and knaves run elections: internet voting and undetectable electoral fraud,” *IEEE Security & Privacy*, vol. 17, no. 4, pp. 62–70, 2019.
- [17] D. Chaum, “Blind signatures for untraceable payments,” in *Advances in Cryptology*, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds., pp. 199–203, Springer, Boston, MA, USA, 1983.
- [18] A. Fujioka, T. Okamoto, and K. Ohta, “A practical secret voting scheme for large scale elections,” in *Advances in Cryptology — AUSCRYPT ’92*, J. Seberry and Y. Zheng, Eds., Springer Berlin Heidelberg, Berlin, Germany, pp. 244–251, 1993.
- [19] X. Chen, Q. Wu, F. Zhang et al., “New receipt-free voting scheme using double-trapdoor commitment \star ,” *Information Sciences*, vol. 181, no. 8, pp. 1493–1502, 2011.
- [20] M. Kumar, S. Chand, and C. P. Katti, “A secure end-to-end verifiable internet-voting system using identity-based blind signature,” *IEEE Systems Journal*, vol. 14, no. 2, pp. 2032–2041, 2020.
- [21] M. Kumar, C. P. Katti, and P. C. Saxena, “A secure anonymous E-voting system using identity-based blind signature scheme,” in *Information Systems Security*, R. K. Shyamasundar, V. Singh, and J. Vaidya, Eds., Springer International Publishing, New York, NY, USA, pp. 29–49, 2017.
- [22] W. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, and C. Su, “Blockchain-based reliable and efficient certificateless signature for IIoT devices,” *IEEE Transactions on Industrial Informatics*, vol. 14, p. 1, 2021.
- [23] W. Wang, H. Huang, L. Zhang, Z. Han, C. Qiu, and C. Su, “BlockSLAP: blockchain-based secure and lightweight Authentication protocol for smart grid,” in *Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 1332–1338, Guangzhou, China, December 2021.
- [24] R. L. Rivest, L. M. Adleman, and M. L. Dertouzos, “On data banks and privacy homomorphisms,” *Found. Secure Computation*, vol. 4, pp. 169–180, 1978.
- [25] K. Peng, R. Aditya, C. Boyd, E. Dawson, and B. Lee, “Multiplicative homomorphic E-voting,” in *Progress in Cryptology - INDOCRYPT 2004*, A. Canteaut and K. Viswanathan, Eds., pp. 61–72, Springer Berlin Heidelberg, Berlin, Germany, 2005.
- [26] J. Dossogne and F. Lafitte, “Blinded additively homomorphic encryption schemes for self-tallying voting,” *Journal of Information Security and Applications*, vol. 22, pp. 40–53, 2015.
- [27] S. M. Toapanta Toapanta, L. J. Chávez Chalén, J. G. Ortiz Rojas, and L. E. Mafla Gallegos, “A homomorphic encryption approach in a voting system in a distributed architecture,” in *Proceedings of the 2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, pp. 206–210, Shenyang, China, July 2020.
- [28] X. Fan, T. Wu, Q. Zheng, Y. Chen, and X. Xiao, “DHS-voting: a distributed homomorphic signcryption E-voting,” in *Dependability in Sensor, Cloud, and Big Data Systems and Applications*, G. Wang, M. Z. A. Bhuiyan, S. De Capitani di Vimercati, and Y. Ren, Eds., pp. 40–53, Springer Singapore, Singapore, Asia, 2019.
- [29] B. Schoenmakers, “A simple publicly verifiable secret sharing scheme and its application to electronic voting,” in *Advances in Cryptology — CRYPTO’99*, M. Wiener, Ed., Springer Berlin Heidelberg, Berlin, Germany, pp. 148–164, 1999.
- [30] L. Yuan, M. Li, C. Guo, W. Hu, and X. Tan, “A verifiable E-voting scheme with secret sharing,” in *Proceedings of the 2015 IEEE 16th International Conference on Communication Technology (ICCT)*, pp. 304–308, Hangzhou, China, October 2015.
- [31] R. Tso, Z.-Y. Liu, and J.-H. Hsiao, “Distributed E-voting and E-bidding systems based on smart contract,” *Electronics*, vol. 8, no. 4, p. 422, 2019.
- [32] X. Yang, X. Yi, S. Nepal, A. Kelarev, and F. Han, “A secure verifiable ranked choice online voting system based on homomorphic encryption,” *IEEE Access*, vol. 6, pp. 20506–20519, 2018.
- [33] X. Fan, T. Wu, Q. Zheng, Y. Chen, M. Alam, and X. Xiao, “HSE-Voting: a secure high-efficiency electronic voting scheme based on homomorphic signcryption,” *Future Generation Computer Systems*, vol. 111, pp. 754–762, 2020.
- [34] J. Li, X. Wang, Z. Huang, L. Wang, and Y. Xiang, “Multi-level multi-secret sharing scheme for decentralized e-voting in cloud computing,” *Journal of Parallel and Distributed Computing*, vol. 130, pp. 91–97, 2019.
- [35] Y. Liu and Q. Zhao, “E-voting scheme using secret sharing and K-anonymity,” *World Wide Web*, vol. 22, no. 4, pp. 1657–1667, 2019.

Research Article

Intelligent Intrusion Detection Based on Federated Learning for Edge-Assisted Internet of Things

Dapeng Man ¹, Fanyi Zeng,¹ Wu Yang,^{1,2} Miao Yu ³, Jiguang Lv,¹ and Yijing Wang⁴

¹Information Security Research Center, Harbin Engineering University, Harbin 150001, China

²Peng Cheng Laboratory, Guangdong 518055, China

³Beijing Institute of Network Data, Beijing 100031, China

⁴Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100864, China

Correspondence should be addressed to Miao Yu; yumiao8706@163.com

Received 9 June 2021; Revised 27 July 2021; Accepted 9 August 2021; Published 5 October 2021

Academic Editor: Qi Jiang

Copyright © 2021 Dapeng Man et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As an innovative strategy, edge computing has been considered a viable option to address the limitations of cloud computing in supporting the Internet-of-Things applications. However, due to the instability of the network and the increase of the attack surfaces, the security in edge-assisted IoT needs to be better guaranteed. In this paper, we propose an intelligent intrusion detection mechanism, FedACNN, which completes the intrusion detection task by assisting the deep learning model CNN through the federated learning mechanism. In order to alleviate the communication delay limit of federal learning, we innovatively integrate the attention mechanism, and the FedACNN can achieve ideal accuracy with a 50% reduction of communication rounds.

1. Introduction

By connecting the Internet with physical objects (including people and equipment) and transmitting information between objects [1], the Internet of Things (IoT) enables the integration of the real world and the data world, thus making our life more smart and fashionable [2]. Current popular applications of IoT include smart metering, smart cities, smart hospitals, smart agriculture, and smart transportation [2].

IoT and social networking applications are growing too fast, resulting in exponential growth of data generated at the edge of the network [3]. Previously, traditional centralized cloud computing architectures could provide centralized remote services by bringing all data together on a central server [4]. However, due to issues such as network bandwidth limitations and data privacy [5], it is impractical and often unnecessary to still send all data to the remote cloud in the same way as before [3] because this process incurs significant data transmission costs and does not meet the real needs of some low-latency services and applications [4].

As a result, edge computing has emerged as an innovative strategy [5]. Edge computing migrates some network functions and data processing to the edge of the network closer to the end user, where these tasks were previously performed in the core network [6], resulting in the so-called edge-assisted IoT. Edge computing has been identified as a viable option to address the limitations of cloud computing in supporting IoT applications [7, 8]. Compared to cloud computing, edge computing helps deliver efficient network communication services to users with lower latency, more flexible access, and protection of data privacy [7, 9].

While the edge-assisted IoT architecture offers unique features and enhanced quality of service, there are still many challenges to network stability and security, and damage from malicious attacks will undermine the benefits of edge computing [7]. In edge-assisted IoT, the threat factors mainly include information data in edge networks, edge nodes, cloud servers, and other system devices. These threats come from a variety of sources, including malware, hacking, exploited system vulnerabilities, unauthorized access, and

other human factors [10]. Once the intrusion is not detected in time, it will cause incalculable damage to the applications and devices in the IoT, especially the threat to personal safety [4]. The research on edge network security is still in the development stage [4], and most of the previous intrusion detection solutions are recommended to be deployed in the core backbone network [11]; however, this does not meet the real security needs, and due to the turnover of the network structure, the applicability of the intrusion detection technology has raised higher requirements.

The rest of this paper is organized as follows. Section 2 presents an overview of intrusion detection for IoT and an overview of federated learning. In Section 3, we present our proposed intrusion detection method. In Section 4, we present our experimental setup, the utilized performance, our experimental results, and the comparison of our work with other competing approaches. Finally, in Section 5, we present our conclusion and future work.

2. Related Work

2.1. Intrusion Detection for IoT. Intrusion detection is a strong active defense mechanism, and there are many intrusion detection methods [12]. Depending on the data source, intrusion detection techniques can generally be divided into two categories: network-based intrusion detection and host-based intrusion detection [13]. The network-based intrusion detection system is one of the solutions for the early detection of network attacks [14]. According to the different detection mechanisms, intrusion detection technology can be divided into two categories: misuse-based intrusion detection and anomaly-based intrusion detection. Misuse-based intrusion detection uses a set of predefined rules and patterns to detect attacks. Anomaly-based intrusion detection uses a precollected dataset with normal or malicious behaviour labels and uses this dataset to train and test a detection model; any network flow that deviates from the model threshold is flagged as an anomalous entry and reported as an attacker. Due to the excellent classification performance of machine learning, researchers have widely used machine learning methods in anomaly-based intrusion detection, such as the Bayesian model, support vector machine, genetic algorithm, and other machine learning models [13]. However, today's network data present larger, complex, and multidimensional features. When facing high-dimensional data features, traditional machine learning methods need to manually extract a large number of features. The process is complex and the computation is large, which cannot meet the accuracy and real-time requirements of intrusion detection [15].

Deep learning, as an important branch of machine learning, has attracted more and more attention since its theory was proposed. The deep learning model has good advantages in dealing with complex data, and it can automatically extract better representation features from large-scale data. For image classification, feedforward neural networks, especially convolutional neural networks (CNN), have achieved well-known advanced results. For language modelling tasks, recurrent neural networks (RNN), especially LSTM, have also achieved remarkable results [16].

Neural network models such as those described above, with good self-learning capabilities, high-speed optimization, and efficient parallel distributed processing, are also very suitable for handling complex data in network traffic [15].

Currently, the commonly used models in DL-based IoT intrusion detection are DBN, CNN, RNN, GAN, etc., [17, 18]. However, the network structure and training process of most depth models are usually complex, and the model parameters are more, which increases the difficulty and energy consumption of training. CNN has the property of weight sharing, which can effectively avoid the problem of more parameters in complex network structure and reduce the dimension of data by convolution layer and pooling layer, which can effectively reduce the network complexity and speed up the intrusion detection rate. Yang and Wang [19] proposed a wireless network intrusion detection method based on an improved convolutional neural network (ICNN). Simulation results show that the proposed model has acceptable performance on public datasets compared with traditional methods. Considering the limited resources of IoT edge equipment, Stahl et al. [20] proposed a solution to perform CNN model detection tasks in a distributed manner using multiple collaborative edge devices. The core mechanism of their method is to partition the CNN layer where the dominant weight is, distribute the weight data and calculation load evenly on all available devices, and then minimize the task execution time.

Recently, researchers have been exploring whether collaborative intrusion detection systems [21] can be a mainstream option for detecting attacks in large and complex networks, such as the IoT [22]. By detecting real worm datasets with accuracy and overhead, Sharma et al. [11] evaluated the applicability and performance of centralized IDS and purely distributed IDS architectures. Through experiments, they observed that when a large number of edge nodes have attacks, centralized IDS has higher network requirements than distributed IDS and does not have more advantages in detection performance. It can be seen that Stahl et al. [20] did provide a reasonable solution for better completing the detection task on resource-constrained edge devices, but they ignored the protection of data privacy and the threat of malicious nodes.

2.2. Federated Learning. In recent years, the emergence of federal learning (FL) has enabled the deep learning model to effectively train, ensure the security and privacy of data, and effectively solve the problem of data island. The original design purpose of FL is to conduct efficient learning among multiple participants or computing nodes, on the premise of ensuring information security in the process of data exchange. FL can be used as the enabling technology of edge network because it can realize the collaborative training of ML and DL models and also be used for the optimization of edge network [23]. In FL, client devices (edge devices) use local data to train the ML model and then send updated model parameters (rather than raw data) to the cloud server for aggregation.

According to [24], federal learning can be divided into three categories: (1) horizontal federated learning, which applies to scenes with different samples but the same characteristics in multiple datasets, (2) vertical federated learning, which applies to scenarios where multiple datasets have the same sample but different feature spaces, and (3) federated transfer learning applies to scenarios where multiple datasets have different samples and different feature spaces. The federal learning architecture in our subsequent proposed approach belongs to horizontal federal learning.

Since the server needs to interact with the client in the edge network, the wireless connection between them is unstable and unreliable, reducing the number of rounds of communication between the server and the client which is necessary to provide security. The delay in communication is the performance bottleneck of the entire learning framework [25, 26]. For this reason, our goal is to ensure the good performance of the detection mechanism while reducing the cumulative number of communication rounds.

In addition, common machine learning (including deep learning) methods usually follow the assumption that data are independent and identically distributed (IID). However, in real networks, especially under the current edge-assisted IoT architecture, different IoT devices belong to a certain user, enterprise, or application scenario, and thus, data distribution often varies greatly, and due to factors such as user groups and geographical associations, these data often have certain correlations, so the data on edge devices are likely to be non-IID, and this situation deserves strong attention, especially when federated learning is introduced and used to collaborate with tens of thousands of devices and their private data for model training. Using only existing machine learning methods to handle non-IID data for training will lead to low model accuracy and poor model convergence [6]. Therefore, our study focuses on the treatment when the data are non-IID.

In this paper, we propose an intelligent intrusion detection mechanism, FedACNN, based on federated learning-(FL-) aided convolutional neural network (CNN) for edge-assisted IoT. FedACNN completes detection tasks using the CNN model in the framework of federated learning. Our FedACNN uses local datasets and computing resources of edge devices for model training and uploads model parameters to a central server for collaborative training. Compared with traditional centralized learning approaches, FedACNN does not require the transfer of raw data to a central server, ensuring model accuracy while reducing the risk of data leakage. At the same time, we incorporate an attention mechanism, which allows fewer communication rounds and lower detection latency while ensuring the performance of the detection model.

3. Proposed Methods

In this section, we first detail the composition of the learning model CNN and then introduce the common FL algorithms. Finally, we innovatively incorporate attention mechanisms into the FL model to constitute an intrusion detection mechanism based on FL-aided CNN.

3.1. CNN for Intrusion Detection. CNN is a kind of artificial neural network (ANN), which is inspired by the study of visual cortex cells. Its important feature is to reduce the number of parameters in the network by local connection and shared weight, to obtain some degree of affine invariance.

Using the neural network model for detection tasks, the structure of the network has a great influence on the detection results. However, most edge devices are limited by memory and computing resources [27], and they cannot store and execute complex CNN models. Therefore, when we set the hierarchical structure of the CNN model, we mainly start by adjusting parameters and optimizing the structure. While ensuring accuracy, the CNN model structure is relatively simple, as shown in Figure 1.

The CNN model structure is mainly composed of the convolution layer, the pooling layer, and the full connection layer. The first layer is the data input layer. In the training process, the data distribution will change, which will bring difficulties to the learning of the next network. So, after the convolution layer, we use batch normalization to force the data back to the normal distribution with mean 0 and variance 1; on the one hand, it makes the data distribution consistent, and on the other hand, it avoids gradient disappearance. We use the ReLU function as a nonlinear activation function to replace the Sigmoid or tanh function commonly used in traditional neural networks, which can effectively accelerate the speed of network convergence and training. The eighth layer is the pooling layer (down-sampling layer), which mainly conducts the downsampling of the input. The pooling operation reduces the output size of the convolution layer, thus reducing the calculation cost and avoiding overfitting. The commonly used pooling methods include the mean-pooling and the max-pooling, and the max-pooling method is selected in this paper. The ninth layer to the eleventh layer is the fully connected layers; the number of neurons in each layer has been marked in Figure 1. The last layer is the output layer of the network, which is mainly used for classification prediction. We use Softmax as the decision function, which generates a fractional vector for each class and returns the maximum index as the predicted class.

3.2. Federated Learning. The FL structure consists of one server and several clients. In this paper, the term “server” is the remote cloud server, and the term “client” is the edge network entities such as edge nodes and edge devices. The basic idea of FL is “the data does not move, the model moves” [27]. As shown in Figure 2, specifically, the server provides a global shared model and the client downloads the model and uses local datasets for training, while updating model parameters. In each communication between the server and the client, the server will get the current model parameters are distributed to each client (also can be described as the client download server model parameters), after the client training, and then, the updated model parameters are uploaded to the server; the server will aggregate client model parameters through some method, as the

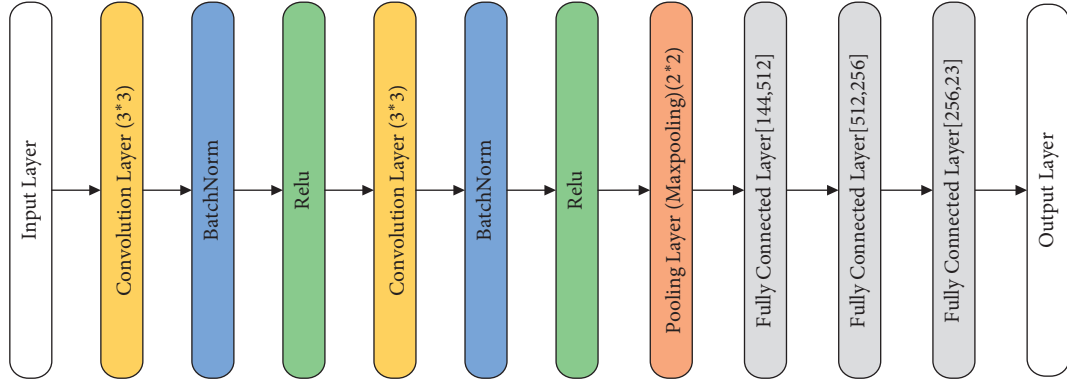


FIGURE 1: Structure of the proposed CNN model.

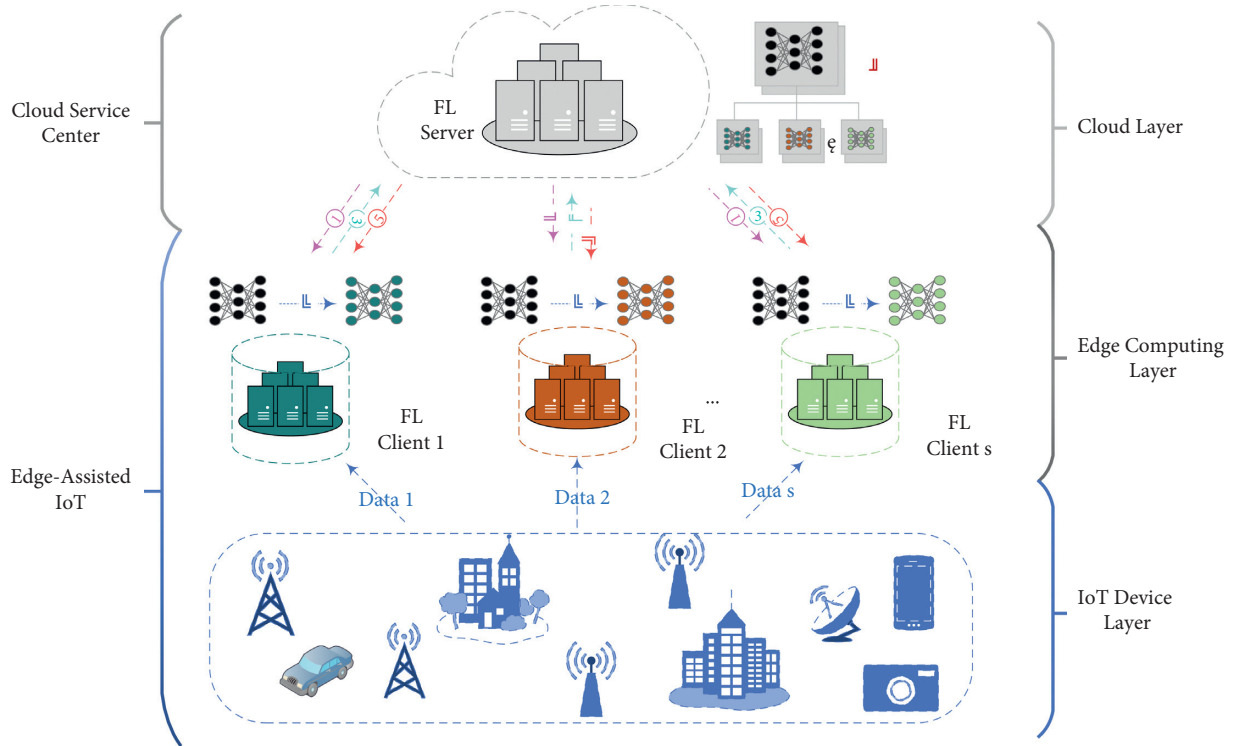


FIGURE 2: Intelligent intrusion detection based on FL-aided CNN for edge-assisted IoT. ① Model initialization; ② local model training and update; ③ upload the updated local model parameters; ④ server aggregates and updates parameters of the global model; ⑤ download the updated global model parameters. Repeat ②–⑤, until model convergence.

updated global model parameters, of this cycle. The server has a variety of aggregation methods for the model parameters uploaded by clients. In [16], the authors proposed a federal averaging algorithm, FedAVG. This algorithm combines the local stochastic gradient descent (SGD) of each client and the execution of model averaging on the server side. In FL scenarios, communication costs are very large constraints, compared with the synchronous random gradient descent algorithm; their FedAVG can greatly reduce the number of communication rounds.

