

Security and Privacy Challenges for Intelligent Internet of Things Devices 2022

Lead Guest Editor: AnMin Fu

Guest Editors: Shui Yu, Jinguang Han, and Kaitai Liang





Security and Privacy Challenges for Intelligent Internet of Things Devices 2022

Security and Communication Networks

**Security and Privacy Challenges for
Intelligent Internet of Things Devices
2022**

Lead Guest Editor: AnMin Fu

Guest Editors: Shui Yu, Jinguang Han, and Kaitai
Liang






Copyright © 2022 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

Dual-Server Identity-Based Encryption with Authorized Equality Test for IoT Data in Clouds

Meng Zhao and Yong Ding 






Research Article (12 pages), Article ID 4905763, Volume 2022 (2022)

UIV-TSP: A Blockchain-Enabled Antileakage Sharing Protection Scheme for Undisclosed IIoT Vulnerabilities

Wenbo Zhang , Jing Zhang, Yifei Shi, and Jingyu Feng 








Research Article (17 pages), Article ID 2500213, Volume 2022 (2022)

DeepGuard: Backdoor Attack Detection and Identification Schemes in Privacy-Preserving Deep Neural Networks

Congcong Chen , Lifei Wei , Lei Zhang , Ya Peng , and Jianting Ning 






Research Article (20 pages), Article ID 2985308, Volume 2022 (2022)

TFPPASV: A Three-Factor Privacy Preserving Authentication Scheme for VANETs

Zongtao Duan , Jabar Mahmood , Yun Yang , Michael Abebe Berwo , Abd al Kader Ahmed Yassin , Muhammad Nasir Mumtaz Bhutta , and Shehzad Ashraf Chaudhry 

Research Article (15 pages), Article ID 8259927, Volume 2022 (2022)

Encoding Test Pattern of System-on-Chip (SOC) Using Annular Scan Chain

Guilin Huang , Zhengjin Zhang , Honghai Wang , Jiabao Jiang , and Qilin Wu 




Research Article (7 pages), Article ID 6974101, Volume 2022 (2022)

Blockchain-Assisted Distributed Fog Computing Control Flow Attestation

Hongchao Li , Tao Shen , Fenhua Bai , and Bei Gong



Research Article (17 pages), Article ID 6128155, Volume 2022 (2022)

Detecting Privacy Leakage of Smart Home Devices through Traffic Analysis

Ting Yang , Guanghua Zhang, Yin Li, Yiyu Yang, He Wang , and Yuqing Zhang 

Research Article (10 pages), Article ID 5655314, Volume 2022 (2022)

A Traceable and Anonymous Data Aggregation Scheme with Fog Computing Devices for Smart Grid

Fan Wu  and Xiong Li 






Research Article (10 pages), Article ID 4548374, Volume 2022 (2022)

TADW: Traceable and Anti-detection Dynamic Watermarking of Deep Neural Networks

Jinwei Dong , He Wang, Zhipeng He, Jun Niu, Xiaoyan Zhu, and Gaofei Wu 

Research Article (11 pages), Article ID 9505808, Volume 2022 (2022)

Design and Analysis of a Provable Secure Two-Factor Authentication Protocol for Internet of Things

Chien-Ming Chen , Shuangshuang Liu , Xuanang Li , Saru Kumari , and Long Li 

Research Article (12 pages), Article ID 4468301, Volume 2022 (2022)

Research Article

Dual-Server Identity-Based Encryption with Authorized Equality Test for IoT Data in Clouds

Meng Zhao¹ and Yong Ding ^{1,2}

¹Guangxi Key Laboratory of Cryptography and Information Security, School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, China

²Cyberspace Security Research Center, Pengcheng Laboratory, Shenzhen, China

Correspondence should be addressed to Yong Ding; stone_dingy@126.com

Received 9 May 2022; Revised 20 August 2022; Accepted 6 September 2022; Published 11 October 2022

Academic Editor: AnMin Fu

Copyright © 2022 Meng Zhao and Yong Ding. 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 massive amounts of data collected by Internet of things (IoT) devices can be stored in clouds to solve the problem of the low storage capacity of IoT terminals. However, the privacy and security of outsourced IoT data may be compromised on the cloud side. Traditional cryptographic technologies can protect data privacy but require the user to retrieve the data for decryption and further processing, which would bring vast amounts of bandwidth and computation burden to users. This paper proposes a dual-server identity-based encryption scheme supporting authorized ciphertext equality test (DS-IBE-AET), where two noncolluding servers with authorizations from users can collaboratively carry out an equality test on outsourced IoT ciphertexts without decrypting the data. DS-IBE-AET can resist offline keyword guessing attacks confronted by existing encryption schemes with equality test in the single server model. Security analysis demonstrates that the proposed DS-IBE-AET scheme offers unforgeability for private keys of users and servers and confidentiality protection for outsourced IoT data and authentication tokens. The performance analysis indicates the practicality of our DS-IBE-AET construction for securing outsourced IoT data in clouds.

1. Introduction

With the advancement of cloud computing, various types of user data produced in the Internet of things, Internet of vehicles, smart grid and other applications can be maintained in the cloud to reduce the local storage costs. In order to protect data privacy on cloud servers, the most common and effective method is to encrypt data, then upload encrypted data to servers. Traditional data encryption technologies can ensure data confidentiality; however, they would make encrypted data unsearchable and incomparable [1, 2]. Thus, users have to retrieve the data from remote cloud servers, then decrypt it for processing. This will not be able to take advantage of the powerful computing resources of the cloud server and bring huge computing overhead to users.