The computation of FedAVG is mainly controlled by three key parameters: C , the fraction of clients, B , local batch size for client update, and E , the number of local epochs. In particular, when $B = \infty$ (i.e., the complete local dataset used

for the client update) and $E = 1$, then FedAVG is equivalent to FedSGD [16]. FedACNN refers to FedAVG as a baseline model framework. See Algorithm 1, for details of the FedAVG algorithm steps.

3.3. Improving FedAVG for Intrusion Detection. Based on FedAVG, inspired by the idea of attention mechanism, we match different importance degrees for the edge nodes involved in collaborative detection. In other words, different clients have different weights. The standard of importance degree is based on the contribution of local model classification performance to the improvement of global model classification performance. The weight is high when the

```

procedure Server:
  initialize  $w_0$ 
  for each round  $t=1, 2, \dots$  do
     $m \leftarrow \max(C \cdot K, 1)$ 
     $S_t \leftarrow$  (random set of  $m$  Clients)
    for each Client  $k \in S_t$  in parallel do
       $w_k(t+1) \leftarrow$  ClientUpdate( $k, w_k(t)$ )
       $w_G(t+1) \leftarrow$  ClientUpdate( $k, w_k(t+1)$ )
procedure Client (k, w)://Run on Client k
   $B \leftarrow$  (split local Client data into batches of size  $B$ )
  for each local epoch  $i$  from 1 to  $E$  do
    for batch  $b \in B$  do
       $w \leftarrow w - \eta * \iota(w; b)$ 
    return  $w$  to Server

```

ALGORITHM 1: FedAVG. Given K clients (indexed by k), B is the local minibatch size, E is the number of local epochs, R is the number of global rounds, C is the fraction of clients, and η is the learning rate.

contribution is large, and vice versa. The server does not only aggregate the average model of clients but also aggregate the weighted model parameters, which can accelerate the convergence of the global model. At the same time, it reduces the adverse effects of model parameters that contribute less to global models. The FedACNN algorithm is described by Algorithm 2. The innovative contributions of this algorithm are mainly reflected as follows.

3.3.1. Server. Like the previous FedAVG algorithm, the server initializes and publishes the model parameters. Each client device downloads the initial model parameters of the server, uses the local dataset for training, and uploads the updated model parameters to the server. After each client device uploads the updated model parameters, the server aggregates the parameters. It is necessary to note that we only use the idea of FedAVG to process non-IID data and apply it to the FedACNN algorithm.

Assuming that the server communicates with the client total t round, in the first round of communication, the server receives updates from s clients, the model parameter after the average aggregation of the server is $W_G(1)$, and $N' = |D_1| + |D_2| + \dots + |D_s|$ is the total number of client data samples. We update the global model on the server side by

$$W_G(1) = \sum_{k=1}^s \frac{|D_k|}{N'} W_k(1), \quad (1)$$

where $W_k(1)$ is the local model parameter of client k in the first round of updating. In the later update aggregation process, an attention mechanism is introduced to aggregate weighted parameters.

3.3.2. Weighted Parameter Aggregation. Inspired by the attention mechanism and FedAGRU [25], FedACNN implements the attention mechanism on the server side, assigns different weights to different clients, and applies the weights to the model parameter aggregation process. The model parameter vector set updated by s clients is expressed

as $W = [W_1, W_2, \dots, W_s]$. Assuming that the model parameter vector W_k updated by the client k is n -dimensional; after the first round of communication ($t > 1$), the Euclidean distance $d(W_G(t), W_k(t))$ between the parameter matrices are calculated by

$$d(W_G(t), W_k(t)) = \sqrt{\sum_{i=1}^n (W_G(t)_i - W_k(t)_i)^2}. \quad (2)$$

Compare the magnitude of d value between the model parameters updated by each client and the model parameters after global aggregation by the server, which is used to measure the contribution of each client's local parameters to the global parametric model optimization. Due to the large dimensional difference between different parameters, the Sigmoid function (see (3)) is used for normalization. The normalized result is $a_k(t)$ (see (4)). Furthermore, using (5) to assign importance $h_k(t)$ to client k ,

$$f_{\text{Sigmoid}} = \frac{1}{1 + e^{-x}}, \quad (3)$$

$$a_k(t) = f_{\text{Sigmoid}}(d(W_G(t), W_k(t))), \quad (4)$$

$$h_k(t) = \frac{s * a_k(t)}{\sum_{k=1}^s a_k(t)}. \quad (5)$$

In the next round of global aggregation, the server will be based on the importance of each client $h_k(t)$ and aggregate model parameters for each the client using (6). This cycle will continue until optimal performance is achieved:

$$W_G(t) = \sum_{k=1}^s \frac{|D_k|}{N^*} h_k(t-1) * W_k(t), \quad (6)$$

where N^* is the total number of data samples for selected clients.

It is important to note that, except for the average aggregation of parameters by the server in the first round of communication, the server aggregates weighted parameters

```

procedure Server:
  initialize  $W_0$ 
  for each round  $t=1, 2, \dots$  do
     $m \leftarrow \max(C \cdot K, 1)$ 
     $S_t \leftarrow$  (random set of  $m$  Clients)
    for each Client  $k \in S_t$  in parallel do
       $W_k(t+1) \leftarrow$  ClientUpdate( $k, W_k(t)$ )
    if  $t=1$  then
       $W_G(1) = \sum_{k=1}^S |D_k|/N' W_k(1)$ 
      Calculate the Client  $K$  importance degree  $h_k(1)$  following equations (3)–(6)
    else
       $W_G(t) = \sum_{k=1}^S |D_k|/N' h_k(t-1) * W_k(t)$ 
      Calculate the Client importance degree  $h_k(t)$  following equations (3)–(6)
      Update parameter  $W_G(t+1), h_k(t+1)$ .
  procedure Client Update (k, w)://Run on Client k
   $B \leftarrow$  (Split local Client data into batches of size B)
  for each local epoch  $i$  from 1 to  $E$  do
    for batch  $b \in B$  do
       $w \leftarrow w - \eta * \nabla(w; b)$ 
    return  $w$  to Server

```

ALGORITHM 2: FedACNN. Given K clients (indexed by k), B is the local minibatch size, E is the number of local epochs, R is the number of global rounds, C is the fraction of clients, and η is the learning rate.

in each subsequent round of communication. According to the contribution of the updated parameters of the client in the previous round to the global model optimization, the importance of each client in each round will be dynamically allocated.

4. Experiments and Results

We use the NSL-KDD dataset to evaluate the model. Our experiments can dynamically select the number of clients, and each client uses local data to train its model and upload updated model parameters to the server for aggregation.

The server used in this experiment is the Windows10 operating system and the processor is Intel(R) Core (TM) i5-10210U CPU@2.11 GHz. We use Python’s deep learning library Pytorch to program FedACNN.

4.1. Dataset Preprocessing. ML algorithms rely on large amounts of data to train models to provide better results. Data is usually stored in storage container devices such as files and databases, which cannot be used directly for training [28]. Before passing the data to the learning model for training, we must preprocess the data.

A key component of an effective intrusion detection system is a good dataset that reflects real-world reality. To verify the effectiveness of the approach proposed in the paper, we conduct experiments using a public intrusion detection dataset-NSL-KDD. NSL-KDD dataset is a commonly used dataset for intrusion detection. The NSL-KDD dataset has advantages over the original KDD CUP 99 dataset, such as it does not include redundant records in the training set, so the classifier is not biased towards more frequent records, etc. The dataset contains normal networks’ behaviour data, as well as four major categories of abnormal

attack data, which are denial-of-service (DoS) attack, user to root (U2R), remote to local attack (R2L), and probing attack (Probe), and their specific descriptions are shown in Table 1. Our work focuses on classifying and detecting four major categories of anomalous attacks. Each sample in the NSL-KDD dataset contains 41 feature attributes and one class identification, and the class identification is used to indicate whether the connection record is normal or a specific type of attack.

To better use the NSL-KDD data as input data for the CNN model, we performed a preprocessing operation on the original dataset. The preprocessing procedure includes numerical, normalization, and visualization.

4.1.1. Numerical. The original NSL-KDD dataset has four character-based feature attributes, namely, protocol, service, flag, and label. We have used the label encoder function to numerically encode each of the four character-based values.

4.1.2. Normalization. After the numerical processing is performed, there is a large difference in magnitude between the values. This situation tends to cause problems such as slower convergence of the network and saturation of neuron outputs; therefore, normalization of the original data is required. We used the Min-Max normalization method (see (7)) to normalize the data to the interval $[0, 1]$:

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad (7)$$

where x^* is the normalized data, x is the original data, x_{\min} is the minimum data value in the current attribute, and x_{\max} is the maximum data value in the current attribute.

TABLE 1: Specific description of each type of attack.

Category	Description	Specific attacks included in the training set	Specific attacks included in the test set
DoS	Sending a large number of packets to the server to make it busy, the attacker tries to prevent legitimate users from accessing the server	back, land, Neptune, pod, smurf, teardrop	apache2, mailbomb, processtable
U2R	The attacker accesses the system through a normal user account and then attempts to gain root access to the system using certain vulnerabilities	buffer-overflow, perl, load module, rootkit	htptunnel, ps, sqlattack, xterm
R2L	The attacker can remotely log into the computer and then use the computer's account and weak password to enter the computer to operate	ftp-write, imap, multihop, phf, guess-passwd, warezclient, spy, warezmaster	sendmail, worm, named, xlock, snmpgetattack, snmpguess, xsnoop
Probe	The attacker purposefully collects information about a computer network to bypass its security controls	ipsweep, Satan, nmap, portsweep	Mscan, saint

4.1.3. *Visualization.* After preprocessing the data, we obtained a numerical dataset with values between $[0, 1]$, and we converted the numerical type data into an $8 * 8$ two-dimensional matrix, with the empty space to be filled with zeros.

In the following experiments, Accuracy, Precision, Recall, and $F1_score$ are selected as the evaluation indexes for evaluating the performance of various methods. The specific description of these evaluation indexes is shown in Table 2. TP and TN correctly classify positive/negative samples; in fact, FP indicates a false alarm that misidentifies a negative sample as a positive sample by mistake negative samples into positive samples and FN means that a positive sample is mistaken for a negative sample, indicating a missing alarm [29].

4.2. *Applicability of FL Model.* We first use Accuracy to evaluate the performance of the centralized learning (CL) model and the federated learning model on NSL-KDD. For a centralized model, we use our proposed local CNN model for experiments. Centralized algorithm uploads data to the server for centralized training. FedAVG [29] is selected as one of the comparison methods of federal learning. In order to better illustrate the advantages of the proposed algorithm, the training models of the comparison federal learning methods are CNN. Reasonable selection of hyperparameters will greatly affect the performance of the algorithm [30]. Under different hyperparameter configurations, we studied the classification performance of the centralized learning model CNN (CL-CNN) and the FL model and determined the reasonable parameters of the algorithms (the specific parameter configuration is shown in the table). The simulation results (see Table 3) show that, under the condition of 40 rounds of iteration, CL-CNN has the highest accuracy of 99.65%; this is because the CL-CNN model has more complete datasets, so its accuracy can be used as the theoretical upper limit of the accuracy of the federal learning model. Compared with FedAVG, FedACNN has higher accuracy of 99.12%. Although it is slightly lower than CL-CNN, it is within an acceptable gap. Experiments show that collaborative training of federal learning can achieve ideal accuracy while protecting data

privacy. In other words, FedACNN sacrifices a bit of accuracy to protect data privacy.

4.3. *Classification Performance Evaluation.* After verifying the applicability of the federal learning model, we use Accuracy to measure the overall classification performance of FedACNN without limiting the number of communication rounds, and use Recall, Precision, and $F1_score$ to measure the specific classification performance of FedACNN. First, we use FedAVG to complete the comparative experiment. The hyperparameters of the FL model are set as $B = 128$, $E = 5$, $K = 10$, $C = 1$, and $\eta = 1e - 2$. Through the experimental results shown in Table 4, we observe that since there are fewer samples of R2L and U2R types of attacks, the indicators of these two types of attacks are lower than those of other types, followed by Probe. Since there are too few samples of the U2R class, FedAVG cannot accurately classify such attacks, while FedACNN has certain classification ability for such attacks. FedACNN also has obvious advantages in detecting other types of attacks, and its overall classification accuracy can reach 99.76%. Figure 3 shows the detection performance of FedACNN for various attacks.

We compare the results of the NSL-KDD dataset after applying traditional machine learning for training and testing with FedACNN and also compare the classification performance of typical deep learning models along with several current innovative mechanisms. The traditional machine learning models used for comparison include Random Forest (RF), SVM, and Naive Bayes [31], typical deep learning models include LeNet-5, DBN, and RNN [19], and innovative mechanisms include IoTDefender [24] and IBWNIDM [19]. Combined with the analysis of the experimental results in [31], it is clear that the RF classifier has the best classification performance compared to SVM and Naive Bayes, and its detection accuracy can reach 99.1% for normal traffic, 98.7% for DoS, and 97.5%, 96.8%, and 97.6% for U2R, R2L, and Probe, respectively. Comparing the above results with the detection results produced by our model, FedACNN has higher detection accuracy for normal traffic and various types of attacks than the three types of traditional machine learning models mentioned above (see Table 5).

TABLE 2: Method evaluation metrics.

Metrics	Narrative description	Equation to describe
Accuracy	The percentage of correct classification records in total records	$TP + TN / TP + TN + FP + FN$
Precision	The percentage of the number of prediction pairs of this category to all the prediction number of this category	$TP / TP + FP$
Recall	The percentage of the number of prediction pairs of this category to all the number of this category	$TP / TP + FN$
F1_score	A measure of classification problems is a harmonic average of precision and recall	$2 * Precision * Recall / Precision + Recall$

TABLE 3: Comparison of overall classification accuracy between the CL model and the FL model.

Method ($B = 128, E = 1, K = 10, C = 1, \eta = 1e-2$)	Accuracy (%) ($R = 10$)	Accuracy (%) ($R = 20$)	Accuracy (%) ($R = 40$)
CL-CNN	98.97	99.40	99.65
FedAVG (CNN)	98.13	98.58	98.90
FedACNN (proposed)	98.73	99.02	99.12

TABLE 4: Comparison of specific classification performance of different methods.

	FedACNN			FedAVG (CNN)		
	Recall	Precision	F1_score	Recall	Precision	F1_score
DoS	99.92	99.93	99.93	99.79	99.77	99.78
U2R	45.59	90.00	60.52	23.07	63.91	33.90
R2L	85.54	90.15	87.78	69.25	80.00	74.24
Probe	88.79	93.86	91.26	75.84	84.08	79.75
Normal	99.81	99.43	99.62	99.49	98.99	99.23
All	83.92	94.67	88.97	73.49	85.35	78.98

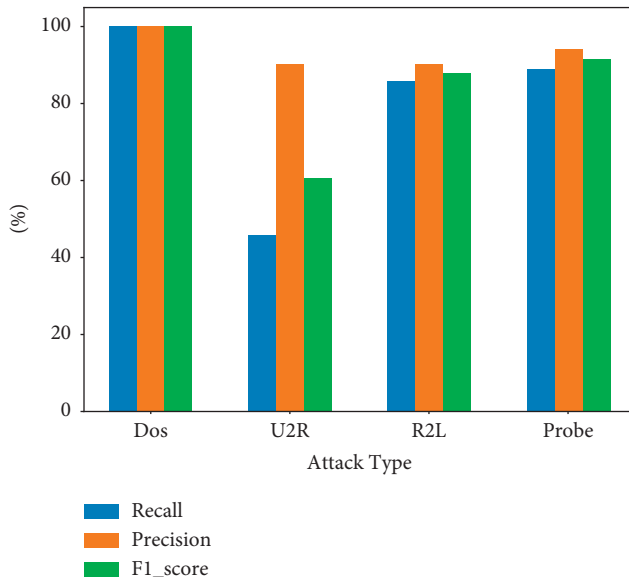


FIGURE 3: The detection performance of FedACNN for various attacks.

As shown in [19], IBWNIDM is a detection mechanism based on the improved CNN model, and its overall detection accuracy reaches 95.36%, which is already higher than the typical deep learning models, LeNet-5, DBN, and RNN.

Comparing the above results with the detection performance of our designed FedACNN, the overall detection accuracy of FedACNN is 4% higher more than. The authors of [24] claim that their IoTDefender is the first framework to apply federated transfer learning to 5G IoT IDSs, but the detection accuracy of IoTDefender for the NSL-KDD dataset is only 81.99%, which is much worse than that of our FedACNN (see Table 6). Through the above comparative analysis, we can learn that FedACNN has better detection performance.

4.4. Communications' Efficiency Assessment. Through the relationship between the number of communication rounds and the accuracy, we evaluate the learning speed of the proposed algorithm and compare the classification accuracy performance of FedACNN and FedAVG (CNN) under different communication rounds. Figures 4 and 5 show the relationship between the number of communication rounds and accuracy. Compared with FedACNN and FedAVG, FedACNN has better performance under the same communication round. In other words, FedACNN needs fewer communication rounds to achieve model convergence. FedACNN can achieve the accuracy of FedAVG in 40 rounds of communication when communicating for 20 rounds, and the number of communication rounds is reduced by 50%. This is due to the addition of the attention mechanism in FedACNN, so the server aggregation of client model parameters is no longer an average aggregation, but a

TABLE 5: Comparison of specific classification performance with traditional machine learning methods.

Classification algorithm	Accuracy for different classes of attacks				
	Normal	Dos	U2R	R2L	Probe
Random forest	99.1	98.7	97.5	96.8	97.6
SVM	98.1	97.8	93.7	91.8	90.7
Naive Bayes	70.3	72.7	70.7	69.8	70.9
FedACNN (proposed)	99.8	99.9	99.1	99.0	99.2

TABLE 6: Comparison of specific classification performance with traditional machine learning methods.

Classification algorithm	LeNet-5	DBN	RNN	IBWNIDM	IoTDefender	FedACNN
Accuracy (%)	86.54	92.45	93.08	95.36	81.99	99.76

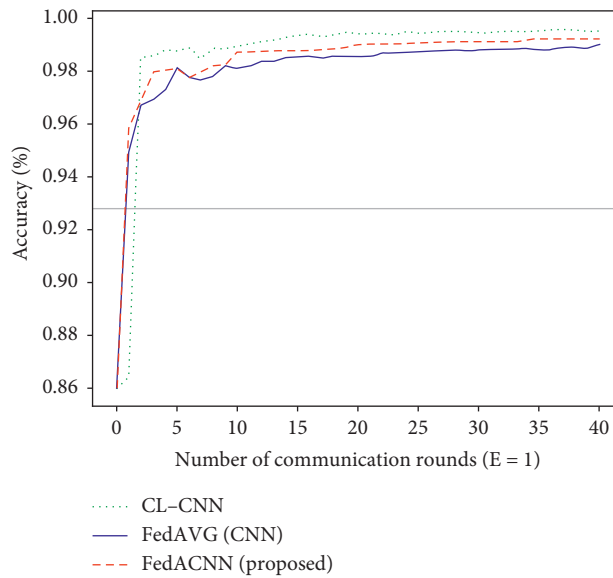


FIGURE 4: Relationship between number of communication rounds and accuracy ($B = 128, E = 1, K = 10, C = 1,$ and $\eta = 1e - 2$).

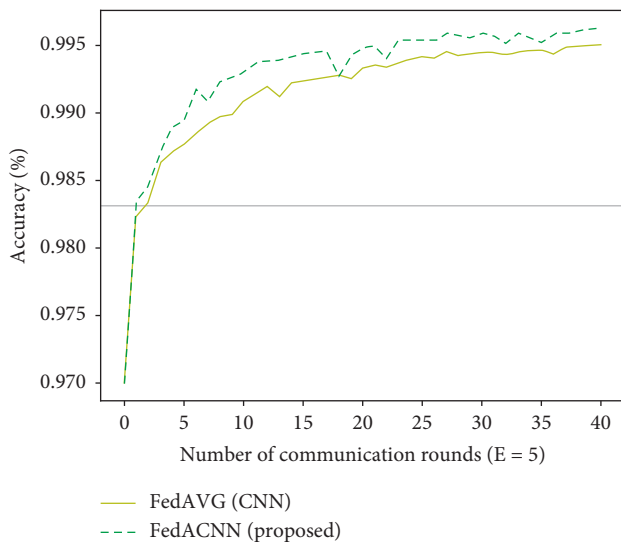


FIGURE 5: Relationship between number of communication rounds and accuracy ($B = 128, E = 5, K = 10, C = 1,$ and $\eta = 1e - 2$).

weighted aggregation according to different importance, allowing parameters that contribute more to the overall model to play a greater role, thus speeding up convergence and reducing the number of communication rounds.

4.5. Comparison of Classification Performance between the Local Model and Global Collaboration Model. We use FedACNN to complete this part of the experiment. On the one hand, each client model uses the local dataset for training, and the local iterations are 40, 120, and 200 rounds, namely, $E = 40, 120,$ and 200 . On the other hand, the FL mechanism is used for collaborative training, the parameters $E = 1,$ and R is 40, 120, and 200, respectively. In each iteration number (communication rounds), we use Accuracy and Precision as evaluation indexes to evaluate the classification performance of these two mechanisms. The results in Table 7 show that due to the incomplete characteristics of the dataset, each client model shows overfitting. In the same number of iterations, the overall classification performance

TABLE 7: Comparison of classification performance between the local model and the global collaboration model.

Model $B=128, K=10, C=1, \eta=1e-2$	Local model: $E=40$ FL model: $E=1, R=40$		Local model: $E=120$ FL model: $E=1, R=120$		Local model: $E=200$ FL model: $E=1, R=200$	
	Accuracy (%)	Precision (%)	Accuracy (%)	Precision (%)	Accuracy (%)	Precision (%)
Client model01	99.07	92.38	99.22	93.04	99.54	93.99
Client model02	98.97	91.95	99.09	91.90	99.53	92.30
Client model03	99.06	91.31	99.27	92.01	99.57	94.53
Client model04	99.08	91.67	99.15	91.58	99.53	91.12
Client model05	99.06	90.11	99.21	92.92	99.56	93.78
Client model06	98.96	91.64	99.23	92.19	99.56	91.03
Client model07	98.98	91.50	99.18	93.01	99.58	94.17
Client model08	98.84	90.89	99.11	90.70	99.36	88.59
Client model09	99.06	90.86	99.22	91.52	99.57	91.62
Client model10	99.03	91.42	99.19	92.20	99.57	91.27
FedACNN	99.12	92.74	99.45	93.06	99.63	94.56

of each client under the FL mechanism is much better than that only using local data to train in the local. This is because when the client only uses local data for model training, the dataset is small and the number of samples is limited. FedACNN, through the federal learning mechanism where the Clients update parameters and the server aggregates them and sends them to the clients, makes each client no longer only use small dataset for training, but use the current global optimal parameters for training, which makes the model detection performance trained by each client better.

5. Conclusions

In this paper, we propose an intelligent intrusion detection mechanism for edge-assisted IoT, FedACNN, which is based on FL-aided CNN. Under the premise of protecting data privacy, the FedACNN can complete the intrusion detection task with relatively ideal performance, and the overall classification accuracy of FedACNN for attack data can reach 99.76%. Since we innovatively integrate the attention mechanism, FedACNN can obtain higher detection accuracy with less communication overhead. By comparing the detection results on the USL-CUP dataset, FedACNN has better accuracy performance than three traditional machine learning models, three typical deep learning models, and two innovative mechanisms. Compared with FedAVG, the number of communication rounds is reduced by 50%.

We believe and expect that FedACNN can make some contributions to the research on security protection for edge-assisted IoT. In the future, we will conduct intrusion detection research on encrypted traffic data of edge-assisted IoT and look forward to making greater contributions to protecting edge-assisted IoT.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant nos. 61771153, 61831007, and 61971154) and the Fundamental Research Funds of the Central Universities (Grant no. 3072020CF0601).

References

- [1] S. S. Swarna and R. Ratna, "Investigation of machine learning techniques in intrusion detection system for IoT network," in *Proceeding of the 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 1164–1167, IEEE, Thoothukudi, India, December 2020.
- [2] S. Anand and A. Sharma, "Assessment of security threats on IoT based applications," *Materials Today: Proceedings*, 2020, In press.
- [3] D. Wu, J. Yan, H. Wang, and R. Wang, "Multiattack intrusion detection algorithm for edge-assisted internet of Things," in *Proceeding of the International Conference on Industrial Internet (ICII)*, pp. 210–218, IEEE, OL, USA, November 2019.
- [4] S. Wang, T. Tuor, T. Salonidis et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.
- [5] A. Alwarafy, K. A. A. Thelaya, M. Abdallah, J. Schneider, and M. Hamdi, "A survey on security and privacy issues in edge-computing-assisted internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4004–4022, 2021.
- [6] W. Y. B. Lim, C. Luong, T. Hoang, Y. Jiao, and C. Liang, "Federated learning in mobile edge networks: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.
- [7] F. Lin, Y. Zhou, X. An, I. You, and K. R. Choo, "Fair resource allocation in an intrusion-detection system for edge computing: ensuring the security of internet of Things devices," *IEEE Consumer Electronics Magazine*, vol. 7, no. 6, pp. 45–50, 2018.
- [8] G. W. Cassales, H. Senger, E. R. de Faria, and A. Bifet, "IDSA-IoT: an intrusion detection system Architecture for IoT networks," in *Proceeding of the Symposium on Computers and Communications (ISCC)*, pp. 1–7, IEEE, 2019.
- [9] C. Wang, D. Wang, G. Xu, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy

- for industry 4.0,” *SCIENCE CHINA: Information Sciences*, vol. 64, 2020.
- [10] Q. Cui, C. Huang, Z. Zhu et al., “Lightweight security mechanism based on heterogeneous construction for virtualized edge system: a markov decision process approach,” in *Proceedings of the International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1003–1009, Nanjing, Jiangsu, China, October 2020.
- [11] R. Sharma, C. A. Chan, and C. Leckie, “Evaluation of centralised vs distributed collaborative intrusion detection systems in multi-access edge computing,” in *Proceedings of the 2020 IFIP Networking Conference (Networking)*, pp. 343–351, Paris, France, June 2020.
- [12] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, “Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques,” *Mobile Networks and Management (MONAMI)*, vol. 235, pp. 30–44, 2017.
- [13] R. Zhao, Y. Yin, Y. Shi, and Z. Xue, “Intelligent intrusion detection based on federated learning aided long short-term memory,” *Physical Communication*, vol. 42, 2020.
- [14] S. Hosseini, “A new machine learning method consisting of GA-LR and ANN for attack detection,” *Wireless Networks*, vol. 26, no. 6, pp. 4149–4162, 2020.
- [15] A. P. D. C. Kelton, J. P. Papa, C. O. Lisboa, R. Munoz, and V. H. C. D. Albuquerque, “Internet of Things: a survey on machine learning-based intrusion detection approaches,” *Computer Networks*, vol. 151, pp. 147–157, 2019.
- [16] B. McMahan, E. Moore, D. Ramage et al., “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 54, pp. 1273–1282, Ft. Lauderdale, FL, USA, April 2017.
- [17] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang, “Understanding node capture attacks in user authentication schemes for wireless sensor networks,” *IEEE Transactions on Dependable and Secure Computing*, 2020.
- [18] X. Yuan, C. Li, and X. Li, “DeepDefense: identifying DDoS attack via deep learning,” in *Proceeding of the International Conference on Smart Computing (SMARTCOMP)*, pp. 1–8, IEEE, Hong Kong, China, May 2017.
- [19] H. Yang and F. Wang, “Wireless network intrusion detection based on improved convolutional neural network,” *IEEE Access*, vol. 7, Article ID 64366, 2019.
- [20] R. Stahl, A. Hoffman, D. M. Gritschneider et al., “DeeperThings: fully distributed CNN inference on resource-constrained edge devices,” *International Journal of Parallel Programming*, vol. 49, 2021.
- [21] J. Arshad, M. A. Azad, M. M. Abdeltaif, and K. Salah, “An intrusion detection framework for energy constrained IoT devices,” *Mechanical Systems and Signal Processing*, vol. 136, 2020.
- [22] X. An, X. Zhou, X. Lü, F. Lin, and L. Yang, “Sample selected extreme learning machine based intrusion detection in fog computing and MEC,” *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 7472095, 2018.
- [23] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, “Federated learning: a survey on enabling technologies, protocols, and applications,” *IEEE Access*, vol. 8, Article ID 140699, 2020.
- [24] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang, “IoTDefender: a federated transfer learning intrusion detection framework for 5G IoT,” in *Proceeding of the 14th International Conference on Big Data Science and Engineering (BigDataSE)*, pp. 88–95, IEEE, Erode, India, April 2021.
- [25] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan, “Intrusion detection for wireless edge networks based on federated learning,” *IEEE Access*, vol. 8, Article ID 217463, 2020.
- [26] N. A. A. A. Marri, B. S. Ciftler, and M. M. Abdallah, “Federated mimic learning for privacy preserving intrusion detection,” in *Proceeding of the International Black Sea Conference on Communications and Networking (Black-SeaCom)*, pp. 1–6, IEEE, Odessa, Ukraine, May 2020.
- [27] S. Qiu, D. Wang, G. Xu, and S. Kumari, “Practical and provably secure three-factor Authentication protocol based on extended chaotic-maps for mobile lightweight devices,” *IEEE Transactions on Dependable and Secure Computing*, 2020.
- [28] A. S. Ahanger, S. M. Khan, and F. Masoodi, “An effective intrusion detection system using supervised machine learning techniques,” in *Proceeding of the 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1639–1644, IEEE, Erode, India, April 2021.
- [29] P. Singh, J. Jayakumar, A. Pankaj, and R. Mitra, “Edge-detect: edge-centric network intrusion detection using deep neural network,” in *Proceeding of the 18th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, IEEE, Las Vegas, Nevada, January 2021.
- [30] D. Yuan, K. Ota, M. Dong et al., “Intrusion detection for smart home security based on data augmentation with edge computing,” in *Proceeding of the International Conference on Communications (ICC)*, pp. 1–6, IEEE, Dublin, Ireland, June 2020.
- [31] S. Revathi and A. Malathi, “A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection,” *International Journal of Engineering Research and Technology*, vol. 2, pp. 1848–1853, 2013.

Research Article

IoT-Based Autonomous Pay-As-You-Go Payment System with the Contract Wallet

Shinya Haga ¹ and Kazumasa Omote ^{1,2}

¹University of Tsukuba, Tsukuba, Japan

²National Institute of Information and Communications Technology, Tokyo, Japan

Correspondence should be addressed to Shinya Haga; s2120545@s.tsukuba.ac.jp

Received 10 June 2021; Revised 22 August 2021; Accepted 27 August 2021; Published 24 September 2021

Academic Editor: Chunhua Su

Copyright © 2021 Shinya Haga and Kazumasa Omote. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In recent years, increasingly many companies have entered the pay-as-you-go business because it has become easier to monitor services constantly due to the development and increase in the number of Internet of Things (IoT) devices. Research is in progress to introduce cryptographic assets into the payment, but if a private key is stolen, the cryptographic assets associated with it can be stolen. To address this issue, this paper proposes a secure automated payment system using contract wallets. This method ensures the security of cryptographic assets even if the private key is stolen. Secure automated payments are enabled by issuing transactions from IoT devices and using internal transactions to link contract wallets and smart contracts that handle the payment of cryptographic assets. Furthermore, the effectiveness of the proposed system is demonstrated on the Ethereum blockchain as a proof of concept, and the cost of gas is measured.

1. Introduction

In recent years, the effects of Internet of Things (IoT) devices on our lives have increased due to the rise of smart cities and smart homes. According to a survey conducted by Statista, Germany [1], the global penetration of IoT devices is expected to reach 50 billion units by 2030, and the market size will continue to grow thereafter. Consequently, increasingly many companies are offering pay-as-you-go services that provide income stability because it has become easier to monitor the operating status of machines and services in real time. In addition, many studies have been conducted on the use of smart contracts for the payment by IoT devices.

However, the wallets utilized for cryptographic asset payments are often managed in the hot wallet state, where the account is connected to the internet. If a private key is compromised and cryptographic assets are stolen, it is essentially impossible to recover the assets. The Coincheck incident [2] is said to have been a case of asset theft as the private key was stolen.

There are smart contracts known as contract wallets that minimize the loss of cryptographic assets caused by private key theft. A contract wallet is a smart contract for managing cryptographic assets securely; thus, users can securely hold these assets. If assets are deposited in the wallet, even if the private key of the wallet owner is stolen, the damage can be minimized because assets can be managed programmatically (e.g., an upper limit can be set on the amount of assets transferred). Typical contract wallets include Argent (Argent, “Argent—the best Ethereum wallet for DeFi,” 2020, <https://www.argent.xyz/>), Dapper (Dapper Labs, “Dapper—your account manager for all things flow,” 2020, <https://www.meetdapper.com/>), and Gnosis Safe (Gnosis, “Gnosis Safe: Overview,” 2020, <https://gnosis-safe.io/>).

Contribution: traditionally, smart contracts and contract wallets have been independent of each other. In this study, we developed a system that enables secure automated payments with IoT devices by using internal transactions to link contract wallets and smart contracts that handle the payment of cryptographic assets. In addition, we demonstrated the effectiveness of our system on the Ethereum

blockchain and measured the cost of gas. Our evaluation results show that the proposed system can operate with a reasonable gas cost. This investigation represents the first attempt to propose and demonstrate such a secure payment system, as far as we know.

The remainder of this paper is organized as follows: Section 2 describes some preliminaries, and Section 3 presents related works. Section 4 provides a detailed explanation of our proposed architecture. Section 5 presents the results of implementing the proposed architecture and its evaluation. Section 6 provides a discussion on the security, privacy, and signature scheme of the proposed system. Finally, we conclude the paper in Section 7.

2. Preliminaries

2.1. Blockchain. Blockchain is a distributed ledger technology invented by S. Nakamoto [3] and is the core technology of many cryptographic assets, including Bitcoin. Blockchain stores data by grouping transactions into blocks and embedding the hash value of one block into the next block that occurs on a peer-to-peer (P2P) network.

The nodes put together a transaction, generate a block, and are rewarded. This sequence of events is called mining. In particular, Bitcoin and Ethereum have employed the proof-of-work (PoW) method to determine the mining node, which is time consuming. The main chain of the blockchain is defined as the longest one from the first block to the current block. Therefore, in order to falsify the data stored in a block, all subsequent blocks would have to be revoked, and thus, a large number of computations in the PoW would be necessary. Such falsification is practically impossible.

Blockchain is tamper resistant. In addition, public blockchains are highly transparent because anyone can participate in the network and can see the data stored in the blocks.

2.2. Smart Contract and Ethereum. A smart contract is a program that is recorded on a blockchain and is falsification resistant. When the program is executed, its state is also stored in the blockchain, ensuring the transparency of the sequence of actions. In addition, there is essentially no single point of failure as the program runs on a P2P network. Because of this characteristic, contracts can be automatically executed without the need for a third party.

Ethereum [4, 5] is the first cryptographic asset to employ smart contracts and enables smart contracts to be implemented in Solidity, Viper, LLL, and other languages. Solidity is a JavaScript-like high-level programming language with Turing completeness (albeit with gas limits on execution). Ethereum uses the elliptic curve digital signature algorithm (ECDSA) [6], where the address of an account is created by taking a hash (keccak256 [7]) of the public key. This account is called an externally owned account (EOA), and the user executes a function in the smart contract through the EOA. Addresses are also assigned to smart contracts, which are generated by encoding the deployed EOA and its nonce,

called recursive length prefix encoding, and then taking a hash of them.

In order to execute a function in a smart contract, the EOA needs to issue a transaction, and a fee called gas is charged to the minor for that transaction. In general, the more complex the processing within a function is, the higher the gas value is. When a function in a smart contract references or executes a function in another smart contract, an internal transaction is issued.

2.3. Contract Wallet. A contract wallet is a smart contract for securely managing cryptographic assets in Ethereum. According to Angelo and Salzer [8], 21% of all smart contracts in the main chain of Ethereum are contract wallets.

The main functions of a contract wallet include controlling access to cryptographic assets, providing recovery from the loss of a private key, and designating whitelists. Basically, anyone with an EOA can deposit money into the account but cannot withdraw more than a particular amount unless certain conditions are met. This feature significantly reduces the risk of cryptographic asset theft, even if the private key of the wallet owner is stolen. The wallet also allows deposits to be made to whitelisted accounts without the requirement to do so, which increases the convenience of the wallet. However, some conditions must be met when designating an arbitrary account for the whitelist. In many cases, multisig is introduced as a condition for this purpose.

Typically, the EOAs of the wallet owner and organization that create and manage the wallet are the targets of the multisig users required for signatures, but some types of contract wallets allow individuals to designate their own trusted third parties. If it is suspected that the private key of the EOA that owns the contract wallet has been stolen, the private key can be updated by locking the wallet and contacting the owner of the trusted EOA designated to the multisig partner. By having the owner of the contract wallet designate a newly created EOA, the EOA whose private key is stolen loses access to the assets, and the wallet owner can protect the assets.

3. Related Work

Angelo and Salzer [8] analyzed the characteristics of contract wallets deployed in the Ethereum main chain and grouped them into six types. To this end, they presented approaches to identify contract wallets by analyzing the source code, bytecode, and execution traces extracted from transaction data. Moreover, they investigated usage scenarios and patterns. In addition, the following researchers have utilized smart contracts purposefully and have proposed various applications.

Wang et al. [9] attempted to solve the integrity problem, which is a key issue in Cloud storage auditing, by using blockchain technology and smart contracts. Traditional auditing schemes necessitate a third-party auditing and imply a limited pay-as-you-go service as they require clients to pay for services in advance. However, Wang et al. replaced third-party auditing with blockchain technology and

achieved fair payments by performing the payments through smart contracts.

Furthermore, Tam et al. [10] proposed a pay-as-you-go automotive insurance application based on smart contracts. The data are stored in a blockchain, making them tamper-proof and traceable, and premiums can be charged explicitly.

Subsequently, Yanqi et al. [11] proposed a secure brokerless pub/sub model using smart contracts and reputation systems [12]. This model combines these two elements and hybrid cipher based on Elgamal [13] and AES to satisfy the security requirements of confidentiality, authentication, scalability, integrity, fairness, and anonymity.

Finally, Thitinan and Nandar [14] proposed an emergency notification service by combining IoT devices and smart contracts. Using environmental data observed by IoT devices as triggers, this system enables reporting to public services such as hospitals and payment of bills through smart contracts.

However, these researchers did not address the proper course of action when the private key of a user EOA is stolen. Hence, this paper proposes a pay-as-you-go service system that takes into account the risks involved.

4. Proposed Architecture

4.1. Overview. We propose an autonomous pay-as-you-go payment system with secure management of assets using contract wallets. Implementing a multisig-compatible contract wallet for payments ensures that even if the private key of a user is stolen, his or her assets will not be compromised. The assumption is that the relevant company intends to develop a pay-as-you-go business while providing services to users using IoT devices. Figure 1 provides an overview of the proposed architecture. The architecture consists of the following seven components:

- (1) Company X deploys a smart contract (i.e., a control contract) for calculating fees and controlling the status of the IoT
- (2) User A requests to register an IoT device, the control contract, and the EOA of company X in his or her contract wallet
- (3) The contract wallet organization registers the IoT device, control contract, and EOA of company X in the contract wallet of the user
- (4) The IoT device refers to its own state regularly from the control contract
- (5) The IoT device sends usage data to the contract wallet of the user
- (6) The control contract calculates the fee from the usage data and returns the value to the contract wallet of the user
- (7) The contract wallet of the user transfers the calculated fee to the EOA of company X

The contract wallet organization is a part of multisig, but it does not have strong authority by itself (e.g., the right to access user assets).

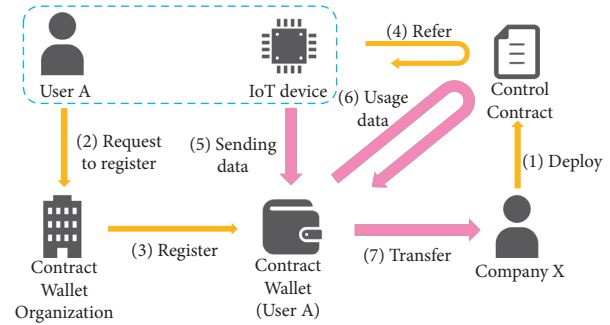


FIGURE 1: Entire flow of the proposed system.

4.2. Use Cases and Comparison with the Conventional Architecture. Use cases include automatic payments for lifelines such as water and electricity. Lifeline and pay-as-you-go tariffs are compatible as they are essential to life and are expected to be used continuously.

As Figure 2 shows, in pay-as-you-go automated payments using smart contracts, it has traditionally been necessary to deposit assets into a smart contract that exists for each IoT device. Consequently, as IoT devices are added to the system, the burden on the user increases as the amount of time to deposit increases. In the proposed architecture, a control contract for each IoT device is registered in its own contract wallet and is paid from the contract wallet, which greatly improves the convenience because there is no need to make a deposit for each IoT device.

4.3. System Model

4.3.1. User. The user has an EOA and can access his or her contract wallet by issuing a transaction. The addresses of the EOA and control contracts of the IoT devices distributed by the company are assumed to be known by the company. In addition, the user checks each time a transaction is issued to see if it is intended.

4.3.2. Company. The company is an organization that distributes and manages IoT devices and control contracts to users and has access to the control contracts. The company has an EOA or wallet address that the user designates as the destination for the assets and notifies the user in advance.

4.3.3. Contract Wallet Organization. This organization manages user-owned contract wallets and is a part of the multisig of the wallet. It is in charge of reviewing control contracts and updating contract wallets. The assets remain in the hands of the users and are not held by the contract wallet organization, so they are not centralized. Therefore, the organization does not have strong authority because it cannot access user assets by itself.