To solve this problem, Boneh et al. [3] proposed a public key encryption scheme with keyword search (PEKS), where the keyword is encrypted and outsourced along with the

encrypted message so that it can be compared with the encrypted trapdoor for realizing privacy-preserving search over the outsourced data. Particularly, the search process relies on the equality test between the encrypted keyword and trapdoor. In 2010, Yang et al. [4] presented a public key encryption scheme with equality test (PKEET), which allowed the cloud server to check whether two ciphertexts had the same plaintext without decryption. Here, these ciphertexts may be generated by different users with different public keys. Since then, many variants supporting ciphertext equality tests with different functions and characteristics have been introduced [5–8].

However, most of these schemes supporting equality tests on outsourced ciphertexts are proposed in the single server model, which cannot resist offline keyword guessing attacks. That is, the cloud server is able to generate ciphertext for any message in the message space in the public key setting, then after being authorized, it can perform the

equality test procedure with outsourced ciphertexts. In this way, the cloud server would find all the ciphertexts that encrypted the chosen message through the equality test procedure. Therefore, the confidentiality of these outsourced ciphertexts is compromised. To address this issue, Zhao et al. [9] proposed a public key encryption scheme with authorized equality test in the dual-server model, where the outsourced data is only stored at the primary server and two servers would not collude to launch attacks against user data. However, since their scheme is designed for as public key setting, they confront complex certificate management problems. Also, Wu et al. [10] designed an identity-based scheme supporting equality test in a dual-server model, while the privacy of the authentication token was not considered.

1.1. Our Contributions. In this paper, we propose a dual-server identity-based encryption scheme supporting authorized equality tests on outsourced IoT ciphertexts (DS-IBE-AET). As in the dual-server model of [9, 10], the front server and back server would not launch collusion attacks to compromise the confidentiality of outsourced IoT data, where these data are only kept at the front server. The equality test procedures can be executed in sequence only after both servers have obtained user authorizations, which can also be conducted in a multiuser setting with the authorizations from different users. The back server can only get internal test results from the front server, which makes it impossible to deduce the information from user data.

In our DS-IBE-AET construction, the encrypted authorization tokens for two servers are in the same format, which should be decrypted using the respective secret key for performing equality test procedures. Compared to [9, 10], our DS-IBE-AET construction designed for the identity-based setting avoids the burden of certificate management. Security analysis demonstrates that the proposed DS-IBE-AET construction guarantees the unforgeability of users' and servers' private keys, the privacy of outsourced data against two servers, as well as the privacy of authorization tokens. The performance analysis indicates that the proposed DS-IBE-AET construction is practical in IoT-related applications.

1.2. Related Works. Public key encryption with equality test is closely related to PEKS. Boneh et al. [3] introduced PEKS to allow the e-mail gateway to test whether the e-mail contained some special keywords, where the gateway did not need to decrypt emails. The main idea behind PEKS is to test the equality of the encrypted keywords and trapdoor. PKEET was first introduced by Yang et al. [4], which allows any entity to perform an equality test on two ciphertexts to determine whether they were generated by the same plaintext, where the ciphertexts may be produced with different public keys. The ciphertext equality test technology has been extensively used in different scenarios, for example, privacy-preserving equi-join in relational databases [7, 11], secure deduplication on cloud data [12, 13], implicit authentication [14], and privacy-preserving road condition monitoring [15].

Since the outsourced ciphertexts can be publicly compared in Yang et al.'s PKEET [4], many solutions supporting authentication mechanisms have been developed. Tang introduced the AoN-PKEET [16] and FG-PKEET [17] to realize coarse-grained and fine-grained authorization for ciphertext equality tests, respectively. Wang et al. [18] presented a public key signcryption scheme with designated equality test to secure messaging services. Lee et al. [19] presented a generic PKEET construction by employing a two-level hierarchical identity-based encryption scheme, a strongly unforgeable one-time signature scheme, and a cryptographic hash function, whose security can be proved in the standard model. Attribute-based and proxy encryption schemes supporting authorized equality testing on ciphertexts had been studied in [20, 21], respectively. Compared with our DS-IBE-AET construction in the dual-server model, these schemes were designed in a public key setting, which faced the complex certificate management problem and cannot resist offline keyword guessing attacks.

Many identity-based encryption schemes (IBE) with ciphertext equality tests (IBEET) have been presented, which can mitigate the complexity of public key certificate management in PKEET. Ma [22] first introduced IBEET by combining PKEET and IBE. Wu et al. [23] put forward a novel IBEET scheme, in which users are divided into different groups, and only the users in the same group can generate ciphertexts with the shared secret token. In [24], Lee et al. noted that Wu et al.'s scheme [23] cannot resist insider attack and presented an improved IBEET construction. Alornyo et al. [25] constructed an IBEET from witness-based encryption technology to resist insider attacks, which offered weak indistinguishability under chosen ciphertext attacks in the random oracle model. Ling et al. [26] introduced group IBEET, where only the group administrator was able to issue authorization tokens to the tester. Compared with our DS-IBE-AET construction in the dual-server model, these schemes engage only a single cloud server, which cannot resist offline keyword guessing attacks.