4.3.4. IoT Device. The IoT device periodically issues a transaction to send the translation of the provided service into usage data to the contract wallet. It also periodically refers to the state of the control contract to determine

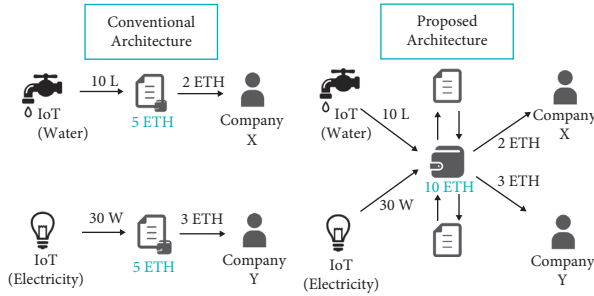


FIGURE 2: Comparison with the conventional architecture.

whether to turn the service on or off. In this implementation, we assume that the secret key of the EOA is hard coded into the IoT device. In addition, it is possible to issue transactions directly from IoT devices using the Web3API of Infura.

4.3.5. Contract Wallet. To store user assets securely, the user can access assets via multisig with the contract wallet organization. In this study, we adopted 2 of 2 multisig. In addition, only when data are sent from the registered IoT device is it possible to move the assets to the EOA of the whitelisted company without the need for a multisig.

4.3.6. Control Contract. The control contract calculates the fee based on the data sent from the contract wallet and returns the value to the contract wallet. It also has a function to turn the state parameters of the IoT device on and off. The source code must be made available to the public by the company.

4.4. System Flow. Figure 3 depicts the flow of automatic asset transfer from IoT devices. Note that transactions are denoted as Tx and internal transactions as InTx.

4.4.1. Whitelist Registration

- (1) The user sends a pre-given EOA of the company to which the fees are to be sent and his or her signature with the corresponding private key to the contract wallet organization. This work is done outside the blockchain.
- (2) The multisig counterpart of the user, the contract wallet organization, scrutinizes its EOA for trustworthiness.
- (3) If reliable, the organization executes the function *Whitelisting* in the contract wallet and adds the EOA to the whitelist of the contract wallet under multisig.

4.4.2. Control Contract Registration

- (1) The user sends the EOA of the IoT device used, the address of the associated control contract, and the EOA of the company to which the payment is made, along with his or her signature, to the contract wallet

organization. This work is done outside the blockchain.

- (2) The contract wallet organization similarly scrutinizes the control contracts published by the company.
- (3) Then, if the control contract is trusted, it executes the function *RegisterIoT* in the contract wallet and registers the EOA of the IoT device with the EOA of the company, as sent by the user under multisig.

4.4.3. Transmission of Data from IoT Devices and Automatic Payment

- (1) The IoT device periodically converts the services provided by the device into usage data, executes the contract wallet function *IoTPayment*, and transfers the data to the contract wallet
- (2) The function *IoTPayment* executes the function *Calculation* in the control contract and obtains the fee value
- (3) If the deposit has a value greater than the fee value obtained from the control contract, the cryptographic asset is sent to the designated EOA based on the fee value
- (4) If the deposit has a value less than the fee value obtained from the control contract, the function *Operation* in the control contract is executed, and the state parameter of the IoT device is turned off

4.4.4. Stopping the IoT Device

- (1) If a user notices something unusual activity, such as an unintended transaction being sent, the user contacts the company and asks the company to stop the IoT device
- (2) When contacted, the company executes the function *Operation* in the control contract and sets the state parameters of the IoT device to off

4.5. Smart Contract. The proposed architecture assumes that the contract wallet has already been deployed on the blockchain, and the user registers the EOA and control contracts of the IoT devices to the contract wallet. The user signature is assumed to be a hash (keccak256) of his or her EOA signed by ECDSA with his or her own private key, as well as a hash (keccak256) of the combined addresses of the user EOA, EOA of the IoT device, and control contract.

The functions of control contracts and contract wallets are as follows (Algorithms 1–3 are contract wallets, and Algorithms 4 and 5 are functions in control contracts).

4.5.1. Whitelisting. This function whitelists the EOA of the company to which the charges are to be paid and is executed by the contract wallet organization at the request of the user. Specifying an EOA of the company allows for payment without any asset withdrawal limits. By verifying the

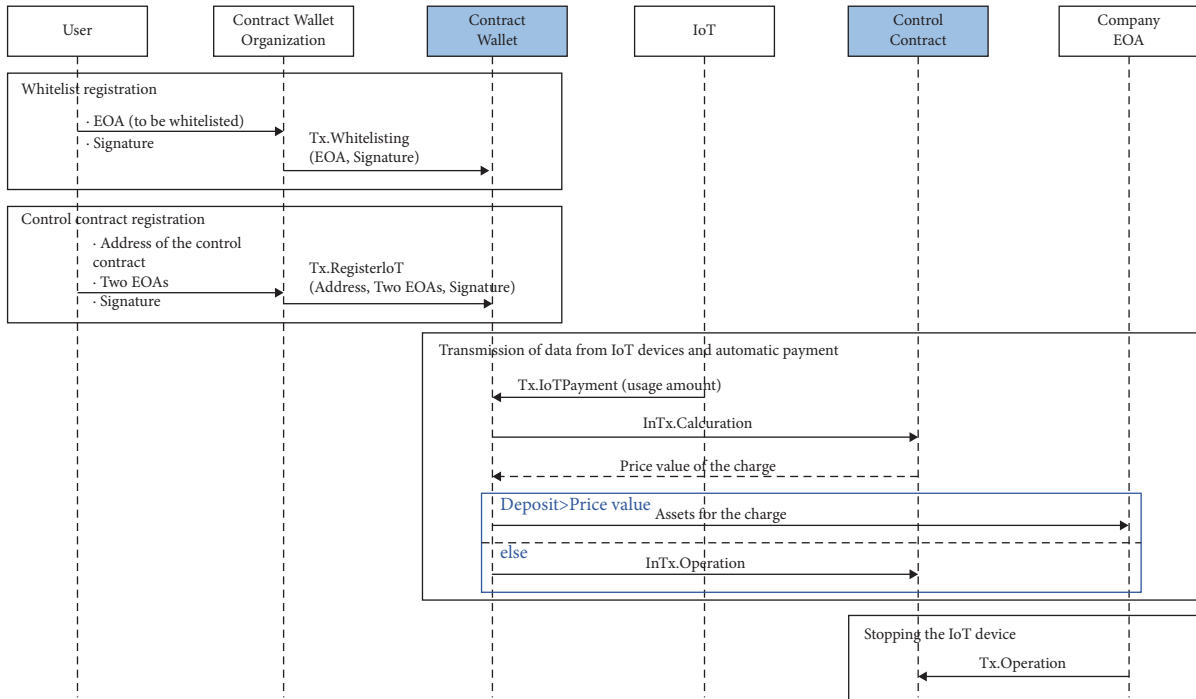


FIGURE 3: System flow.

Input: user EOA, company EOA, and user signature
Output: none
 (1) Generate a hash (keccak256) of the company EOA
 (2) Decrypt the public key and EOA of the signer from the hash and user signature using ECDSA
 (3) **if** decrypted EOA == user EOA **then**
 (4) Add company EOA to the whitelist
 (5) Emit the event
 (6) **else**
 (7) Error handling
 (8) **end if**

ALGORITHM 1: Whitelisting.

Input: user EOA, IoT EOA, address of the control contract, company EOA, and user signature
Output: none
 (1) Generate a hash (keccak256) of the user EOA, company EOA, and control contract address, concatenated in that order
 (2) Decrypt the public key and EOA of the signer from the hash and user signature using ECDSA
 (3) **if** decrypted EOA == user EOA **then**
 (4) Tie the IoT EOA to the control contracts and company EOA and insert it into the IoT registration list
 (5) Emit the event
 (6) **else**
 (7) Error handling
 (8) **end if**

ALGORITHM 2: RegisterIoT.

```

Input: usage amount
Output: none
(1) if a transaction issuer in the IoT registration list then
(2) Run the function Calculation in the control contract tied to the issuer EOA
(3) if determine if the value obtained by function Calculation is greater than the value of deposit then
(4) Transfer assets for the value obtained from the calculation to the company EOA, which is linked to the issuer EOA
(5) Emit the event
(6) else
(7) Run the function Operation in the control contract
(8) end if
(9) Error handling
(10) end if

```

ALGORITHM 3: IoTPayment.

```

Input: usage amount
Output: charge value
(1) Calculate the charge value from usage amount
(2) return value

```

ALGORITHM 4: Calculation.

```

Input: none
Output: none
(1) if issuer EOA == the company EOA then
(2) Convert the IoT on/off state
(3) else
(4) Error handling
(5) end if

```

ALGORITHM 5: Operation.

TABLE 1: Execution environment.

Name	Version
Arduino-IDE	1.8.13
Solidity	0.5.17

signatures sent outside of the blockchain by the user within this function, together with the verification of the transactions themselves, we achieve 2 of 2 multisig.

4.5.2. RegisterIoT. This function associates the EOA of an IoT device with the EOA of the company and address of the control contract and registers them to the IoT registration list. It is executed by the wallet organization at the request of the user. By registering to the IoT registration list, the IoT device can execute the function IoTPayment. This function is executed in multisig in a similar way to the function Whitelisting.

4.5.3. IoTPayment. This function executes asset payment based on the usage data given by the IoT device. It checks whether the EOA of the IoT device is registered in the contract wallet. If so, the function obtains the charge value

by executing the function Calculation in the control contract and then transfers the cryptographic assets of the value to the EOA of the company to which it is linked. If the deposit is less than a given charge value, the function Operation in the control contract, which switches the service provided by the IoT device on and off, is executed.

4.5.4. Calculation. This function is executed internally by the function IoTPayment. It computes the fees from the given usage data and returns the value to the contract wallet.

4.5.5. Operation. This function manages the status of the IoT devices and changes the variables that control the turning on and off of the IoT device services. It cannot be executed by anyone other than the company unless the private key of the company is obtained because it checks if the EOA of the transaction issuer is the EOA of the company or contract wallet.

5. Implementation and Evaluation

We demonstrated the feasibility of the proposed architecture by implementing a prototype of the proposed method in

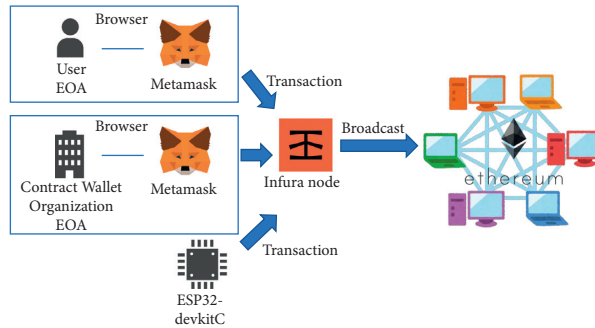


FIGURE 4: System components.

Transactions

Latest 20 from a total of 20 transactions

Txn Hash	Block	Age	From	To	Value	[Txn Fee]
0x62e7a2cee31da0...	6934770	1 min ago	0x5cad396b242e73...	OUT → 0x85f0aa756988e67...	0 Ether	0.000036815
0x8f889086d3539dc...	6934749	6 mins ago	0x5cad396b242e73...	OUT → 0x85f0aa756988e67...	0 Ether	0.000036741
0x4ecf390f4ee8ecc...	6934728	11 mins ago	0x5cad396b242e73...	OUT → 0x85f0aa756988e67...	0 Ether	0.000036741
0x97b1c78875c4ae...	6934707	16 mins ago	0x5cad396b242e73...	OUT → 0x85f0aa756988e67...	0 Ether	0.000036741

FIGURE 5: Transaction from the IoT device.

Transactions Internal Txns

Latest 9 internal transactions

Parent Txn Hash	Block	Age	From	To	Value
0x8f889086d3539dc...	6934749	6 mins ago	→ 0x85f0aa756988e67...	0xa4820181be90b5...	0.00000000000008355 Ether
0x4ecf390f4ee8ecc...	6934728	11 mins ago	→ 0x85f0aa756988e67...	0xa4820181be90b5...	0.00000000000011635 Ether
0x97b1c78875c4ae...	6934707	16 mins ago	→ 0x85f0aa756988e67...	0xa4820181be90b5...	0.0000000000001197 Ether

FIGURE 6: Internal transaction from the contract wallet.

Latest 20 internal transactions

Parent Txn Hash	Block	Age	From	To	Value
0x62e7a2cee31da0...	6934770	1 hr 27 mins ago	→ 0x85f0aa756988e67...	→ 0x41c0da607bfb6df...	0 Ether
0x62e7a2cee31da0...	6934770	1 hr 27 mins ago	→ 0x85f0aa756988e67...	→ 0x41c0da607bfb6df...	0 Ether
0x8f889086d3539dc...	6934749	1 hr 32 mins ago	→ 0x85f0aa756988e67...	→ 0xa4820181be90b5...	0.00000000000008355 Ether
0x8f889086d3539dc...	6934749	1 hr 32 mins ago	→ 0x85f0aa756988e67...	→ 0x41c0da607bfb6df...	0 Ether

FIGURE 7: Company EOA receives assets.

TABLE 2: Gas cost.

Transaction	Gas cost	Transaction fee
Whitelisting	29,247 gas	734977.11 Gwei
RegisterIoT	84,047 gas	2112101.11 Gwei
IoTPayment (normal)	36,741 gas	923301.33 Gwei
IoTPayment (insufficient funds)	36,615 gas	920134.95 Gwei

TABLE 3: Address.

Characteristic	Address
Company EOA	0xa4820181BE90b5570Ce03bbF605b65796d6B6bBA
IoT device EOA	0x5cAd396b242E73BC9f4827F33aa45fdc73061801
Contract wallet address	0x85f0aA756988E67fE64baaB613C3aA896AA5Ac84
Control contract address	0x41c0Da607bfB6DF20dB79D49d27F6D94c3cdedE9

Rinkeby, an Ethereum test network, using MetaMask and Infura. The device and smart contracts were written in Arduino language and Solidity language, respectively.

5.1. System Components

5.1.1. ESP32-DevKitC (1 Machine). This component plays the role of an IoT device. In this implementation, instead of actually measuring the services provided by the IoT devices and transferring the usage data, random integers are utilized as usage data. The development environment for IoT devices is the Arduino-IDE. The CPU is an Xtensa dual-core 32 bit LX6 microprocessor. The memory is 520 KiB (SRAM). The capacity is 4 MiB (flash memory).

5.1.2. Infura. Infura is the node-hosting service of Ethereum. Users can join the Ethereum blockchain through this service without the need to set up a full Ethereum node. Because the JSON-RPC API of Web3 is available, transactions can be issued from IoT devices via http or WebSocket.

5.1.3. MetaMask. This component is a Google Chrome extension, and the service facilitates EOA management on the browser. It is possible to specify the node to be used, and an Infura node is utilized in this implementation.

Table 1 describes the execution environment, and Figures 4 and 5 show the overall configuration diagram.

Figure 6 indicates that an internal transaction is issued from the contract wallet. Here, the contract wallet executes the function Calculation in the control contract and receives the value.

Figure 7 Demonstrates that the assets are transferred from the contract wallet to the company EOA.

These observations confirm that our proposed architecture involving automatic assets' transfer works properly.

Table 2 lists the gas cost incurred for each executed function, and Table 3 provides the addresses of the IoT, company, and contract wallet.

There are 51 and 25 lines of code for the contract wallet and control contract, respectively. The lines of code were measured using the cloc command on macOS Catalina

10.15.7. The gas price was set to 25.13 Gwei, which is the average in the mainnet on July 1, 2021.

5.2. Experiment. ESP32 executed the contract wallet function IoTPayment every 5 min through Infura using random values between 1000 and 2000. To confirm the transactions, we utilized a website called Etherscan (Etherscan, "Etherscan," 2020, <https://etherscan.io/>), which enabled us to check transactions in Ethereum. Figure 7 presents the transaction records for each of the addresses identified on this site.

Figure 5 shows that the transaction is executed in roughly 5 min.

6. Discussion

6.1. Security Analysis. The possible attacks on our proposed system can be classified into the following four patterns. We discuss them in terms of "purpose of attack," "assumed attack means," and "countermeasures."

6.1.1. Attack from a Third Party to the Private Key of the IoT. The purpose of this attack is for a third party to impersonate the IoT device and force the user to pay an unusual fee.

As a means of attack, by stealing the private key in the IoT in some way, an attacker can issue a transaction with the free value as the amount used by the user.

The countermeasure would be that the contract wallet organization would monitor the blockchain and notify users whenever a transaction related to their EOA is issued. This notification system enables users to provide notice immediately when an unintended transaction is issued. The user can then contact the company to stop the service and request the return of the moved assets. In addition, because the destination of the assets is specified in the contract wallet as the EOA of the company, merely stealing the private keys of the IoT device will not allow the assets to be moved to the wallet controlled by the attacker.

6.1.2. Attack from a Third Party to the Private Key of the Users. The purpose of this attack is for a third party to impersonate the owner of the contract wallet and attempt to access the contract wallet.

As a means of attack, by stealing the private key of the user in some way, an attacker can attempt to issue a transaction that transfers the assets in the contract wallet to the attacker's account.

Against this attack, the contract wallet can minimize the damage by having most of the user's assets moved to the contract wallet. This is because contract wallets are designed so that even the owner of the contract wallet can restrict the withdrawal limit. Therefore, if the owner moves many assets from his EOA to his contract wallet, the attacker cannot move all assets pooled in the contract wallet even if he has the owner's private key. To remove the restriction, a digital signature with the owner's private key and the private key of the contract wallet management organization must be verified in the contract wallet. In addition, when an asset withdrawal occurs, the transaction is issued, and a notification is sent to the user from the contract wallet organization so that he or she is made aware of the unintended transaction. Moreover, by designating a newly created EOA as the owner of the contracted wallet under the multisig of the previous EOA of the user and the EOA of the contract wallet organization, the attacker will lose access to the wallet.

6.1.3. Attack from the User Side. The purpose of this attack is for the user to receive services but not to pay for them.

It is assumed that the user will pay for the assets used in the period at a fixed time. As a means of attack, by reducing the amount of assets in the contract wallet in advance, the user receives more services than he or she can pay for with his or her assets by the time the next payment is made (e.g., even if the assets in the contract wallet are set to zero, the user will still receive the service until the next payment).

The attack can be handled by preparing a security deposit, which we did not implement in this study. If there are not enough assets in the contract wallet to pay at the time of payment, the benefit to the user side can be reduced by collecting the deposit.

6.1.4. Attack from the Company Side. The purpose of this attack is for the company to receive fees but not to provide the service.

This type of attack is virtually impossible because the program is written in the smart contracts in such a way that users are charged and paid for the amount they use.

6.2. Privacy Issue. The proposed method has an issue that the data are exposed to the public chain without encryption. However, this is not severe because the EOA of the user is generally not linked to the user's personal information (e.g., users of cryptographic assets accept that the remittance history is public).

To solve this problem, instead of using a public blockchain, a consortium or private blockchain can be employed. Either of these blockchains can be used to partially restrict the disclosure of information; however, the possibility that the blockchain nodes may act fraudulently cannot be eliminated. Therefore, it is necessary to consider this aspect.

6.3. Private Key Distribution Using the Threshold Signature. The proposed architecture uses a multisig contract wallet to distribute the access rights to the assets to achieve a secure payment method, but there is also a method to distribute the private key itself [15, 16] by using secret sharing [17]. This method is called the threshold signature scheme (TSS) (Binance, "Threshold Signatures Explained," 2020). By applying this signature scheme to a wallet of cryptographic assets, it is possible to create a decentralized wallet.

The main difference between contract wallets and TSS-based wallets is that the former are implemented on smart contracts, whereas the latter are implemented as ordinary software outside the blockchain. The secret share of a private key is normally distributed among various devices. By collecting a threshold number of secret shares synchronously with its devices, TSS-based wallets can decrypt the private key and issue transactions. Therefore, we can distribute the privileges by allowing others to manage the secret share, similarly to multisig.

However, TSS is a relatively new technology, and its communication protocol is more complex than that of other public key cryptographic methods. We are in the process of developing new signature methods, including previously undiscovered attacks. Therefore, TSS is a promising technology, although there is scope for further study.

7. Conclusion

In this study, we implemented and evaluated a pay-as-you-go automatic payment system using contract wallets to reduce the risk of cryptographic asset theft due to private key theft. We implemented a contract wallet and control contract using the Solidity language and connected them by utilizing internal transactions. We have also implemented ESP32 using the Arduino language to demonstrate that the proposed method is feasible as it allows the IoT devices to issue transactions and reference states from the blockchain. This ESP32 only keeps the private key to issue transactions. In addition, we recorded the transactions that occurred and measured the gas cost. In the proposed architecture, the control contracts can be freely programmed to be more complex, but it is necessary to account for the increased gas cost.

In future work, we plan to address privacy issues such as encryption and to consider ways to reduce the gas cost by using the TSS and other methods. We will also consider more deeply about the key management of IoT devices.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by Grant-in-Aid for Scientific Research (B) (19H04107).

References

- [1] Statista, “Number of internet of things (iot) connected devices worldwide in 2018, 2025 and 2030,” [Online]. Available: <https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/>, 2019.
- [2] Fortune Media Group Holdings, “How to steal \$500 million in cryptocurrency,” [Online]. Available: <https://fortune.com/2018/01/31/coincheck-hack-how/>, 2018.
- [3] S. Nakamoto, “Bitcoin: a peer-to-peer electronic cash system,” [Online]. Available: <http://www.bitcoin.org/bitcoin.pdf>, 2009.
- [4] Ethereum, “A next-generation smart contract and decentralized application platform,” [Online]. Available: <https://ethereum.org/en/whitepaper/>, 2021.
- [5] G. Wood, “Ethereum: a secure decentralised generalised transaction ledger petersburg version fa00ff1 – 2021-05-12,” [Online]. Available: <https://ethereum.github.io/yellowpaper/paper.pdf>, 2021.
- [6] D. Johnson, A. Menezes, and S. Vanstone, “The elliptic curve digital signature algorithm (ecdsa),” *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.
- [7] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, “Keccak, advances in cryptology - eurocrypt 2013,” in *Proceeding of the Annual international conference on the theory and applications of cryptographic techniques*, pp. 313-314, Springer, Athens, Greece, May 2013.
- [8] M. D. Angelo and G. Salzer, “Characteristics of wallet contracts on ethereum,” in *Proceeding of the 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, Sepember 2020.
- [9] H. Wang, H. Qin, and M. Zhao, “Blockchain- based fair payment smart contract for public cloud storage auditing,” *Information Sciences*, vol. 519, pp. 348–362, 2020, [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025520300621>.
- [10] V. H. Tam, M. Lenin, M. Mukesh, and A. Ermyas, “Blockchain-based data management and analytics for micro-insurance applications,” in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 2539–2542, Beijing, China, November 2017.
- [11] Z. Yanqi, L. Yannan, M. Qilin, Y. Bo, and Y. Yong, “Secure pub-sub: blockchain-based fair payment with reputation for reliable cyber physical systems,” *IEEE Access*, vol. 6, pp. 12295–12303, 2018.
- [12] K. Bok, J. Yun, Y. Kim, J. Lim, and J. Yoo, “User reputation computation method based on implicit ratings on social media,” *TIIS*, vol. 11, no. 3, pp. 1570–1594, 2017.
- [13] T. Elgamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.
- [14] T. Thitinan and A. Y. Nandar, “Emergency service for smart home system using ethereum blockchain: system and architecture,” in *Proceeding of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 888–893, IEEE, Kyoto, Japan, March 2019.
- [15] D. Jack, K. Yashvanth, L. Eysa, and S. Abhi, “Threshold ecdsa from ecdsa assumptions: the multiparty case,” in *Proceeding of the IEEE Symposium on Security and Privacy (SP)*, pp. 1051–1066, IEEE, San Francisco, CA, USA, May 2019.
- [16] G. Rosario and G. Steven, “Fast multiparty threshold ecdsa with fast trustless setup,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1179–1194, New York, NY, USA, October 2018.
- [17] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

Research Article

Multi-Authority Criteria-Based Encryption Scheme for IoT

Jianguo Sun , Yang Yang , Zechao Liu , and Yuqing Qiao 

College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China

Correspondence should be addressed to Zechao Liu; liuzechao@hrbeu.edu.cn

Received 6 April 2021; Accepted 7 July 2021; Published 17 July 2021

Academic Editor: Qi Jiang

Copyright © 2021 Jianguo Sun et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Currently, the Internet of Things (IoT) provides individuals with real-time data processing and efficient data transmission services, relying on extensive edge infrastructures. However, those infrastructures may disclose sensitive information of consumers without authorization, which makes data access control to be widely researched. Ciphertext-policy attribute-based encryption (CP-ABE) is regarded as an effective cryptography tool for providing users with a fine-grained access policy. In prior ABE schemes, the attribute universe is only managed by a single trusted central authority (CA), which leads to a reduction in security and efficiency. In addition, all attributes are considered equally important in the access policy. Consequently, the access policy cannot be expressed flexibly. In this paper, we propose two schemes with a new form of encryption named multi-authority criteria-based encryption (CE) scheme. In this context, the schemes express each criterion as a polynomial and have a weight on it. Unlike ABE schemes, the decryption will succeed if and only if a user satisfies the access policy and the weight exceeds the threshold. The proposed schemes are proved to be secure under the decisional bilinear Diffie–Hellman exponent assumption (q-BDHE) in the standard model. Finally, we provide an implementation of our works, and the simulation results indicate that our schemes are highly efficient.

1. Introduction

As an emerging concept, the Internet of Things (IoT) offers great convenience to our daily lives since it provides individuals with ultra-fast data transmission and quality storing services by edge infrastructure. Many well-known IT enterprises such as Google, Microsoft, and Amazon have deployed edge computing platforms to integrate edge infrastructure and various devices, so that individuals can benefit in many fields [1]. Unfortunately, due to the complexity of architecture, there are inevitably some security risks in IoT, especially that some unsupervised edge infrastructures may quietly capture users' sensitive information or be compromised by malicious users, which poses a severe threat to individuals [2, 3]. For example, edge devices may reveal sensitive data such as health records and personal finances to the public. Therefore, data security in IoT has become a significant concern for many enterprises or individuals.

To alleviate this situation, Yeh et al. [4] proposed an access control framework for IoT with the property of

attribute revocation. Qiu et al. [5] constructed an authentication and key agreement (AKA) protocol for lightweight devices in IoT. The protocol was proved to be secure in the random oracle model and enjoyed desirable computing efficiency. Wang et al. [6] conducted a detailed analysis of the vulnerability for IoT devices and offered targeted countermeasures depending on the types of attacks. However, traditional public-key techniques only support one-to-one encryption, i.e., messages encrypted by public keys can only be decrypted by their corresponding private keys. This means that there needs to be sufficient storage space to store the ciphertext in practical applications, whereas edge devices generally have limited storage capacity.

Attribute-based encryption (ABE) is an effective encryption tool that provides fine-grained and one-to-many access control for outsourcing data in IoT [7]. According to different encryption mechanisms, ABE can be divided into ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE). In CP-ABE, the data owner can construct an access policy and embed it into the ciphertext, and the user's attribute set is embedded in the secret key. On the contrary,

the private keys in KP-ABE are associated with the access policy, and the ciphertext is labeled with attributes. A user can successfully recover messages if and only if his/her attributes satisfy the access policy. Many excellent ABE schemes for access control in IoT have been proposed [8–11]. However, most of them have two problems. On the one hand, only a single attribute authority (AA) manages the whole attribute set and generates the secret keys. If a large number of users request private keys, the server will be at risk of crashing. Furthermore, once the attribute authority is compromised, any user with unauthorized attributes will be able to decrypt the ciphertext. Therefore, ABE schemes supporting multiple authorities should be considered, i.e., the attribute universe should be managed by multiple attribute authorities. In this way, even if an authority compromises or collapses, a user can still obtain the secret key from other authorities. On the other hand, all attributes in the access policy of the previous schemes are regarded at the same level, which ignores the scenario that some attributes may be more important than the others. More precisely, in an IoT-based medical system, it is desirable to grant doctors higher weights than the nurses.

In order to distinguish the importance among attributes, some weighted ABE schemes [12–14] have been proposed. Liu et al. [12] proposed a weighted CP-ABE scheme. However, in the scheme, the attribute universe is managed by a single central authority. Wang et al. [13] constructed a multi-authority weighted ABE scheme in cloud computing. In the scheme, CA is still required in the key generation phase, which reduces the security of the scheme. Yan et al. [14] introduced a weighted attribute-based encryption scheme. However, the weight corresponding to each attribute is specified by a central authority, while in the actual scenario of encryption, the data owner should be allowed to decide the weight of each attribute in the access policy. To address the above problems, Phuong et al. [15] first proposed criteria-based encryption (CE) scheme, which supports the weighting of each criterion in the access policy. To be precise, each criterion is expressed as a polynomial, each root of which corresponds to a case satisfying the polynomial-associated criterion. The access policy consists of a series of weighted criteria containing at least one case. For this, the main difference between ABE and CE is that each criterion contains multiple satisfying cases and has a reasonable weight specified by the encryptor. An instance of intuition is provided as follows. Suppose that in a smart medical system, the government needs to monitor the health of community members. Since medical data involve sensitive information of individuals and are not available to others, the receivers need to meet certain restrictions to make access possible ((the receiver must be an authorized chief physician, weighted 5, and marked as a criterion P_1) AND (the receiver has more than 5 years of work experience, weighted 2, and marked as a criterion P_2) OR (the receiver is a community manager employed by the government, weighted 1, and marked as a criterion P_3) OR (the receiver is a community member holding a legal device, weighted 6, and marked as a criterion P_4)). And in order to access the data, the cumulative weight of the receiver must be more than 5. Bob is a

community manager hired by the government and has 6 years of work experience related to medical treatment. He cannot obtain approval for not reaching the cumulative weight threshold as required. Alice is a chief physician who has served the community for seven years. She satisfies both the access policy and the threshold, so she can be authorized. As shown in Figure 1, the criterion P_3 corresponds to two cases (roots): the receiver is a community manager and appointed by the government. But unfortunately, the issue of generating keys by only a single authority is still unsolved in their scheme.

In this paper, we propose two types of multi-authority criteria-based encryption schemes, named MA-CE-Verify Root and MA-CE-Root Equality, respectively, which aim to solve the problems we mentioned above. Specifically, we denote each criterion as a polynomial. One can assign a weight for each criterion freely according to demands. In addition, the corresponding cases of satisfying the criteria are represented as the roots of polynomials. In the first scheme, at least a case (or root) of each criterion specified in the access policy should be held by the decryptor, and the cumulative weight needs to exceed the threshold as well for successful decryption, while in the second scheme, only if the decryptor satisfies all the cases (or all roots) for each criterion and the cumulative weight exceeds the threshold, he/she can decrypt correctly. Moreover, in our schemes, multiple authorities manage the global criterion universe and perform key generation, which solves the bottleneck of performance and improves the security of the system.

1.1. Our Contributions. In this work, our main contributions can be summarized as follows:

- (1) We propose two types of multi-authority criteria-based encryption schemes, which support the weighting of each criterion. In our schemes, multiple AAs jointly manage the criterion universe using the (t, n) -threshold sharing technology. Furthermore, data owners can freely set the weight of each criterion as required. Thus, flexible access control is provided by our schemes.
- (2) The security proof shows that our schemes achieve indistinguishability under chosen-plaintext attack (IND-CPA) under the decisional bilinear Diffie-Hellman exponent assumption (q-BDHE).
- (3) We implement the proposed schemes and provide theoretical analysis. The results show that our constructions have desirable performance in practical situations.

1.2. Related Work. Goyal et al. [16] proposed attribute-based encryption (ABE) that provides one-to-many encryption. In their works, ABE is divided into two forms: ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE). Sahai et al. [17] realized a revocable ABE (RABE) scheme, in which the outsourcing server updates the encrypted data to revoke the user's decryption permission. On the downside, the complexity of bilinear-pairing operations makes it difficult to

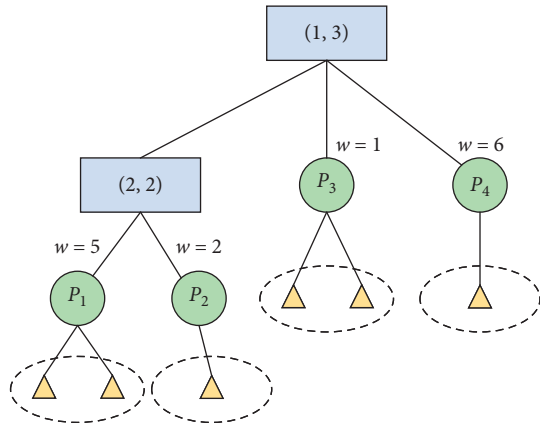


FIGURE 1: The example of access policy.

directly apply this scheme to IoT. Agrawal et al. [18] proposed two versatile ABE architectures with short ciphertext and key. One limitation is that the scheme does not consider that different attributes in the access policy are at different levels of importance, i.e., the attributes do not carry reasonable weights. Waters [19] and Agrawal et al. [20] proposed ABE schemes that support arbitrary length input and provide a general ABE structure. In these schemes, the management of attribute universe and key generation are only implemented by a single attribute authority. Once the authority is corrupted, the adversary can directly generate the key of any user with legal status to decrypt the message [21]. ABE schemes with multiple authorities have been proposed to solve this issue. Lewko et al. [22] constructed an ABE scheme in which any party can become an attribute authority. Moreover, the scheme can resist collision attacks. However, the construction based on composite order group seriously affects the execution efficiency of the scheme. In [23–26], the schemes are provided for different practical application scenarios. Unfortunately, these schemes are limited by some security issues or computational complexity. In this context, there are obstacles to directly applying them in IoT scenarios. Sandor et al. [27] presented an efficient decentralized multi-authority ABE scheme that can significantly solve the key escrow problem for mobile devices. Generally, decentralizing ABE solves the problem of accessing encrypted data when the attributes of users come from multiple authorities, in which each authority is only in charge of issuing attributes and keys in its domain. However, in the schemes, an adversary can still compromise the server of AA to obtain some information that he should not have. The issue can be solved by using (t, n) -threshold sharing in our works. The adversary cannot get any information related to the key unless the number of corrupted authorities is greater than t .

1.3. Organization. In Section 2, we present the notation and preliminaries. In Section 3, we provide three components. The system model and some requirements of the schemes are described in Section 3.1. We define the framework of the schemes in Section 3.2, while the security model is given in

Section 3.3. In Section 4, we illustrate how to construct our two schemes. We give the security proof of our schemes in Section 5. The performance analysis of proposed schemes is represented in Section 6. At the end of our work, the conclusions and extensions are put forward in Section 7.

2. Preliminaries

We now introduce some notations and preliminaries.

2.1. Notation. For a positive integer n , $[1, n] = \{1, 2, \dots, n\}$. For vector \vec{u} and \vec{v} , let $\langle \vec{u}, \vec{v} \rangle$ be the inner product of two vectors. We use $a \in_R S$ to denote a random element a drawn from set S uniformly. For a matrix M , its i -th row is denoted by M_i , and its (i, j) -element is $M_{i,j}$. We use the symbol $C \models \mathbb{A}$ to denote the criterion set C satisfies the access structure \mathbb{A} . Note that the (monotonic) access structure used in this work is similar to that in literature [8], so the concrete concept is not repeated here. For any set S , $\text{len}(S)$ denotes the number of its elements.

2.2. Bilinear Maps. Let \mathbb{G} and \mathbb{G}_T be two multiplicative cyclic groups of order p , where p is a large prime number and \mathbb{G} is generated by g . Let $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be an admissible bilinear map, if it satisfies the following properties:

- (1) Bilinearity: for any $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(g^a, h^b) = e(g, h)^{ab}$.
- (2) Nondegeneracy: for any $g \in \mathbb{G}$, $e(g, g) \neq 1$.
- (3) Computability: for any $g, h \in \mathbb{G}$, there is an efficient algorithm to calculate $e(g, h)$.

2.3. (t, n) -Threshold Secret Sharing. Suppose that several participants intend to share a secret with each other, while they do not hope that any one of them can obtain the secret independently, due to the privacy requirement of the secret. Secret sharing is a technique proposed to be used in the scenario above. In the secret-sharing scheme, each party can obtain a share of the secret, which is actually a part of information about the secret, and the whole secret can be reconstructed only by the cooperation of participants, which means that any party cannot know what the secret is individually. There have been many various secret-sharing schemes suitable for different situations proposed, and the (t, n) -threshold sharing is one of the most widely applicable and basic schemes among them. It was first proposed by Shamir [28] and then improved into many practical schemes, such as [21, 29]. In this work, we adopt the definition in [21].

We take the set $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_n\}$ as n members of the system. The identity of each member x_i ($i \in [1, n]$) is taken from the finite field $\text{GF}(p)$. Let the positive integer t ($t \leq n$) denote a threshold. Additionally, let S_i represent the subsecret of each member, such that $S = \sum_{i=1}^n S_i$. The (t, n) -threshold secret sharing can be described as follows.

Share. Each member constructs the polynomial $q_i(x)$ of degree $t - 1$, such that $S_i = q_i(0)$. For $j = 1$ to n , each

member calculates subshare $\eta_{ij} = q_i(x_j)$ and assigns (x_j, η_{ij}) to member \mathcal{P}_j .

Reconstruction. Suppose that there is a function $Q(x)$, such that $Q(x) = \sum_{j=1}^n q_j(x)$. Each member calculates the share $\eta_i = \sum_{j=1}^n \eta_{ji} = \sum_{j=1}^n Q(x_j)$. The shares of any t members are sufficient to reconstruct the function $Q(x)$ according to the Lagrange interpolating formula. The master secret S can be constructed by $S = Q(0)$.

2.4. Linear Secret-Sharing Schemes. We make use of Linear Secret-Sharing Schemes (LSSSs) in [22]. A secret-sharing scheme Π defined on a set of parties \mathcal{P} is linear over \mathbb{Z}_p if

- (1) The shares for each party constitute a vector over \mathbb{Z}_p .
- (2) The matrix M with ℓ rows and n columns is called the share-generating matrix. And the function ρ maps M_i to a party $\rho(i)$, where $i \in [1, n]$. When it comes to the column vector $\vec{v} = (s, r_2, r_3, \dots, r_n) \in \mathbb{Z}_p^n$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, r_3, \dots, r_n \in \mathbb{Z}_p$ are randomly chosen, then $M\vec{v}$ is the vector composed of ℓ shares of the secret in accordance with Π . The share $(M\vec{v})_i$ belongs to party $\rho(i)$.

Linear reconstruction is defined as follows: suppose that Π is an LSSS of the access structure \mathbb{A} . Let $S \in \mathbb{A}$ be any authorized set, and define $I \subset \{1, 2, \dots, \ell\}$ as $I = \{i: \rho(i) \in S\}$. Then, there exists a set of constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ that satisfy the proposition; if $\{\lambda_i\}$ are valid shares of any secret s according to Π , then $s = \sum_{i \in I} \omega_i \lambda_i$.

Definition 1 (Decisional Bilinear Diffie–Hellman Exponent Assumption (q-BDHE)). Let \mathbb{G} be a group of prime order p and g_i be short for g^{a^i} . Given $a, s \in \mathbb{Z}_p$ and $h = g^s$, the decision q-BDHE problem [30] can be defined as follows: the adversary is given a vector

$$\vec{y} = (g, g^s, g_1, \dots, g_q, g_{q+2}, \dots, g_{2q}), \quad (1)$$

and it is hard to distinguish $e(g_{q+1}, h) \in \mathbb{G}_T$ from a random element in \mathbb{G}_T . There is an algorithm \mathcal{B} that outputs $b \in \{0, 1\}$ with advantage ε in solving decisional q-BDHE in \mathbb{G} if

$$|\Pr[\mathcal{B}(\vec{y}, T = e(g, g)^{a^{q+1}s})] - \Pr[\mathcal{B}(\vec{y}, T = R)]| \geq \varepsilon. \quad (2)$$

The decisional q-BDHE assumption holds if there is no polynomial-time algorithm that can solve the (decision) q-BDHE problem with non-negligible advantage.

Mathematically, the Vieta's theorem is used to express the relationship between the root of a polynomial and its coefficients. In our schemes, it is a building block for computing the elements of the ciphertext/secret key.

Definition 2 (Vieta's theorem) (see [15]). Let $P_i = (a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0)$ represent a polynomial of degree n , and its coefficients are expressed as the vector

$$\vec{u} = (a_n, a_{n-1}, \dots, a_1, a_0). \quad (3)$$

For any \vec{x} , we represent as follows:

$$\vec{x} = \left(\underbrace{x \cdot x \dots x}_n, \underbrace{x \dots x}_{n-1}, \dots, x, 1 \right), \quad (4)$$

where element x is a root of P_i , if the inner product $\langle \vec{u}, \vec{x} \rangle = 0$. Suppose $\{x_i\}_{i \in [1, n]}$ are the roots of P_i ; then, we have

$$\begin{cases} x_1 + x_2 + \dots + x_n = -\left(\frac{a_{n-1}}{a_n}\right), \\ (x_1 x_2 + x_1 x_3 + \dots + x_n x_{n-1}) = \frac{a_{n-2}}{a_n}, \\ x_1 x_2 \dots x_n = (-1)^n \frac{a_0}{a_n}. \end{cases} \quad (5)$$

3. Multi-Authority Criteria-Based Encryption

3.1. System Model and Requirements. In this section, we define the notion of the system model and illustrate some requirements in our multi-authority criteria-based encryption schemes. As shown in Figure 2 [31], the system consists of a global central authority (CA), multiple criterion authorities (AAs), the edge infrastructures, data owners (DO), and data consumers (user). Here, we give the formal definition of them as follows.