The dual-server model has been generally employed in designing secure and privacy-preserving systems for resisting keyword guessing attacks launched by cloud servers, where two semitrusted servers would not collude with each other [27]. In [28], Tang introduced an amended FG-PKEET scheme in the two-proxy setting, where the equality test procedure had to be interactively carried out by two proxies. In this way, the ciphertexts can be protected against offline message recovery attacks. Wu et al. [10] proposed a dual-server identity-based encryption with equality test for mobile health social networks, in which two servers with authentication tokens can collaborate to complete the ciphertext equality test, while the privacy of the authentication token was not considered. Recently, Zhao et al. [9] proposed a public key encryption construction supporting authorized equality test on outsourced IoT data in a non-colluding dual-server model. Compared with [9], our DS-IBE-AET construction is developed in an identity-based setting, which can avoid the complex certificate management problem.

1.3. Paper Organization. The remainder of this paper is structured as follows. Section 2 reviews the preliminaries. The system model, security requirements, and system framework are introduced in Section 3. A concrete DS-IBE-AET scheme is presented in Section 4, while security and performance are analyzed in Section 5. Finally, the paper is concluded in Section 6.

2. Preliminaries

2.1. Bilinear Groups. Suppose $\mathbb{G} = \langle g \rangle$ and \mathbb{G}_T are two cyclic groups of prime order q . The mapping $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear pairing if the following conditions are satisfied:

2.1.1. Bilinearity. For any $g_1, g_2 \in_R \mathbb{G}$ and $\alpha, \beta \in_R \mathbb{Z}_q^*$,

$$\hat{e}(g_1^\alpha, g_2^\beta) = \hat{e}(g_1, g_2)^{\alpha\beta}. \quad (1)$$

2.1.2. Non-Degeneracy. There exists $g_1, g_2 \in \mathbb{G}$ such that,

$$\hat{e}(g_1, g_2) \neq 1. \quad (2)$$

2.1.3. Efficiency. For $g_1, g_2 \in_R \mathbb{G}$, there exists an efficient algorithm to compute $\hat{e}(g_1, g_2)$.

2.2. Complexity Assumptions. The security of our DS-IBE-AET construction relies on the following complexity assumptions.

CDH assumption. Suppose $\mathbb{G} = \langle g \rangle$ is a cyclic group of prime order q . Given a tuple (g, g^a, g^b) where $a, b \in_R \mathbb{Z}_q^*$, no probabilistic polynomial-time algorithm \mathcal{A} can compute g^{ab} with nonnegligible probability.

CBDH assumption. Suppose $\mathbb{G} = \langle g \rangle$ and \mathbb{G}_T are two cyclic groups of prime order q and satisfy bilinear pairing $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Given a tuple (g, g^a, g^b, g^c) where $a, b, c \in_R \mathbb{Z}_q^*$, no probabilistic polynomial-time algorithm \mathcal{A} can compute $\hat{e}(g, g)^{abc}$ with nonnegligible probability.

3. System Model and Security Requirements

3.1. System Model. As shown in Figure 1, in a DS-IBE-AET system, there are three types of entities, namely, a key generation center (KGC), users, and servers. KGC is an honest entity that is responsible for initializing the DS-IBE-AET system by producing the master private key and public parameters. It also issues the private keys for all users, the front server S_f and back server S_b according to their identities, respectively.

In the DS-IBE-AET system, both the data sender and the data recipient are system users. The data sender encrypts the data using the identity of the data recipient and the system public parameters, and the generated ciphertexts are only sent to the front server S_f for storage. The data recipient is able to retrieve ciphertexts from the front server S_f and run the decryption procedure using his/her private key. Also, a data recipient is able to authorize the front server S_f and back server S_b to perform equality test on his/her ciphertexts without decryption. The authorization tokens are encrypted

using the identities of two servers, so that they can only be decrypted by the two servers.

The front server S_f has huge storage resources for maintaining user data in ciphertext format. Both the front server S_f and back server S_b have powerful computing capabilities for collaboratively performing equality tests on user ciphertexts after being authorized. With the authorizations from users, the front server S_f is able to generate internal results of equality tests on ciphertexts, which are then sent to the back server S_b to further confirm whether two ciphertexts encrypt the same message. The authorization tokens only allow two servers to collaboratively perform equality tests on users' ciphertexts.

3.2. Security Requirements. In the DS-IBE-AET system, the front server and back server would not launch collusion attacks to compromise the privacy of user ciphertexts. A secure DS-IBE-AET system must meet the following security conditions:

3.2.1. Unforgeability of User Private Key. The private key generated by KGC for user cannot be forged by any entity.

3.2.2. Unforgeability of Server Private Key. The private keys generated by KGC for the front server S_f and back server S_b cannot be forged by any entity.

3.2.3. Data Privacy against the front Server. The front server S_f cannot deduce the private information of users from the stored ciphertexts before and after being authorized by users to perform equality tests.

3.2.4. Data Privacy against the Back Server. After obtaining the users' authorizations for performing the equality test, the back server S_b cannot deduce the private information of users from the received internal results.

3.2.5. Privacy Protection on Authentication Token. The authentication tokens generated for the front server S_f and back server S_b can only be decrypted by themselves, respectively.

3.3. System Framework. A DS-IBE-AET scheme consists of nine polynomial-time procedures, namely, Setup, UKeyExt, SKeyExt, Encrypt, Decrypt, Authen, DecAuth, EqTest_f, and EqTest_b.

3.3.1. Setup. On input of the security parameter δ , the system setup procedure, which is run by KGC, generates the system master private key mpk and system public parameters $param$. We denote $(mpk, param) \leftarrow \text{Setup}(1^\delta)$. Note that $param$ is the implicit input for the following eight procedures.