- (1) The central authority (CA) in the whole system is considered to be completely trusted and in charge of system establishment and initialization, including the generation of system parameters and the master public key. When a user (or AA) requests registration, CA verifies the legitimacy of his identity and assigns a unique gid for the user and an aid for the AA, respectively. Besides, CA determines the threshold t in threshold sharing among attribute authorities, which is necessary for the process of secret key generation. In contrast, we note that CA is not responsible for any other issues in the system except for what has been described above. In other words, CA does not participate in the threshold sharing among AAs and key generation, which is the core of decentralization.
- (2) A criterion authority (AA) mainly generates the component of the user secret key associated with the criteria in its domain and plays a role in system establishment as well. What's worthy of mention is that, compared with common multi-authority CP-ABE, in our proposed system, all AAs manage the entire criterion universe together. We use the technique of threshold sharing among AAs so that each AA shares a piece of secret key calling its private key, which can ensure that a malicious user cannot get any information unless the number of corrupted authorities exceeds t . After that, CA accepts public

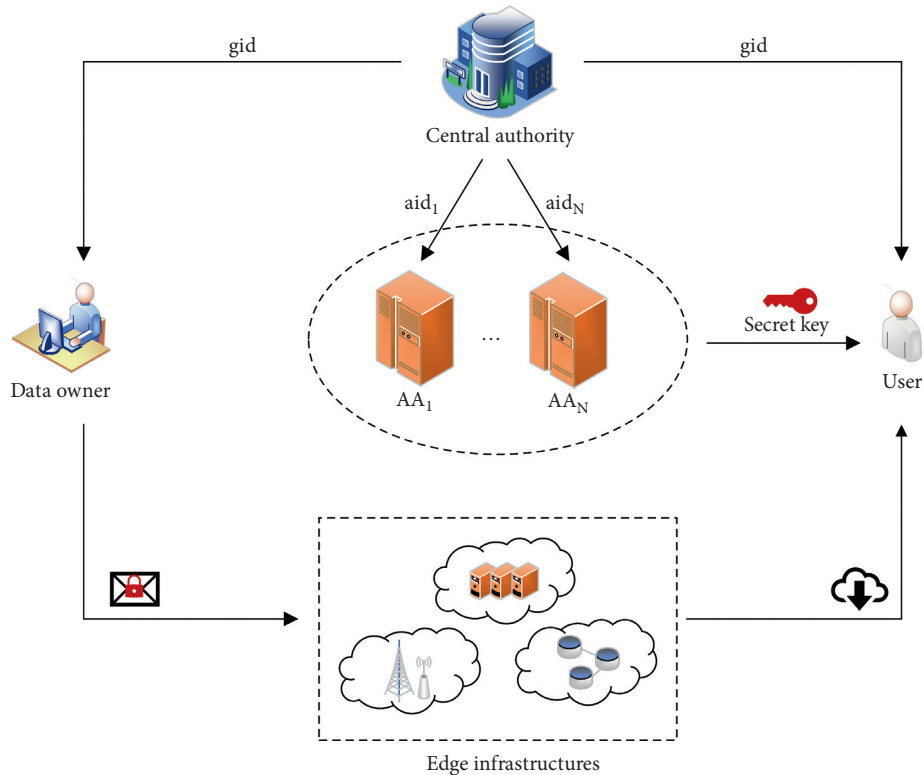


FIGURE 2: The framework of system model.

keys from all AAs to generate the system public key. Finally, when a user requests for his/her user's secret key, each AA only distributes its corresponding share of user secret key. Namely, there is no need for an AA to communicate with any other AA during the period of encryption and key generation.

- (3) A data owner (DO) encrypts the data. He/she specifies the access policy over criteria, the weight of each criterion, and the cumulative weight threshold that a user needs to satisfy. Concretely, DO runs the encryption algorithm and generates a ciphertext associated with all these requirements above and then uploads the ciphertext to edge infrastructure.
- (4) The user obtains a global identity gid issued by CA and AAs. Besides, any user in the system can download the encrypted data but can get access to the plaintext only when he/she satisfies both the access policy and weight requirement that the data owner specifies.
- (5) Each edge infrastructure is an entity that provides storage and computing services for DO. It accepts encrypted data sent by DO. Then, the data can be obtained by any registered user in the system.

For precision and unambiguity, some default definitions and requirements in our proposed schemes are provided here. In the system model, we suppose that CA is unconditionally credible and cannot be compromised. On the other hand, a user can download whichever encrypted data he wants but can recover the corresponding plaintext if and

only if he/she satisfies both the access policy and the cumulative weight threshold. Moreover, since the weights reflect the difference in importance among criteria when formulating an access policy, the ideal situation is that the user criteria that satisfy the policy contain more relatively significant (higher weight) criteria rather than a simple patchwork of low-weight criteria. Therefore, we consider that data owners are all sufficiently rigorous to design access policies, endue weight on each criterion, and set the thresholds over criteria. Furthermore, there are at least two authorities in the system.

3.2. Syntax of Scheme. The syntax of the multi-authority criteria-based encryption scheme consists of the following PPT algorithms:

- (1) $GlobalSetup(1^\lambda) \rightarrow pp$: the algorithm is performed by CA. It takes as input security parameter λ . It consists of three steps. CA first performs the group generation algorithm $\mathcal{G}(1^\lambda)$ to obtain $GP = (\mathbb{G}, \mathbb{G}_T, e, g, p)$ and defines criterion universe U with size n . Then, it chooses $\varphi_i \in_R \mathbb{Z}_p$ to label each polynomial P_i . Eventually, CA receives registration requests from users and AAs and records the number of AAs as n_θ . It outputs public parameter $pp = (GP, n, t, \{\varphi_i\}_{i \in [1, n]}, n_\theta)$.
- (2) $AASetup(pp) \rightarrow (pk_\theta, sk_\theta)$: the algorithm is performed by CA. For each authority AA_θ , it first chooses α_θ at random, such that $\alpha = \sum_{\theta=1}^{n_\theta} \alpha_\theta$. Note that the value of α is secret to any AA_θ . Then, all the

authorities run (t, n) -threshold secret sharing according to α_θ . Each authority AA_θ outputs pk_θ and keeps sk_θ as secret.

- (3) $CASetup(pp, \{pk_\theta\}_{\theta \in [1, n_\theta]}, d) \longrightarrow (PK, MSK)$: the algorithm is performed by CA and AA. It receives public parameter pp , public keys $\{pk_\theta\}_{\theta \in [1, n_\theta]}$ from all the AAs, and degree d of polynomials. It outputs public key PK and implicitly keeps values (α, a) for secret.
- (4) $Encrypt(PK, m, (\mathbb{A}, \rho), \vec{w}, \tau) \longrightarrow CT$: the algorithm is performed by DO. It takes in public parameter pp , the public key PK, a message m , an access structure (\mathbb{A}, ρ) , a weight vector \vec{w} , and weight threshold τ . It outputs a ciphertext CT.
- (5) $KeyGen(pp, PK, gid, C_{gid}) \longrightarrow SK_{gid}$: the algorithm is performed by the user with identity gid . It takes in pp , public key PK, the global identity gid of a user, and the set of cases C_{gid} belonging to the user. It outputs a SK_{gid} .
- (6) $Decrypt(pp, SK_{gid}, CT) \longrightarrow (m/\perp)$: it takes in the public parameter pp , the secret key SK_{gid} , and the ciphertext CT. The algorithm outputs either a message m or the distinctive symbol \perp .

For the correctness of our schemes, we require that for the $CT \leftarrow Encrypt(PK, m, (\mathbb{A}, \rho), \vec{w}, \tau)$ and the $SK_{gid} \leftarrow KeyGen(pp, PK, gid, C_{gid})$, one can execute $Decrypt(pp, SK_{gid}, CT)$ algorithm to obtain the correct message m with overwhelming probability.

3.3. Security Model. Here, the IND-CPA security [16] for proposed scheme is defined in the following game which has a challenger \mathcal{C} and an adversary \mathcal{A} .

Init. \mathcal{C} performs the algorithm GlobalSetup, AA Setup, and CA Setup and then sends the pp and PK to \mathcal{A} .

Phase 1. \mathcal{A} repeatedly performs private key associated with sets of case C .

Challenge. \mathcal{A} specifies two messages $m_0, m_1 \in \mathbb{G}_T$, a challenge access structure \mathbb{A}^* , a vector \vec{w}^* , and a weight threshold τ^* to \mathcal{C} . The default condition is that \mathcal{C} cannot satisfy the access structure \mathbb{A}^* . Then, \mathcal{C} randomly picks an element $b \in \{0, 1\}$ and executes Encrypt algorithm to generate $m_b \in \mathbb{G}_T$ under \mathbb{A}^* . Finally, \mathcal{A} obtains the ciphertext CT^* from \mathcal{C} .

Phase 2. \mathcal{A} can repeatedly make the same queries as Phase 1, except that \mathcal{C} cannot satisfy \mathbb{A}^* .

Guess. The adversary outputs a guess b' of b .

The advantage of the adversary \mathcal{A} in this game is defined as $\Pr[b' = b] - 1/2$.

Definition 3. The proposed multi-authority criteria-based encryption scheme is secure if all polynomial-time adversaries have at most a negligible advantage in the above game.

4. Construction

In this section, we first provide an overview of the proposed schemes and then give the detailed constructions of the two schemes.

4.1. Overview. What we first consider is how to find a form to express the criteria. In our schemes, the criterion is related to a polynomial, and each root of the polynomial corresponds to a case that satisfies the criteria. The first scheme requires that the user satisfies at least one case of the criterion, while in the second, there is a stricter restriction that the user must satisfy all cases of the criteria. In this context, our scheme improves the flexibility of access policy in practical application. Specifically, recall the access policy described in Figure 1. DO specifies an access policy $\mathbb{A} = (P_1)AND (P_2)OR (P_3)OR (P_4)$, and the cumulative weight threshold is set to $\tau = 6$. The observation is that the criterion set with cumulative weight exceeding τ can be expressed as $T = \{(4), (1, 2), (1, 3), (1, 4), (2, 4), (3, 4), (1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4), (1, 2, 3, 4)\}$. Clearly, Alice is a chief physician who has served the community for seven years. The case set and criterion set can be described as $C_{Alice} = \{1, 2\}; S_{Alice} = \{(1), (2), (1, 2)\}$. She can successfully decrypt the data due to the fact that set $C_{Alice} \models \mathbb{A}$ (i.e., $S_{C_{Alice}} = \{(1, 2)\}$ and $W_{Alice} = T \cap S_{C_{Alice}} = \{(1, 2)\}$). Bob is a community manager hired by the government and has 6 years of work experience related to medical treatment. He cannot decrypt the message successfully, since $W_{Bob} = T \cap S_{C_{Bob}} = \emptyset$.

From the practical perspective, the first scheme is suitable for edge computing platforms, while the second is suitable for users' private edge devices because those devices are more vulnerable to attacks by adversaries. Moreover, we introduce the multi-authority mechanism to solve the security problem caused by all attributes being managed by one authority. In this work, the criterion universe is jointly managed by n_θ AAs. The restriction is that there is no collusion between AAs. Specifically, CA cannot interact with users except for generating global unique identities for them. The user can reconstruct the secret key, which has the term of $e(g, g)^\alpha$, after interacting with t different AAs. This way, we make it impossible for each AA to generate a valid key individually. Meanwhile, data owners can assign a reasonable weight for each criterion and the cumulative weight according to their requirements, which makes the scheme suitable for real application scenarios.

4.2. MA-CE-Verify Root Scheme. Here, we provide our first multi-authority criteria-based encryption scheme that requires the user to have at least one root of a polynomial (or criterion).

- (1) $GlobalSetup(1^\lambda) \longrightarrow pp$: CA first runs $\mathcal{G}(1^\lambda)$ to obtain $GP = (\mathbb{G}, \mathbb{G}_T, e, g, p)$, where g is a generator of \mathbb{G} and \mathbb{G} and \mathbb{G}_T are two multiplicative cyclic groups with the same order p , such that $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. Then, CA defines the criterion universe U with size n and chooses $\varphi_i \in_R \mathbb{Z}_p$ to label each

polynomial P_i . Moreover, CA receives registration requests from AAs and users, records the number of AAs as n_θ , and generates the global unique identity aid, $\text{gid} \in \mathbb{Z}_p$ for AA and user, respectively. At last, CA defines threshold t according to the value n_θ . It outputs public parameter $pp = (\text{GP}, n, t, \{\varphi_i\}_{i \in [1, n]}, n_\theta)$.

- (2) AA Setup (pp) \longrightarrow (pk_θ, sk_θ): firstly, each authority AA_θ ($\theta \in [1, n_\theta]$) chooses the secret $\alpha_\theta \in_R \mathbb{Z}_p$, such that master secret $\alpha = \sum_{\theta=1}^{n_\theta} \alpha_\theta$. Then, AA_θ randomly sets a polynomial $q_\theta(x)$ of degree $t-1$ which satisfies $\alpha_\theta = q_\theta(0)$. Other AA_ξ ($\xi = 1, 2, \dots, \theta-1, \theta+1, \dots, n_\theta$) obtains the value $s_{\theta\xi} = q_\theta(\text{aid}_\xi)$ calculated by AA_θ . Meanwhile, AA_θ calculates $s_{\theta\theta} = q_\theta(\text{aid}_\theta)$ for itself. Finally, AA_θ calculates its secret key $sk_\theta = \sum_{\xi=1}^{n_\theta} s_{\xi\theta}$ and public key $pk_\theta = e(g, g)^{sk_\theta}$.
- (3) CA Setup ($pp, \{pk_\theta\}_{\theta \in [1, n_\theta]}, d$) \longrightarrow (PK, MSK): CA randomly picks t of n_θ AAs' public keys. Additionally, it picks $h_1, \dots, h_n \in_R \mathbb{G}$ and calculates as

$$\begin{aligned} e(g, g)^\alpha &= \prod_{\theta=1}^t pk_\theta \prod_{\xi=1, \xi \neq \theta}^t \text{aid}_\xi / \text{aid}_\xi - \text{aid}_\theta \\ &= e(g, g) \sum_{\theta=1}^t sk_\theta \prod_{\xi=1, \xi \neq \theta}^t \text{aid}_\xi / \text{aid}_\xi - \text{aid}_\theta \end{aligned} \quad (6)$$

Then, CA selects $a \in_R \mathbb{Z}_p$ and calculates g^a . For criterion universe U , it picks a set of d -degree polynomials $\{P_i\}_{i \in [1, n]}$ with coefficients $(a_{i,d}, a_{i,d-1}, \dots, a_{i,0})$ and labels P_i with φ_i . In this case, the set of polynomials can be described as

$$\left\{ \begin{array}{l} \vec{P}_1 = \left(a_{1,d}, a_{1,d-1}, \dots, a_{1,0}, \frac{1}{\varphi_1} \right), \\ \vec{P}'_1 = \left(a_{1,d}, a_{1,d-1}, \dots, a_{1,0}, 1 \right), \\ \dots, \\ \vec{P}_n = \left(a_{n,d}, a_{n,d-1}, \dots, a_{n,0}, \frac{1}{\varphi_n} \right), \\ \vec{P}'_n = \left(a_{n,d}, a_{n,d-1}, \dots, a_{n,0}, 1 \right). \end{array} \right. \quad (7)$$

For $i = 1$ to n , CA computes $g^{\varphi_i} \vec{P}_i$ and $g^{\vec{P}'_i}$. It outputs

$$\text{PK} = \left(g, g^a, e(g, g)^\alpha, \left\{ g^{\varphi_i} \vec{P}_i, g^{\vec{P}'_i}, h_i \right\}_{i \in [1, n]} \right), \quad (8)$$

and keeps the values (α, a) for secret.

- (4) Encrypt (PK, m , (\mathbb{A}, ρ) , \vec{w}, τ) \longrightarrow CT: in this phase, the encryption algorithm sets the access policy

(\mathbb{A}, ρ) , where the size of the matrix \mathbb{A} is $\ell \times n$, and the function ρ maps \mathbb{A}_i to a criterion. Then, it specifies the weight vector $\vec{w} = (w_1, \dots, w_n)$, where the element w_i represents the weight of each criterion. Also, it takes $y_2, y_3, \dots, y_n \in_R \mathbb{Z}_p$ to construct vector $\vec{v} = (s, y_2, y_3, \dots, y_n) \in \mathbb{Z}_p^n$, where the first element $s \in_R \mathbb{Z}_p$ is the secret value to be shared. For $i = 1$ to ℓ , it computes $\lambda_i = \vec{v} \cdot M_i$. After completing the above processes, it computes the set $T = \{(k_1^i, k_2^i, \dots, k_{\mu_i}^i)\}$ according to weight threshold τ , where μ_i indicates the length of i -th subset and $k_j^i \in \{1, 2, \dots, n\}$ denotes index in U . Finally, the algorithm calculates

$$C_0 = m \cdot e(g, g)^{\alpha s},$$

$$C'_0 = g^s,$$

$$\begin{aligned} &\cdot \left\{ C_i = g^{a\lambda_i} \cdot g^{-\varphi_{\rho(i)} \vec{P}_{\rho(i)}^s}, C'_i = g^{a\lambda_i} \cdot g^{-\vec{P}'_{\rho(i)} s} \right\}_{i \in [1, \ell]}, \\ &\cdot \left\{ \widehat{C}_i = \prod_{j=1}^{\mu_i} h_{k_j^i}^s \right\}_{i \in [1, \text{len}(T)]}. \end{aligned} \quad (9)$$

It outputs ciphertext as $\text{CT} = (C_0, C'_0, \{C_i, C'_i\}_{i \in [1, \ell]}, \{\widehat{C}_i\}_{i \in [1, \text{len}(T)]}, T)$.

- (5) KeyGen ($pp, \text{PK}, \text{gid}, C_{\text{gid}}$) \longrightarrow SK_{gid} : the key generation algorithm is implemented by the user interacting with t AAs according to the requirements. The restriction is that AA_θ cannot communicate with each other.

Let z_{φ_x} be a root of the polynomial at x . For each root z_{φ_x} that belongs to user, AA creates the vector $\vec{z}_{\varphi_x} = (z_{\varphi_x}^d, z_{\varphi_x}^{d-1}, \dots, z_{\varphi_x}, 1)$. We use $C_{\text{gid}} \subseteq \{C_x\}_{x \in [1, n]}$ to denote a set of cases, which belong to the user with gid . Let $P = \{P_1, P_2, \dots, P_{\text{len}(C_{\text{gid}})}\}$ denote the set of criteria requested by the user and $S = \{(P_1), (P_2), \dots, (P_{\text{len}(C_{\text{gid}})}), (P_1, P_2), \dots, (P_1, P_2, \dots, P_{\text{len}(C_{\text{gid}})})\} = \{k_1^1, \dots, k_{v_1}^1\}$ be all combinations of entities in set P , where v_1 denotes the length of subset and $k_j^1 \in [1, n]$ denotes index in U . AA_θ picks $\delta_\theta \in_R \mathbb{Z}_p$ and calculates as

$$L_\theta = g^{\delta_\theta},$$

$$\forall C_{\theta, x} \in C_{\text{gid}} K_{\theta, x} = (g^{\varphi_x})^{\vec{z}_{\varphi_x} \cdot \delta_\theta},$$

$$K'_{\theta, x} = g^{\vec{P}'_x \cdot \delta_\theta}, \quad (10)$$

$$\left\{ \widehat{K}_{\theta, i} = g^{sk_\theta} g^{a\delta_\theta} \prod_{j=1}^{v_i} h_{k_j^i}^{\delta_\theta} \right\}_{i \in [1, \text{len}(S)]}.$$

After interacting with t AAs, the user constructs the secret key as

$$\begin{aligned}
L &= \prod_{\theta=1}^t L_{\theta} \prod_{\xi=1, \xi \neq \theta}^t \text{aid}_{\xi} / \text{aid}_{\xi} - \text{aid}_{\theta} \\
&= g^{\sum_{\theta=1}^t \delta_{\theta} \cdot \prod_{\xi=1, \xi \neq \theta}^t \text{aid}_{\xi} / \text{aid}_{\xi} - \text{aid}_{\theta}}.
\end{aligned} \tag{11}$$

For all $C_{\theta, x} \in C_{\text{gid}}$, we have

$$\begin{aligned}
K_x &= \prod_{\theta=1}^t \left((K_{\theta, x})^{\vec{Z}_{\varphi_x} \cdot \delta_{\theta}} \right)^{\prod_{\xi=1, \xi \neq \theta}^t \text{aid}_{\xi} / \text{aid}_{\xi} - \text{aid}_{\theta}} = g^{\varphi_x \cdot \vec{Z}_{\varphi_x} \cdot \sum_{\theta=1}^t \delta_{\theta} \cdot \prod_{\xi=1, \xi \neq \theta}^t \text{aid}_{\xi} / \text{aid}_{\xi} - \text{aid}_{\theta}}, \\
K'_x &= \prod_{\theta=1}^t \left(g^{\vec{P}'_x \cdot \delta_{\theta}} \right)^{\prod_{\xi=1, \xi \neq \theta}^t \text{aid}_{\xi} / \text{aid}_{\xi} - \text{aid}_{\theta}} = g^{\vec{P}'_x \cdot \sum_{\theta=1}^t \delta_{\theta} \cdot \prod_{\xi=1, \xi \neq \theta}^t \text{aid}_{\xi} / \text{aid}_{\xi} - \text{aid}_{\theta}}, \\
\hat{K}_l &= \prod_{\theta=1}^t (\hat{K}_{\theta, l})^{\prod_{\xi=1, \xi \neq \theta}^t \text{aid}_{\xi} / \text{aid}_{\xi} - \text{aid}_{\theta}} = \prod_{\theta=1}^t \left(g^{sk_{\theta}} g^{a \delta_{\theta}} \prod_{j=1}^{v_l} h_{k_j}^{\delta_{\theta}} \right)^{\prod_{\xi=1, \xi \neq \theta}^t \text{aid}_{\xi} / \text{aid}_{\xi} - \text{aid}_{\theta}}.
\end{aligned} \tag{12}$$

For simplicity, we make $u = \sum_{\theta=1}^t \delta_{\theta} \cdot \prod_{\xi=1, \xi \neq \theta}^t \text{aid}_{\xi} / \text{aid}_{\xi} - \text{aid}_{\theta}$. For this, the secret key of the user can be represented as

$$\begin{aligned}
S, L &= g^u, \forall C_x \in C_{\text{gid}} K_x = g^{\varphi_x \cdot \vec{Z}_{\varphi_x} \cdot u}, K'_x = g^{\vec{P}'_x \cdot u}, \\
&\cdot \left\{ \hat{K}_l = g^{\alpha} \cdot g^{au} \cdot \prod_{j=1}^{v_l} h_{k_j}^u \right\}_{l \in [1, \text{len}(S)]}.
\end{aligned} \tag{13}$$

(6) $\text{Decrypt}(pp, SK_{\text{gid}}, CT) \rightarrow (m/\perp)$: the decryption algorithm can successfully be invoked by the user

with a valid identity. Namely, the user can download encrypted data from the edge infrastructures, and they can decrypt data successfully if their case set C_{gid} satisfies access policy and the requirement of cumulative weight.

Suppose that the ciphertext CT is encrypted under the access policy (\mathbb{A}, ρ) . We recall the definition of LSSS. Let $I \subset \{1, 2, \dots, \ell\}$ represent a case such that $\rho(i) \in C_{\text{gid}}$. To decrypt the ciphertext, the user with SK_{gid} computes $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$; if λ_i is valid share corresponding access policy (\mathbb{A}, ρ) , then the secret $s = \sum_{i \in I} \omega_i \lambda_i$ can be calculated. To summarize, the decryption process is as follows:

$$\begin{aligned}
&\prod_{i \in I} \left(e(C'_i, L) \cdot \frac{e(C_i, K_{\rho(i)}) \cdot e(C'_0, K_{\rho(i)'})}{e(C'_i, K_{\rho(i)})} \right)^{\omega_i} \\
&= \prod_{i \in I} \left(e(g^{a \lambda_i} \cdot g^{-\vec{P}'_{\rho(i)'s}}, g^u) \cdot \frac{e(g^{a \lambda_i} \cdot g^{-\varphi_{\rho(i)} \cdot \vec{P}_{\rho(i)} \cdot s}, g^{\varphi_{\rho(i)} \cdot \vec{Z}_{\rho(i)} \cdot u}) \cdot e(g^s, g^{\vec{P}'_{\rho(i)'s} \cdot u})}{e(g^{a \lambda_i} \cdot g^{-\vec{P}'_{\rho(i)'s}}, g^{\varphi_{\rho(i)} \cdot \vec{Z}_{\varphi_{\rho(i)} \cdot u})} \right)^{\omega_i} \\
&= \prod_{i \in I} \left(e(g^{a \lambda_i}, g^u) \cdot \frac{\left(g^{-\varphi_{\rho(i)} \cdot \vec{P}_{\rho(i)} \cdot s}, g^{\varphi_{\rho(i)} \cdot \vec{Z}_{\rho(i)} \cdot u} \right)}{e(g^{-\vec{P}'_{\rho(i)'s}}, g^{\varphi_{\rho(i)} \cdot \vec{Z}_{\varphi_{\rho(i)} \cdot u})} \right)^{\omega_i} \\
&= e(g, g)^{as u}.
\end{aligned} \tag{14}$$

Define set $W = \{T \cap S_{C \in \mathbb{A}}\}$. For each $w \in W$, let w_T and w_S denote the index w in set T and S , respectively. Then, compute

$$\begin{aligned} J &= \prod_{w \in W} \left(\frac{e(C'_0, \widehat{K}_{w_S})}{e(\widehat{C}_{w_T}, L)} \right)^{1/W} \\ &= \prod_{w \in W} \left(\frac{e(g^s, g^\alpha \cdot g^{au} \prod_{j=1}^{\text{len}(w)} h_{w_j}^u)}{e(\prod_{j=1}^{\text{len}(w)} h_{w_j}^s, g^u)} \right)^{1/W} \\ &= \prod_{w \in W} e(g^s, g^\alpha \cdot g^{au})^{1/W} \\ &= e(g, g)^{\alpha s} \cdot e(g, g)^{\alpha u}. \end{aligned} \quad (15)$$

The user can recover the plaintext m from the following equation:

$$m = C_0 \cdot \frac{e(g, g)^{\alpha s u}}{J}. \quad (16)$$

4.3. MA-CE-Root Equality Scheme. Here, we provide our second scheme, which needs all the roots (or cases) of each polynomial (or criterion) to be held by the user.

- (1) Global Setup (1^λ) \rightarrow pp : this algorithm is similar to scheme MA-CE-Verify Root. CA runs $\mathcal{G}(1^\lambda)$ to obtain $GP = (\mathbb{G}, \mathbb{G}_T, e, g, p)$ and defines the criterion universe U with size n . CA also generates unique identity for AAs and users, respectively. Then, it chooses a threshold t and picks $\{\varphi_i\}_{i \in [1, n]} \in_R \mathbb{Z}_p$. Note that φ_i is not used to label the polynomial P_i . It outputs public parameter $pp = (GP, n, t, \{\varphi_i\}_{i \in [1, n]}, n_\theta)$.
- (2) AA Setup (pp) \rightarrow (pk_θ, sk_θ) : the algorithm is similar to the AA Setup in the first scheme. For each authority AA_θ , it inputs the public parameter pp and returns a pair of keys (pk_θ, sk_θ) , where sk_θ is kept secret for other AAs.
- (3) CA Setup ($pp, \{pk_\theta\}_{\theta \in [1, n_\theta]}, d$) \rightarrow (PK, MSK) : CA randomly chooses t public keys from n_θ AAs. In addition, it picks $h_1, \dots, h_n \in_R \mathbb{G}$ and calculates

$$e(g, g)^\alpha = \prod_{\theta=1}^t pk_\theta \prod_{i=1, i \neq \theta}^t h_{i \cdot \text{aid}_i / \text{aid}_\theta - \text{aid}_\theta}. \quad (17)$$

Then, CA randomly picks $a \in \mathbb{Z}_p$ and computes g^a . For $i = 1$ to n , it picks a set of d -degree polynomials $\{P_i\}_{i \in [1, n]}$, which can be described as

$$\begin{cases} \vec{P}_1 = \left(1, \frac{a_{1,d-1}}{a_{1,d}}, \dots, \frac{a_{1,0}}{a_{1,d}} \right), \\ \vec{P}_2 = \left(1, \frac{a_{2,d-1}}{a_{2,d}}, \dots, \frac{a_{2,0}}{a_{2,d}} \right), \\ \vec{P}_n = \left(1, \frac{a_{n,d-1}}{a_{n,d}}, \dots, \frac{a_{n,0}}{a_{n,d}} \right). \end{cases} \quad (18)$$

It outputs $PK = (g, g^a, e(g, g)^\alpha, \{g^{\varphi_i} \vec{P}_i, h_i\}_{i \in [1, n]})$ and keeps the values (α, a) for secret.

- (4) Encrypt $(PK, m, (\mathbb{A}, \rho), \vec{w}, \tau) \rightarrow$ CT: the encryption algorithm sets the access policy (\mathbb{A}, ρ) , the size of the matrix is $\ell \times n$, and the function ρ maps \mathbb{A}_i to a criterion. Then, it specifies the weight vector $\vec{w} = (w_1, \dots, w_n)$, where the element w_i represents the weight of each criterion. Moreover, it constructs the vector $\vec{v} = (s, r_2, r_3, \dots, r_n) \in \mathbb{Z}_p^n$. For $i = 1$ to ℓ , it computes $\lambda_i = \vec{v} \cdot M_i$ and set $T = \{(k_1^i, k_2^i, \dots, k_{\mu_i}^i)\}$ according to the weight threshold τ , where μ_i indicates the length of i -th subset and $k_j^i \in \{1, 2, \dots, n\}$ denotes index in U . Finally, the algorithm computes as

$$\begin{aligned} C_0 &= m \cdot e(g, g)^{\alpha s}, \\ C_0' &= g^s \\ &\cdot \left\{ C_i = g^{a \lambda_i} \cdot g^{-\varphi_{\rho(i)}} \vec{P}_{\rho(i)^s} \right\}_{i \in [1, \ell]}, \\ &\cdot \left\{ \widehat{C}_i = \prod_{j=1}^{\mu_i} h_{k_j^i}^s \right\}_{i \in [1, \text{len}(T)]}. \end{aligned} \quad (19)$$

It outputs ciphertext as $CT = (C_0, C_0', \{C_i\}_{i \in [1, \ell]}, \{\widehat{C}_i\}_{i \in [1, \text{len}(T)]}, T)$.

- (5) Key Gen ($pp, PK, \text{gid}, C_{\text{gid}}$) \rightarrow SK_{gid} : the user with gid interacts with any t AAs to obtain the key according to requirements. It takes the set $\text{Roots}_x = \{x_1, x_2, \dots, x_d\}$ to represent all the roots of the polynomial at x . According to Vieta's theorem, AA uses Roots_x to construct the following vector:

$$\vec{z}_{\varphi_x} = (1, z_{x_{d-1}}, z_{x_{d-2}}, \dots, z_{x_0}). \quad (20)$$

Let $C_{\text{gid}} \subseteq \{C_x\}_{x \in [1, n]}$ represent the cases belonging to the user with identity gid , set $P = \{P_1, P_2, \dots, P_{\text{len}(C_{\text{gid}})}\}$ denote the set of criteria requested by the user, and set $S = \{(P_1), (P_2), \dots, (P_{\text{len}(C_{\text{gid}})}), (P_1, P_2), \dots, (P_1, P_2, \dots, P_{\text{len}(C_{\text{gid}})})\} = \{k_1^i, \dots, k_{\nu_i}^i\}$ be all combinations of entities in set P , where ν_i denotes the

length of l -th subset and $k_j^l \in [1, n]$ denotes index in U . AA_θ picks $\delta_\theta \in_R \mathbb{Z}_p$ and calculates as

$$\begin{aligned} L_\theta &= g^{\delta_\theta}, \\ \forall C_{\theta,x} \in C_{\text{gid}} K_{\theta,x} &= (g^{\varphi_x})^{\vec{Z}_{\varphi_x \cdot \delta_\theta}} \\ &\cdot \left\{ \widehat{K}_{\theta,l} = g^{sk_\theta} g^{a\delta_\theta} \prod_{j=1}^{v_l} h_{k_j^l}^{\delta_\theta} \right\}_{l \in [1, \text{len}(S)]}. \end{aligned} \quad (21)$$

After interacting with t AAs, the user constructs the secret key SK_{gid} as follows:

$$\begin{aligned} L &= g^u, \\ \forall C_x \in C_{\text{gid}} K_x &= (g^{\varphi_x})^{\vec{Z}_{\varphi_x \cdot u}} \\ &\cdot \left\{ \widehat{K}_l = g^\alpha g^{au} \prod_{j=1}^{v_l} h_{k_j^l}^u \right\}_{l \in [1, \text{len}(S)]}. \end{aligned} \quad (22)$$

(6) $\text{Decrypt}(pp, \text{SK}_{\text{gid}}, \text{CT}) \rightarrow (m/\perp)$: to recover the encrypted data under access policy (\mathbb{A}, ρ) , the user first calculates constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ and then computes

$$\begin{aligned} &\prod_{i \in I} (e(C_i, L) \cdot e(C'_0, K_{\rho(i)}))^{\omega_i} \\ &= \prod_{i \in I} \left(e(g^{a\lambda_i} \cdot g^{-\varphi_{\rho(i)}} \vec{P}_{\rho(i)^s}, g^u) \cdot e(g^s, (g^{\varphi_x})^{\vec{Z}_{\varphi_x \cdot u}}) \right)^{\omega_i} \\ &= \prod_{i \in I} e(g^{a\lambda_i}, g^u)^{\omega_i} \\ &= e(g, g)^{asu}. \end{aligned} \quad (23)$$

For $w \in W = \{T \cap S_{C \neq \mathbb{A}}\}$, the symbols w_T and w_S denote the index w in set T and S , respectively; then, compute

$$\begin{aligned} J &= \prod_{w \in W} \left(\frac{e(C'_0, \widehat{K}_{w_S})}{e(\widehat{C}_{w_T}, L)} \right)^{1/|W|} \\ &= e(g^s, g^\alpha \cdot g^{au})^{|W|/|W|} \\ &= e(g, g)^{\alpha s} \cdot e(g, g)^{asu}. \end{aligned} \quad (24)$$

The user can recover the plaintext $m = C_0 \cdot e(g, g)^{asu}/J$.

5. Security Proof

To prove the security of our constructions, the theorem in [8] is introduced as shown below.

Theorem 1. *If the decisional q -BDHE assumption holds, then any polynomial-time adversary cannot selectively break*

the MA-CE-Verify Root scheme with a challenge matrix of size $\ell^ \times n^*$, where $n^* \leq q$.*

Here, we briefly overview the proof technique under the decisional q -BDHE assumption. Suppose that there exists an adversary A with a nonnegligible advantage ε can selectively break the proposed scheme. \mathcal{A} is allowed to select a matrix with the size of at most $q \times q$. Here, the restriction is that the key queried from the challenger cannot decrypt the message. Then, we construct a PPT simulator B , which solves the q -BDHE assumption.

Init. \mathcal{B} first receipts the q -BDHE challenge $\vec{y} = (g, g^s, g_1, \dots, g_q, g_{q+2}, \dots, g_{2q})$ and \vec{T} . Then, \mathcal{A} sends the challenge structure (\mathbb{A}^*, ρ^*) , a weight vector $\vec{w}^* = \{w_1, w_2, \dots, w_m\}_{m \in [1, n]}$, and a weight threshold τ^* to \mathcal{B} .

Setup. In this phase, \mathcal{B} picks $\alpha' \in_R \mathbb{Z}_p$ and implicitly takes $\alpha = \alpha' + a^{q+1}$ by making $e(g, g)^\alpha = e(g^{\alpha'}, g) \cdot e(g, g)^{a^{q+1}}$. Then, \mathcal{B} randomly generates φ_x for polynomial P_x . The symbol Φ represents the collection of indexes i , such that $\rho^*(i) = P_x$. Next, \mathcal{B} takes

$$\begin{aligned} \psi_x &= g^{\varphi_x} \vec{P}_x \prod_{i \in \Phi} g^{a\mathbb{A}_{i,1}^*} \cdot g^{a^2\mathbb{A}_{i,2}^*} \dots g^{a^{n^*}\mathbb{A}_{i,n^*}^*}, \\ \beta_x &= g^{\vec{P}'_x} \prod_{i \in \Phi} g^{a\mathbb{A}_{i,1}^*} \cdot g^{a^2\mathbb{A}_{i,2}^*} \dots g^{a^{n^*}\mathbb{A}_{i,n^*}^*}, \\ \gamma_x &= g^{\varphi_x} \prod_{i \in \Phi} g^{a\mathbb{A}_{i,1}^*} \cdot g^{a^2\mathbb{A}_{i,2}^*} \dots g^{a^{n^*}\mathbb{A}_{i,n^*}^*}. \end{aligned} \quad (25)$$

We note that $\psi_x = g^{\varphi_x} \vec{P}_x$, $\beta_x = g^{\vec{P}'_x}$ and $\gamma_x = g^{\varphi_x}$ if $\Phi = \emptyset$. Finally, \mathcal{B} chooses $\eta_1, \eta_2, \dots, \eta_n \in_R \mathbb{Z}_p$ and computes

$$h_j = \begin{cases} g^{\eta_j/\sigma}, & \text{if } P_x \text{ has combination } \leq \tau^*, \\ g^{\eta_j}, & \text{otherwise,} \end{cases} \quad (26)$$

which implicitly takes $\sigma = w_1 + w_2 + \dots + w_m$.

Phase 1. \mathcal{B} replies private key queries for $C = \{z_{\varphi_x}\}$, where C cannot satisfy (\mathbb{A}^*, ρ^*) . For each z_{φ_x} , \mathcal{B} first creates vector $\vec{z}_{\varphi_x} = \{z_{\varphi_x}^d, z_{\varphi_x}^{d-1}, \dots, z_{\varphi_x}, 1\}$ and chooses $r \in_R \mathbb{Z}_p$. Then, according to the definition of LSSS, \mathcal{B} calculates a vector $\vec{w} = (w_1, w_2, \dots, w_{n^*}) \in \mathbb{Z}_p^{n^*}$ such that $w_1 = -1$. For all i such that $\rho^*(i) \in S$, we have that the inner product $\langle \vec{w}, \mathbb{A}_i^* \rangle = 0$. Finally, \mathcal{B} implicitly defines u as

$$u = r + w_1 a^q + w_2 a^{q-1} + \dots + w_{n^*} a^{(q-n^*+1)}. \quad (27)$$

Therefore, the value L can be denoted as

$$L = g^r \prod_{i=1}^{n^*} (g^{a^{-i+1}})^{\omega_i} = g^u. \quad (28)$$

We now consider $z_{\varphi_x} \in S$ for the case that there is no i such that $\rho^*(i)$ has a root equal to z_{φ_x} . \mathcal{B} can simply take $K_x = L^{\varphi_x} \vec{z}_{\varphi_x}$. Otherwise, it calculates as

$$K_x = \left(g^{\varphi_x} \prod_{i \in \Phi} g^{a^{\mathbb{A}_{i,1}^*}} \cdot g^{a^2 \mathbb{A}_{i,2}^*} \dots g^{a^{n^*} \mathbb{A}_{i,n^*}^*} \right)^{\vec{z}_{\varphi_x u}}. \quad (29)$$

Note that by defining u , K_x has the form of $\mathbb{A}_{i,j}^* a^j w_j a^{q-j+1}$ in the exponent for some j . However, we have that $\langle \vec{w}, \mathbb{A}_i^* \rangle = 0$, and the term of $g^{a^{q+1}}$ can be cancelled. Consequently, K_x can be expressed as

$$\begin{aligned} K_x &= \left(g^{\varphi_x} \prod_{i \in \Phi} g^{a^{\mathbb{A}_{i,1}^*}} \cdot g^{a^2 \mathbb{A}_{i,2}^*} \dots g^{a^{n^*} \mathbb{A}_{i,n^*}^*} \right)^{\vec{z}_{\varphi_x u}} \\ &= L^{\varphi_x} \vec{z}_{\varphi_x} \prod_{i \in \Phi} \prod_{j=1}^{n^*} g^{a^j \cdot r} \cdot \left(\prod_{k=1, k \neq j}^{n^*} g^{a^{q+j-k+1}} \right)^{\mathbb{A}_{i,j}^*}. \end{aligned} \quad (30)$$

We now consider simulating the value of \widehat{K}_l . Let $P = \{P_1, P_2, \dots, P_{\text{len}(C_{\text{gid}})}\}$ be the set of criteria corresponding to the criterion universe U and $S = \{k_1^l, \dots, k_{v_l}^l\}$ ($l \in [1, 2^{\text{len}(C_{\text{gid}})}]$) be all combinations of entities in set P . For $l = 1$ to $2^{\text{len}(C_{\text{gid}})}$, we have

$$\begin{aligned} \widehat{K}_l &= g^\alpha g^{au} \prod_{j=1}^{v_l} g^{\eta_{k_j} u} \\ &= g^{\alpha'} \cdot g^{au} \cdot g^{ar} \cdot g^{a^{q+1} w_1} \cdot g^{a^q w_2} \dots g^{a^{q-n^*+2} w_{n^*}} \cdot L^{\sum_{j=1}^{v_l} \eta_{k_j}} \\ &= g^{\alpha'} \cdot g^{ar} \cdot g^{a^q w_2} \dots g^{a^{q-n^*+2} w_{n^*}} \cdot L^{\sum_{j=1}^{v_l} \eta_{k_j}} \\ &= g^{\alpha'} \cdot g^{ar} \prod_{l=2}^{n^*} g^{a^{q-l+2} w_l} \cdot L^{\sum_{j=1}^{v_l} \eta_{k_j}}. \end{aligned} \quad (31)$$

Otherwise, we have

$$\begin{aligned} \widehat{K}_l &= g^\alpha g^{au} \prod_{j=1}^{v_l} g^{\eta_{k_j} t / \sigma} \\ &= g^{\alpha'} \cdot g^{a^{q+1}} \cdot g^{ar} \cdot g^{a^{q+1} w_1} \cdot g^{a^q w_2} \dots g^{a^{q-n^*+2} w_{n^*}} L^{\sum_{j=1}^{v_l} \eta_{k_j} / \sigma} \\ &= g^{\alpha'} \cdot g^{ar} \cdot \prod_{l=2}^{n^*} \left(g^{a^{q-l+2}} \right)^{w_l} L^{\sum_{j=1}^{v_l} \eta_{k_j} / \sigma}. \end{aligned} \quad (32)$$

Challenge. We show how to build challenge ciphertext. \mathcal{A} submits two messages m_0 and m_1 to \mathcal{B} . The simulator \mathcal{B} selects $b \in \{0, 1\}$ at random and constructs $C_0 = m_b \cdot \vec{T} \cdot e(g^{\alpha'}, h)$, $C'_0 = h$. Then, it picks $y'_2, y'_3, \dots, y'_{n^*} \in_R \mathbb{Z}_p$ and secret s using the vector

$$\vec{v} = (s, sa + y'_2, sa^2 + y'_3, \dots, sa^{n-1} + y'_{n^*}) \in \mathbb{Z}_p^{n^*}. \quad (33)$$

Finally, \mathcal{B} chooses threshold value τ^* and performs Encrypt algorithm to construct C_i, C'_i , and \widehat{C}_i as follows:

$$\begin{aligned} C_i &= g^a \vec{v}^{\mathbb{A}_i^*} \cdot g^{-\varphi_i} \vec{P}_i^s \\ &= g^{sa^{\mathbb{A}_{i,1}^*}} \cdot g^{(sa^2 + ay'_2)^{\mathbb{A}_{i,2}^*}} \dots g^{(sa^{n^*} + ay'_{n^*})^{\mathbb{A}_{i,n^*}^*}} \cdot (g^s)^{-\varphi_i} \vec{P}_i \cdot \left(g^{a^{\mathbb{A}_{i,1}^*}} \cdot g^{a^2 \mathbb{A}_{i,2}^*} \dots g^{a^{n^*}} \right)^{-s} \\ &= (g^s)^{-\varphi_i} \vec{P}_i \left(\prod_{j=1}^{n^*} (g^a)^{\mathbb{A}_{i,j}^*} y'_j \right), \\ C'_i &= g^a \vec{v}^{\mathbb{A}_i^*} \cdot g^{-\vec{P}_i^s} \\ &= g^{sa^{\mathbb{A}_{i,1}^*}} \cdot g^{(sa^2 + ay'_2)^{\mathbb{A}_{i,2}^*}} \dots g^{(sa^{n^*} + ay'_{n^*})^{\mathbb{A}_{i,n^*}^*}} \cdot (g^s)^{-\vec{P}_i} \cdot \left(g^{a^{\mathbb{A}_{i,1}^*}} \cdot g^{a^2 \mathbb{A}_{i,2}^*} \dots g^{a^{n^*}} \right)^{-s} \\ &= (g^s)^{-\vec{P}_i} \left(\prod_{j=1}^{n^*} (g^a)^{\mathbb{A}_{i,j}^*} y'_j \right), \left\{ \widehat{C}_i = \prod_{j=1}^{\mu_i} \mathcal{B}_{k_j^i}^s \right\}_{i \in [1, \text{len}(T)]}. \end{aligned} \quad (34)$$

Phase 2. \mathcal{A} can adaptively make queries the same as Phase 1 with the restriction that none of those cases satisfy the access structure corresponding to the Challenge phase.