3.3.2. UKeyExt. On input of the master private key mpk and a user identity ID_i , the user key extraction procedure, which is run by KGC, generates a private key usk_i for user ID_i . We denote $usk_i \leftarrow \text{UKeyExt}(mpk, ID_i)$.

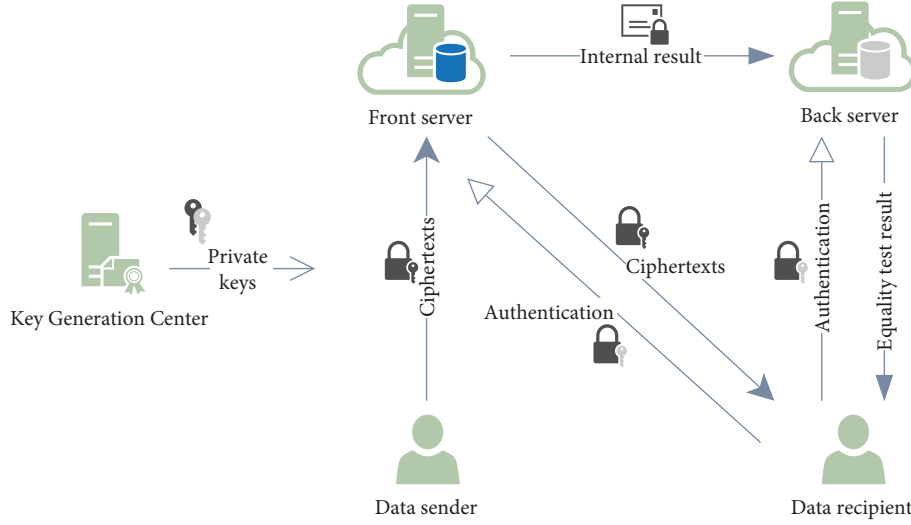


FIGURE 1: A system model of DS-IBE-AET.

3.3.3. *SKeyExt*. On input of the master private key mpk and the identity S_f of the front server (resp. S_b of the back server), the server key extraction procedure, which is run by KGC, generates a private key ssk_f for the front server S_f (resp. ssk_b for the back server S_b). We denote $ssk_f/ssk_b \leftarrow SKeyExt(mpk, S_f/S_b)$.

3.3.4. *Encrypt*. On input of the identity ID_i of the data recipient and a message m , the data encryption procedure, which is performed by the data sender, generates a ciphertext c and sends it to the front server S_f . We denote $c \leftarrow Encrypt(ID_i, m)$.

3.3.5. *Decrypt*. On input of the private key usk_i of the data recipient ID_i and a ciphertext c , the data decryption procedure, which is performed by data recipient, outputs a plaintext m or \perp that signifies an error in decryption. We denote $m/\perp \leftarrow Decrypt(usk_i, c)$.

3.3.6. *Authen*. On input of the private key usk_i of user ID_i and the identities (S_f, S_b) of the front server and back server, the authentication token generation procedure, which is carried out by the user ID_i , generates ciphertext authentication tokens $\hat{t}_{i,f}$ and $\hat{t}_{i,b}$ for two servers. Note that the tokens $\hat{t}_{i,f}$ and $\hat{t}_{i,b}$ are sent to the front server S_f and back server S_b , respectively. We denote $(\hat{t}_{i,f}, \hat{t}_{i,b}) \leftarrow Authen(usk_i, S_f, S_b)$.

3.3.7. *DecAuth*. On input of the private key ssk_f of the front server S_f (resp. ssk_b of the back server S_b) and a ciphertext authentication token $\hat{t}_{i,f}$ (resp. $\hat{t}_{i,b}$), the authentication decryption procedure, which is performed by the front server S_f (resp. the back server S_b), outputs a plaintext authentication token $\tau_{i,f}$ (resp. $\tau_{i,b}$) or \perp that signifies an error in decryption. We denote $\tau_{i,f}/\perp \leftarrow DecAuth(ssk_f, \hat{t}_{i,f})$ for the front server S_f and $\tau_{i,b}/\perp \leftarrow DecAuth(ssk_b, \hat{t}_{i,b})$ for the back server S_b .

3.3.8. *EqTest_f*. On input of the plaintext authentication tokens $\tau_{i,f}$ and $\tau_{j,f}$ of two users ID_i and ID_j , respectively, and their ciphertexts c and c' , the front equality test procedure, which is performed by the front server S_f , outputs an internal result Γ and sends it to the back server S_b . We denote $\Gamma \leftarrow EqTest_f(\tau_{i,f}, \tau_{j,f}, c, c')$.

3.3.9. *EqTest_b*. On input of the plaintext authentication tokens $\tau_{i,b}$ and $\tau_{j,b}$ of two users ID_i and ID_j , respectively, and an internal result Γ , the back equality test procedure, which is performed by the back server S_b , outputs 1 if c and c' encrypt the same message or 0 otherwise. We denote $1/0 \leftarrow EqTest_b(\tau_{i,b}, \tau_{j,b}, \Gamma)$.

A DS-IBE-AET construction must be sound in the sense that if the procedures are performed honestly, the following conditions hold:

- (i) The private key extracted by KGC for some users can be validated by such a user.
- (ii) The private key extracted by KGC for each server can be validated by such a server.
- (iii) The ciphertext generated by the data encryption procedure can be decrypted by the data decryption procedure.
- (iv) The ciphertext authentication token generated by the authentication token generation procedure can be decrypted by the authentication decryption procedure.
- (v) For any two ciphertexts that encrypt the same message, which may belong to different users, the front and back equality test procedures can collaboratively output 1.
- (vi) For any two ciphertexts that encrypt different messages, which may belong to different users, the front and back equality test procedures collaboratively output 0 with overwhelming probability.