Guess. The adversary \mathcal{A} eventually outputs a guess bit $b' \in \{0, 1\}$ of b . If \mathcal{A} correctly guesses $b' = b$, then \mathcal{B} returns 0 to guess that $\vec{T} = e(g, g)^{a^{q+1} s}$; otherwise, it outputs 1 to

demonstrate that it considers \tilde{T} is a random element obtained from group \mathbb{G}_T . When \tilde{T} is a tuple, the simulator \mathcal{B} performs a perfect simulation. In this case, we have that

$$\Pr\left[\mathcal{B}\left(\vec{y}, \tilde{T} = e(g, g)^{a^{q+1}s}\right) = 0\right] = \frac{1}{2} + A \text{ dv}_{\mathcal{A}}. \quad (35)$$

When \tilde{T} is a random element in \mathbb{G}_T , \mathcal{B} simulates a completely random challenge ciphertext for adversary \mathcal{A} , and we have

$$\Pr[\mathcal{B}(\vec{y}, \tilde{T} = R) = 0] = \frac{1}{2}. \quad (36)$$

Consequently, \mathcal{B} can play the decisional q-BDHE game with non-negligible advantage.

Theorem 2. *If the decisional q-BDHE assumption holds, then no polynomial-time adversary can selectively break our MA-CE-Root Equality scheme with a challenge matrix of size $\ell^* \times n^*$, where $n^* \leq q$.*

The proof of this theorem is similar to Theorem 1 (here we omit the proof process).

6. Performance Analysis

We now provide theoretical analysis and implementation evaluation of the two schemes in this section.

6.1. Theoretical Analysis. There is the comparison of the four schemes, including [12–14] and our two schemes, in terms of storage overhead and computation cost. Let \mathbb{P} indicate a pairing operation. \mathbb{E} and \mathbb{E}_T denote an exponential operation of group \mathbb{G} and \mathbb{G}_T , respectively. $|g|$ and $|g_T|$ represent the size of elements in group \mathbb{G} and \mathbb{G}_T , respectively. In our schemes, N represents the size of the criterion universe, while it represents attribute universe in [12–14]. n_ℓ and n_u denote the number of criteria (or attributes) in the access matrix and the number of criteria that are satisfied by the user, respectively. Let n_a denote the number of attributes managed by attribute authority. l is the number of all criterion sets with cumulative weight greater than τ . l_w is the size of the criterion set that satisfies the access policy and cumulative weight.

We first compare the storage overhead of the four schemes, as shown in Table 1. In terms of ciphertext size, our schemes are better than [12–14], since they require storing a large amount of leaf nodes information of the access tree. It can be observed that [13] is superior to our schemes in terms of key size and public key size. The reason is that the public key in our schemes needs to contain information corresponding to the criterion. All weights are specified by the trusted authority TA in [13]. Different from [13], the Key Gen phase of our schemes requires enumerating the criterion set that exceeds the weight. The performance of our schemes in terms of key size is comparable to that of [14].

However, the scheme in [14] cannot support multiple authorities, and the weight of each attribute is specified by TA. This inevitably limits the ability of the scheme in practical scenarios.

Table 2 shows the computation cost of these schemes in Key Gen, Encrypt, and Decrypt phases. In the Key Gen phase, the scheme in [13] performs better than other schemes, because the calculation of all the criterion sets takes up the main computation cost in our schemes and the scheme of [12]. In Encrypt and Decrypt phases, our schemes cost less time than [13] in practical application, since the computation cost of the latter is occupied by a large number of exponential and pairing operations. Moreover, it can be seen that the performance of [14] is similar to our first scheme and slightly inferior to the second scheme. The advantage of our schemes is that users can flexibly choose the weights in the access policy according to different application scenarios.

6.2. Implementation and Evaluation. We implement the proposed schemes in Charm [32] using Python 3.6.5. The programs adopt the Pairing-Based Cryptography (PBC) library version-0.5.14. We pick the symmetric curve with a 512 bit base field, and it provides 160 bit group order. All our programs were executed on VMware @ Workstation Pro 15.5.5 with a dual core Intel (R) Core (TM) i7-7700HQ CPU @2.8 GHz and 2.0 GB RAM running Ubuntu 18.04. All experimental results are taken from the average value of the program executed 20 times.

Figure 3 shows the value of key generation time with threshold t ($t = 1, 2, \dots, 10$). We set the number of AAs to 10 in the system. As known from the figure, with the increase of threshold t , the time consumed for key generation is fixed basically, due to the fact that the user requests keys from t AAs in the meantime, while the time consumption of each AA for calculating subshare of a key is almost the same. Moreover, the value of t is generally within 10 in actual application scenarios. In summary, it can be considered that the time consumption is hardly affected by the threshold t in the KeyGen phase.

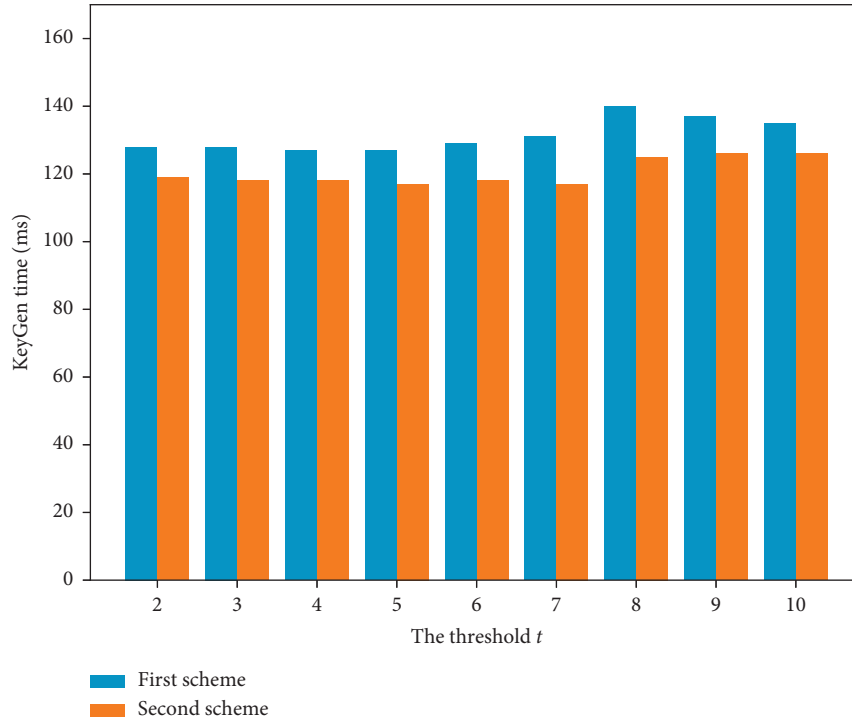
Figure 4 shows the time consumption of Key Gen, Encrypt, and Decrypt algorithms as the number of user attributes increases in the proposed schemes. We take the number n_θ of AAs as 10 and the threshold τ as 6. The performance of scheme-2 is slightly better than scheme-1 because the former has shorter ciphertext and key, which reduces exponential and pairing operations. We observe that the time consumption of each stage shows a nonlinear increasing trend. What mainly affects computational efficiency are summarized as follows. The first aspect is that the encryption algorithm needs to calculate all cases T that exceed the cumulative threshold τ . Another reason is that calculating the criteria set S that belong to the user dominates the execution time of the key generation algorithm (see Section 4 (Key Gen)). In addition, it takes a

TABLE 1: Comparison of storage overhead.

Schemes	PK size	Ciphertext size	Key size
LMX13 [12]	$(2N + 2) g + g_T $	$(4n_\ell + 4) g + g_T $	$7n_u g $
WZZ14 [13]	$(N + 2) g + g_T $	$(n_\ell n_a + 1) g + g_T $	$(n_u + 1) g $
YYZ20 [14]	$(N + 2) g + g_T $	$(3n_\ell + 2) g + g_T $	$(2n_u + 4)$
Our scheme-1	$(3N + 2) g + g_T $	$(2n_\ell + l) g + g_T $	$(2n_u + l + 1) g $
Our scheme-2	$(2N + 2) g + g_T $	$(n_\ell + l) g + g_T $	$(n_u + l + 1) g $

TABLE 2: Comparison of computation cost.

Schemes	KeyGen	Encrypt	Decrypt
LMX13 [12]	$7n_u E$	$E_T + (4n_\ell + 4)E + P$	$(5n_u + 1)P$
WZZ14 [13]	$(n_u + 1)E$	$E_T + (n_\ell n_a + 1)E + P$	$(n_u l_w + 1)P$
YYZ [14]	$(2n_u + 4)E$	$E_T + (3n_\ell + 2)E + P$	$(3n_u + 1)P$
Our scheme-1	$(2n_u + l + 1)E$	$E_T + (4n_\ell + l + 1)E + P$	$(4n_u + 2l_w)P$
Our scheme-2	$(n_u + l + 1)E$	$E_T + (2n_\ell + l + 1)E + P$	$(2n_u + 2l_w)P$

FIGURE 3: The value of key generation time with threshold t .

relatively long time to evaluate the intersection of set T and S in the decryption phase. Nevertheless, our schemes enjoy tolerable computational efficiency for the following reasons. Clearly, the time consumption does not exceed 130 ms in all phases. To be precise, when a user owns 30 attributes, the time consumption of the first scheme is

123 ms, while that of the second scheme is 120 ms. Therefore, the efficiency of our proposed schemes is acceptable in practical scenarios. Furthermore, we remark that in the IoT scenario, the relatively intensive computation can be offloaded to some outsourced equipment, and the rest of the operations remain on the receiver.

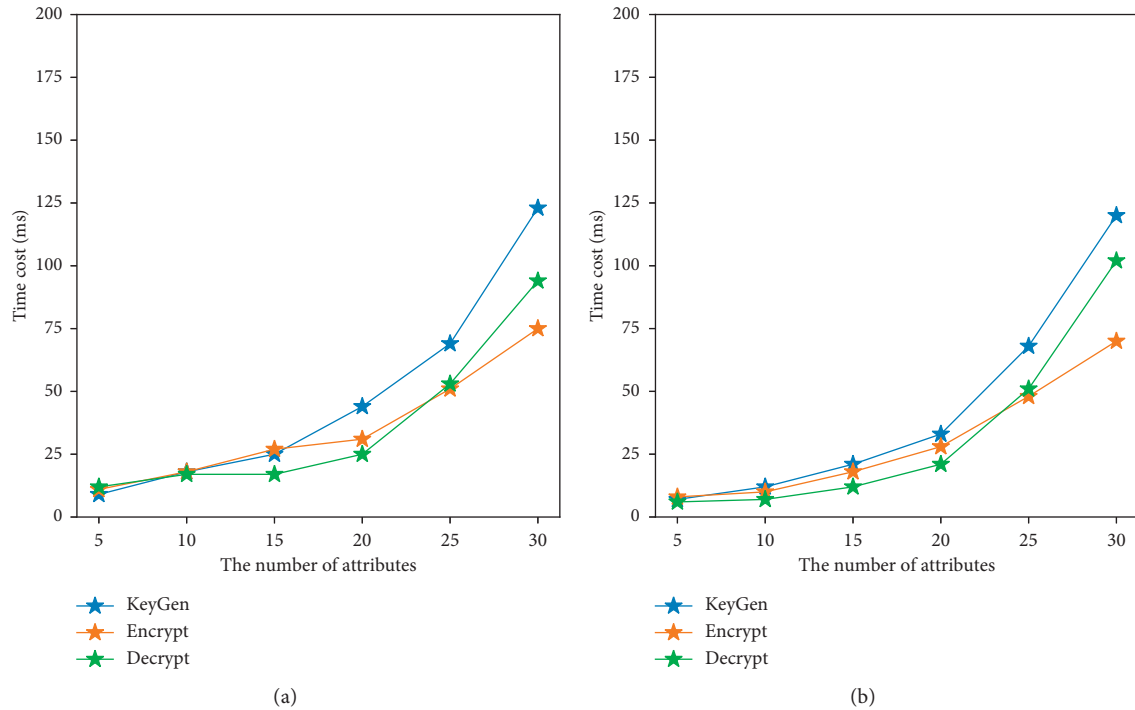


FIGURE 4: Time cost of the two proposed schemes.

7. Conclusion

In this paper, we propose two multi-authority criteria-based encryption schemes that support data access control in IoT and are proved to be secure in the standard model. Specifically, they solve the problem of security bottleneck and server overload caused by involving only a single authority in the phase of key generation. Moreover, each criterion carries a weight specified by the encryptor, which allows the access policy to be expressed more flexibly. The theoretical analysis and simulation evaluation demonstrate that our schemes can conform to the actual application scenarios. The remaining problem is that the time consumption of each phase in the schemes increases nonlinearly, which limits the size of the criterion universe. In future work, we are committed to constructing more lightweight frameworks.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by the Fundamental Research Funds for the Central Universities (no. 3072020CFJ0601) and CSIC 722 Innovation Fund (no. KCJJ2019-12).

References

- [1] C. Wang, D. Wang, G. Xu, and D. He, *Efficient Privacy-Preserving User Authentication Scheme with Forward Secrecy for Industry 4.0*, Science China Information Sciences, Beijing, China.
- [2] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: a survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.
- [3] D. Wang, X. Zhang, Z. Zhang, and P. Wang, "Understanding security failures of multi-factor authentication schemes for multi-server environments," *Computers & Security*, vol. 88, Article ID 101619, 2020.
- [4] L.-Y. Yeh, P.-Y. Chiang, Y.-L. Tsai, and J.-L. Huang, "Cloud-based fine-grained health information access control framework for lightweight IoT devices with dynamic auditing and attribute revocation," *IEEE Transactions on Cloud Computing*, vol. 6, no. 2, pp. 532–544, 2018.
- [5] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, 2020.
- [6] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang, "Understanding node capture attacks in user authentication schemes for wireless sensor networks," *IEEE Transactions on Dependable and Secure Computing*, 2020.
- [7] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of the EUROCRYPT*, pp. 457–473, Aarhus, Denmark, May 2005.
- [8] B. Waters, "Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization," in *Proceedings of 14th International Conference on Practice and Theory in Public Key Cryptography (PKC'11)*, pp. 53–70, Taormina, Italy, March 2011.

- [9] N. Attrapadung, “Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more,” in *Proceedings of the EUROCRYPT*, pp. 457–473, Aarhus, Denmark, May 2014.
- [10] Q. Li, J. Ma, R. Li, J. Xiong, and X. Liu, “Provably secure unbounded multi-authority ciphertext-policy attribute-based encryption,” *Security and Communication Networks*, vol. 8, no. 18, pp. 4098–4109, 2015.
- [11] J. Li, S. Wang, and Y. Li, “An efficient attribute-based encryption scheme with policy update and file update in cloud computing,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6500–6509, 2019.
- [12] X. Liu, J. Ma, J. Xiong et al., “Ciphertext-policy weighted attribute-based encryption for fine-grained access control,” in *Proceedings of the International Conference on Intelligent Networking and Collaborative Systems*, pp. 51–57, IEEE, Xi’an, China, May 2013.
- [13] Y. Wang, D. Zhang, and H. Zhong, “Multi-authority based weighted attribute encryption scheme in cloud computing,” in *Proceedings of the International Conference on Natural Computation*, pp. 1033–1038, IEEE, Xiamen, China, August 2014.
- [14] X. Yan, X. Yuan, Q. Zhang, and Y. Tang, “Traceable and weighted attribute-based encryption scheme in the cloud environment,” *IEEE Access*, vol. 8, pp. 38285–38295, 2020.
- [15] T. V. X. Phuong, G. Yang, and W. Susilo, “Criteria-based encryption,” *The Computer Journal*, vol. 61, no. 4, pp. 512–525, 2018.
- [16] V. Goyal, O. Pandey, A. Sahai et al., “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 89–98, Alexandria, VA, USA, October 2006.
- [17] A. Sahai, H. Seyalioglu, and B. Waters, “Dynamic credentials and ciphertext delegation for attribute-based encryption,” in *Proceedings of the CRYPTO*, pp. 199–217, Santa Barbara, CA, USA, August 2012.
- [18] S. Agrawal and M. Chase, “Fast attribute-based message encryption,” in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 665–682, Dallas, TX, USA, October 2017.
- [19] B. Waters, “Functional encryption for regular languages,” in *Proceedings of the CRYPTO*, pp. 218–235, Santa Barbara, CA, USA, August 2012.
- [20] S. Agrawal, M. Maitra, and S. Yamada, “Attribute based encryption for deterministic finite automata from DLIN,” in *Proceedings of the Theory of Cryptography Conference*, pp. 91–117, Nuremberg, Germany, November 2019.
- [21] W. Li, K. Xue, Y. Xue, and J. Hong, “A robust and verifiable threshold multi-authority access control system in public cloud storage,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1484–1496, 2015.
- [22] A. Lewko and B. Waters, “Decentralizing attribute-based encryption,” in *Proceedings of the EUROCRYPT*, pp. 568–588, Tallinn, Estonia, May 2011.
- [23] J. Wei, W. Liu, and X. Hu, “Secure and efficient attribute-based access control for multi-authority cloud storage,” *IEEE Systems Journal*, vol. 12, no. 2, pp. 1731–1742, 2016.
- [24] Q. Li, J. Ma, R. Li, X. Liu, J. Xiong, and D. Chen, “Secure, efficient and revocable multi-authority access control system in cloud storage,” *Computers & Security*, vol. 59, pp. 45–59, 2016.
- [25] J. Li, X. Chen, S. S. M. Chow, Q. Huang, D. S. Wong, and Z. Liu, “Multi-authority fine-grained access control with accountability and its application in cloud,” *Journal of Network and Computer Applications*, vol. 112, pp. 89–96, 2018.
- [26] Q. M. Malluhi, A. Shikfa, V. D. Tran, and V. C. Trinh, “Decentralized ciphertext-policy attribute-based encryption schemes for lightweight devices,” *Computer Communications*, vol. 145, pp. 113–125, 2019.
- [27] V. K. A Sandor, Y. Lin, X. Li, F. Lin, and S. Zhang, “Efficient decentralized multi-authority attribute-based encryption for mobile cloud data storage,” *Journal of Network and Computer Applications*, vol. 129, pp. 25–36, 2019.
- [28] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [29] T. P. Pedersen, “A threshold cryptosystem without a trusted party,” in *Proceedings of the EUROCRYPT*, pp. 522–526, Brighton, UK, April 1991.
- [30] Y. Rouselakis and B. Waters, “Efficient statically-secure large-universe multi-authority attribute-based encryption,” *Financial Cryptography and Data Security*, in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 315–332, San Juan, Puerto Rico, January 2015.
- [31] <https://support.microsoft.com/zh-cn/visio>.
- [32] J. A. Akinyele, C. Garman, I. Miers et al., “Charm: a framework for rapidly prototyping cryptosystems,” *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.