Definition 1. (Soundness): A DS-IBE-AET construction is sound if, for any security parameter δ , any master private key and public parameters $(mpk, \text{param}) \leftarrow \text{Setup}(1^\delta)$, any private keys $usk_i \leftarrow \text{UKeyExt}(mpk, ID_i)$, $usk_j \leftarrow \text{UKeyExt}(mpk, ID_j)$ of two users ID_i and ID_j , any private key of the front server $ssk_f \leftarrow \text{SKeyExt}(mpk, S_f)$, and any private key of the back server $ssk_b \leftarrow \text{SKeyExt}(mpk, S_b)$, the following conditions are satisfied:

- (i) The private key usk_i can be verified as valid in the verification step by the user ID_i .
- (ii) The private key ssk_f can be verified as valid in the verification step by the front server S_f , and the private key ssk_b can be verified as valid in the verification step by the back server S_b .
- (iii) For any message m , $\text{Decrypt}(usk_i, \text{Encrypt}(ID_i, m)) = m$.
- (iv) $\text{DecAuth}(ssk_f, \hat{t}_{i,f}) = \tau_{i,f}$ and $\text{DecAuth}(ssk_b, \hat{t}_{i,b}) = \tau_{i,b}$, where $(\hat{t}_{i,f}, \hat{t}_{i,b}) \leftarrow \text{Authen}(usk_i, S_f, S_b)$.
- (v) For any two messages m, m' such that $c \leftarrow \text{Encrypt}(ID_i, m)$ and $c' \leftarrow \text{Encrypt}(ID_i, m')$, if $m = m'$, then $\text{EqTest}_b(\tau_{i,b}, \tau_{j,b}, \Gamma) = 1$, otherwise $\Pr[\text{EqTest}_b(\tau_{i,b}, \tau_{j,b}, \Gamma) = 0] \geq 1 - \varepsilon(\cdot)$, where $\Gamma \leftarrow \text{EqTest}_f(\tau_{i,f}, \tau_{j,f}, c, c')$, $\tau_{i,f} = \text{DecAuth}(ssk_f, \hat{t}_{i,f})$, $\tau_{i,b} = \text{DecAuth}(ssk_b, \hat{t}_{i,b})$, $\tau_{j,f} = \text{DecAuth}(ssk_f, \hat{t}_{j,f})$, $\tau_{j,b} = \text{DecAuth}(ssk_b, \hat{t}_{j,b})$, $(\hat{t}_{i,f}, \hat{t}_{i,b}) \leftarrow \text{Authen}(usk_i, S_f, S_b)$, $(\hat{t}_{j,f}, \hat{t}_{j,b}) \leftarrow \text{Authen}(usk_j, S_f, S_b)$, and $\varepsilon(\cdot)$ represents a negligible function.

4. Concrete DS-IBE-AET Construction

This section presents a concrete DS-IBE-AET construction in bilinear groups, where a running process is shown in Figure 2, and the frequently used symbols are summarized in Table 1.

4.1. System Setup. Given a security parameter δ , KGC chooses cyclic groups $\mathbb{G} = \langle g \rangle$ and \mathbb{G}_T satisfying bilinear mapping $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where groups \mathbb{G} and \mathbb{G}_T have prime order q . KGC selects four cryptographic hash functions $H_1: \{0, 1\}^* \rightarrow \mathbb{G}$, $H_2: \{0, 1\}^{\xi_m} \rightarrow \mathbb{G}$, $H_3: \mathbb{G}_T \rightarrow \mathbb{G}$ and $H_4: \mathbb{G}_T \times \mathbb{G} \rightarrow \{0, 1\}^{\xi_{\mathbb{G}} + \log_q}$, where $\xi_{\mathbb{G}}$ and ξ_m , respectively, denote the element size in group \mathbb{G} and message space. Also, KGC picks three random elements $d_1, d_2, d_3 \in \mathbb{Z}_q^*$ and computes the following:

$$\begin{aligned} V_1 &= g^{d_1}, \\ V_2 &= g^{d_2}, \\ V_3 &= g^{d_3}. \end{aligned} \quad (3)$$

At last, KGC keeps the master private key $mpk = (d_1, d_2, d_3)$ secret and publishes the public parameter $\text{param} = (\delta, \mathbb{G}, \mathbb{G}_T, q, \hat{e}, g, H_1, H_2, H_3, H_4, V_1, V_2, V_3)$.

4.2. User Key Extraction. Given the identity of user ID_i , KGC generates the private key $usk_i = (usk_{i,1}, usk_{i,2}, usk_{i,3})$ as follows:

$$\begin{aligned} usk_{i,1} &= H_1(ID_i)^{d_1}, \\ usk_{i,2} &= H_1(ID_i)^{d_2}, \\ usk_{i,3} &= H_1(ID_i)^{d_3}. \end{aligned} \quad (4)$$

The private key usk_i is sent to the user ID_i via secure channel. Note that the user ID_i is able to validate usk_i as follows:

$$\hat{e}(usk_{i,1}, g) = \hat{e}(H_1(ID_i), V_1), \quad (5)$$

$$\hat{e}(usk_{i,2}, g) = \hat{e}(H_1(ID_i), V_2), \quad (6)$$

$$\hat{e}(usk_{i,3}, g) = \hat{e}(H_1(ID_i), V_3). \quad (7)$$

4.3. Server Key Extraction. Given the identity of the front server S_f , KGC generates the private key as follows:

$$ssk_f = H_1(S_f)^{d_1}, \quad (8)$$

which is sent to the front server S_f via secure channel. Note that the front server S_f is able to validate ssk_f as follows:

$$\hat{e}(ssk_f, g) = \hat{e}(H_1(S_f), V_1). \quad (9)$$

Similarly, KGC can generate the private key for the back server S_b as follows:

$$ssk_b = H_1(S_b)^{d_1}, \quad (10)$$

and the back server S_b is able to validate ssk_b as follows:

$$\hat{e}(ssk_b, g) = \hat{e}(H_1(S_b), V_1). \quad (11)$$

4.4. Data Encryption. For a message $m \in \{0, 1\}^{\xi_m}$, the sender randomly picks an element $\alpha \in \mathbb{Z}_q^*$, and computes the ciphertext $c = (c_1, c_2, c_3)$, where

$$\begin{aligned} c_1 &= g^\alpha, \\ c_2 &= H_2(m) \cdot H_3(\hat{e}(H_1(ID_i), V_1)^\alpha) \\ &\quad \cdot H_3(\hat{e}(H_1(ID_i), V_2)^\alpha), \\ c_3 &= (m \parallel \alpha) \oplus H_4(\hat{e}(H_1(ID_i), V_3)^\alpha \parallel H_2(m)). \end{aligned} \quad (12)$$

The ciphertext c is sent to the front server S_f .

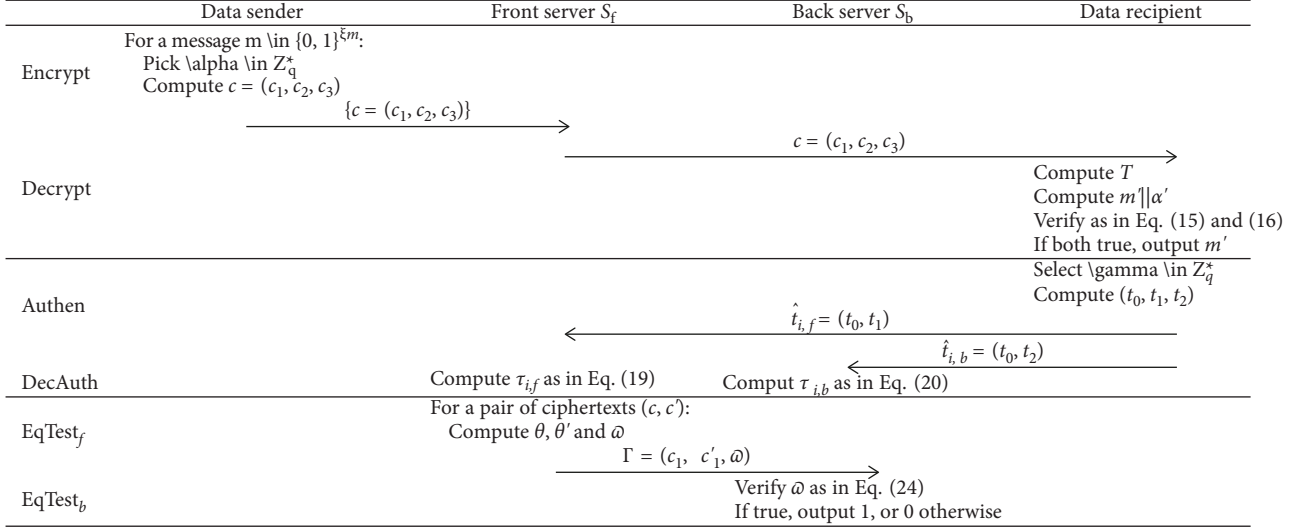


FIGURE 2: A running process of the proposed DS-IBE-AET construction.

TABLE 1: Notations.

Symbol	Meaning
δ	Security parameter
\mathbb{G}, \mathbb{G}_T	Cyclic groups of prime order q satisfying bilinear pairing $\hat{e}: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$
H_1, H_2, H_3, H_4	Cryptographic hash functions
g	A generator of \mathbb{G}
$mpk = (d_1, d_2, d_3)$	The master private key
$param$	The public parameter
$usk_i = (usk_{i,1}, usk_{i,2}, usk_{i,3})$	Private key of user ID_i
ssk_f, ssk_b	Private keys of S_f and S_b
$c = (c_1, c_2, c_3)$	Ciphertext of message m
$\hat{t}_{i,f} = (t_0, t_1), \hat{t}_{i,b} = (t_0, t_2)$	Ciphertext authentication tokens of user ID_i for S_f and S_b
$\tau_{i,f}, \tau_{i,b}$	Plaintext authentication tokens of user ID_i for S_f and S_b
$\Gamma = (c_1, c'_1, \omega)$	Internal test result of equality test

4.5. *Data Decryption.* For ciphertext $c = (c_1, c_2, c_3)$, the user ID_i decrypts it with the private key usk_i as follows. The user ID_i computes the following:

$$T = \frac{c_2}{H_3(\hat{e}(usk_{i,1}, c_1)) \cdot H_3(\hat{e}(usk_{i,2}, c_1))}, \quad (13)$$

$$m' \parallel \alpha' \leftarrow c_3 \oplus H_4(\hat{e}(usk_{i,3}, c_1) \parallel T). \quad (14)$$

Next, the user ID_i checks whether both of the following equalities hold:

$$c_1 = g^{\alpha'}, \quad (15)$$

$$T = H_2(m'). \quad (16)$$

If so, m' is outputted, otherwise \perp is outputted.

4.6. *Authorization.* The user ID_i randomly picks an element $\gamma \in \mathbb{Z}_q^*$ and computes the following:

$$t_0 = g^\gamma t_1 = usk_{i,1} \cdot H_3(\hat{e}(H_1(S_f)), V_1)^\gamma, \quad (17)$$

$$t_1 = usk_{i,1} \cdot H_3(\hat{e}(H_1(S_f)), V_1)^\gamma. \quad (18)$$

Then, the encrypted authorization tokens $\hat{t}_{i,f} = (t_0, t_1)$ and $\hat{t}_{i,b} = (t_0, t_2)$ are sent to the front server S_f and to the back server S_b , respectively.

4.7. *Token Decryption.* Given the encrypted authorization token $\hat{t}_{i,f}$, the front server S_f computes the following equation:

$$\tau_{i,f} = \frac{t_1}{H_3(\hat{e}(ssk_f, t_0))}. \quad (19)$$

The back server S_b can decrypt the token $usk_{i,b}$ in the similar way as follows:

$$\tau_{i,b} = \frac{t_2}{H_3(\hat{e}(ssk_b, t_0))}. \quad (20)$$

then, these two servers are able to validate the recovered tokens as in (5) and (6), respectively.

4.8. Front Server Ciphertext Test. For the ciphertexts $c = (c_1, c_2, c_3)$ and $c' = (c'_1, c'_2, c'_3)$ of two users ID_i and ID_j , respectively, the front server S_f generates the internal test result $\Gamma = (c_1, c'_1, \omega)$ with their respective tokens $\tau_{i,f}$ and $\tau_{j,f}$. The front server S_f computes the following equation:

$$\theta = \frac{c_2}{H_3(\tilde{e}(\tau_{i,f}, c_1))}, \quad (21)$$

$$\theta' = \frac{c'_2}{H_3(\tilde{e}(\tau_{j,f}, c'_1))}, \quad (22)$$

then, it computes

$$\omega = \frac{\theta}{\theta'}, \quad (23)$$

the internal test result $\Gamma = (c_1, c'_1, \omega)$ is sent to the back server S_b .

4.9. Back Server Ciphertext Test. For the internal test result $\Gamma = (c_1, c'_1, \omega)$ on the ciphertexts of users ID_i and ID_j , the back server S_b checks the following equality with the received tokens $\tau_{i,b}$ and $\tau_{j,b}$:

$$\omega = \frac{H_3(\tilde{e}(\tau_{i,b}, c_1))}{H_3(\tilde{e}(\tau_{j,b}, c'_1))}, \quad (24)$$

if it holds, then "1" is outputted, which means the two ciphertexts c and c' of users ID_i and ID_j encrypt the same message; otherwise "0" is outputted, which means different messages are encrypted in two ciphertexts.

Theorem 1. *The proposed DS-IBE-AET construction in bilinear groups is sound.*

Proof. 1 First, for the first element $usk_{i,1}$ in the private key usk_i of user ID_i , the equality in (1) holds as follows:

$$\tilde{e}(usk_{i,1}, g) = \tilde{e}(H_1(ID_i)^{d_1}, g) = \tilde{e}(H_1(ID_i), V_1). \quad (25)$$

The equalities in (6) and (7) for the other two elements $usk_{i,2}$ and $usk_{i,3}$ can be verified in the similar way.

Second, for the private key ssk_f for the front server S_f , the equality in (9) holds as follows:

$$\tilde{e}(ssk_f, g) = \tilde{e}(H_1(ID_i)^{d_1}, g) = \tilde{e}(H_1(ID_i), V_1). \quad (26)$$

The equality in (11) for the back server S_b can be verified in the similar way.

Third, for the correctness of decryption on user ciphertexts, since

$$\begin{aligned} T &= \frac{c_2}{H_3(\tilde{e}(usk_{i,1}, c_1)) \cdot H_3(\tilde{e}(usk_{i,2}, c_1))} \\ &= \frac{H_2(m) \cdot H_3(\tilde{e}(H_1(ID_i), V_1)^\alpha) \cdot H_3(\tilde{e}(H_1(ID_i), V_2)^\alpha)}{H_3(\tilde{e}(H_1(ID_i)^{d_1}, g^\alpha)) \cdot H_3(\tilde{e}(H_1(ID_i)^{d_2}, g^\alpha))} \\ &= \frac{H_2(m) \cdot H_3(\tilde{e}(H_1(ID_i), V_1)^\alpha) \cdot \tilde{e}(H_1(ID_i), V_2)^\alpha}{H_3(\tilde{e}(H_1(ID_i)^{d_1}, g^\alpha)) \cdot H_3(\tilde{e}(H_1(ID_i)^{d_2}, g^\alpha))} \\ &= H_2(m). \end{aligned} \quad (27)$$

we have

$$\begin{aligned} m' \|\alpha' &= c_3 \oplus H_4(\tilde{e}(usk_{i,3}, c_1) \| T) \\ &= (m \|\alpha) \oplus H_4(\tilde{e}(H_1(ID_i), V_3)^\alpha \\ &\| H_2(m)) \oplus H_4(\tilde{e}(H_1(ID_i)^{d_3}, g^\alpha) \| H_2(m)) b \\ &= (m \|\alpha) \oplus H_4(\tilde{e}(H_1(ID_i), g^{d_3})^\alpha \| H_2(m)) \\ &\oplus H_4(\tilde{e}(H_1(ID_i)^{d_3}, g^\alpha) \| H_2(m)) \\ &= (m \|\alpha), \end{aligned} \quad (28)$$

thus, the equalities (15) and (16) hold, which means the message m can be successfully decrypted.

Four, for the authorization token decryption, it can be seen that

$$\begin{aligned} \tau_{i,f} &= \frac{t_1}{H_3(\tilde{e}(ssk_f, t_0))} \\ &= \frac{usk_{i,1} \cdot H_3(\tilde{e}(H_1(S_f), V_1)^y)}{H_3(\tilde{e}(H_1(S_f)^{d_1}, g^y))} \\ &= \frac{usk_{i,1} \cdot H_3(\tilde{e}(H_1(S_f), g^{d_1})^y)}{H_3(\tilde{e}(H_1(S_f)^{d_1}, g^y))} \\ &= usk_{i,1}. \end{aligned} \quad (29)$$

$$\begin{aligned} \tau_{i,b} &= \frac{t_2}{H_3(\tilde{e}(ssk_b, t_0))} \\ &= \frac{usk_{i,b} \cdot H_3(\tilde{e}(H_1(S_b), V_1)^y)}{H_3(\tilde{e}(H_1(S_b)^{d_1}, g^y))} \\ &= \frac{usk_{i,b} \cdot H_3(\tilde{e}(H_1(S_b), g^{d_1})^y)}{H_3(\tilde{e}(H_1(S_b)^{d_1}, g^y))} \\ &= usk_{i,2}. \end{aligned}$$

Thus, the tokens for the front server S_f and back server S_b can be correctly decrypted as in (19) and (20).

Five, for an authorized equality test on ciphertexts, since

$$\begin{aligned}
\theta &= \frac{c_3}{H_3(\tilde{e}(\tau_{i,1}, c_1))} \\
&= \frac{H_2(m) \cdot H_3(\tilde{e}(H_1(ID_i), V_1)^\alpha) \cdot H_3(\tilde{e}(H_1(ID_i), V_2)^\alpha)}{H_3\tilde{e}((H_1(ID_i)^{d_1}, g^\alpha))} \\
&= \frac{H_2(m) \cdot H_3(\tilde{e}(H_1(ID_i), g^{d_1})^\alpha) \cdot H_3(\tilde{e}(H_1(ID_i), V_2)^\alpha)}{H_3\tilde{e}((H_1(ID_i)^{d_1}, g^\alpha))} \\
&= H_2(m) \cdot H_3(\tilde{e}(H_1(ID_i), V_2)^\alpha), \\
\theta r &= \frac{c_2'}{H_3(\tilde{e}(\tau_{j,1}, c_1'))} \\
&= \frac{H_2(m') \cdot H_3(\tilde{e}(H_1(ID_j), V_1)^{\alpha'}) \cdot H_3(\tilde{e}(H_1(ID_j), V_2)^{\alpha'})}{H_3\tilde{e}((H_1(ID_j)^{d_1}, g^{\alpha'}))} \\
&= \frac{H_2(m') \cdot H_3(\tilde{e}(H_1(ID_j), g^{d_1})^{\alpha'}) \cdot H_3(\tilde{e}(H_1(ID_j), V_2)^{\alpha'})}{H_3\tilde{e}((H_1(ID_j)^{d_1}, g^{\alpha'}))} = H_2(m') \cdot H_3(\tilde{e}(H_1(ID_j), V_2)^{\alpha'}). \tag{30}
\end{aligned}$$

we have

$$\begin{aligned}
\omega &= \frac{\theta}{\theta'} \\
&= \frac{H_2(m) \cdot H_3(\tilde{e}(H_1(ID_i), V_2)^\alpha)}{H_2(m') \cdot H_3(\tilde{e}(H_1(ID_j), V_2)^{\alpha'})} \\
&= \frac{H_2(m) \cdot H_3(\tilde{e}(H_1(ID_i), g^{d_2})^\alpha)}{H_2(m') \cdot H_3(\tilde{e}(H_1(ID_j), g^{d_2})^{\alpha'})} \\
&= \frac{H_2(m) \cdot H_3(\tilde{e}(H_1(ID_i)^{d_2}, g^\alpha))}{H_2(m') \cdot H_3(\tilde{e}(H_1(ID_j)^{d_2}, g^{\alpha'}))} \tag{31} \\
&= \frac{H_2(m) \cdot H_3(\tilde{e}(usk_{i,2}, c_1))}{H_2(m') \cdot H_3(\tilde{e}(usk_{j,2}, c_1'))} \\
&\stackrel{m=m'}{\Leftrightarrow} \frac{H_3(\tilde{e}(usk_{i,2}, c_1))}{H_3(\tilde{e}(usk_{j,2}, c_1'))}.
\end{aligned}$$

It can be seen that if the messages m and m' are the same, then the equality in (24) holds.

Therefore, the proposed DS-IBE-AET construction in bilinear groups is sound. \square

5. Analysis and Comparison

5.1. Security Analysis

Theorem 2. *The proposed DS-IBE-AET construction in the dual-server model can guarantee the unforgeability of the private keys of users.*

Proof. 2 As shown in Sections 4.2, the private keys of users are generated by KGC using their private keys (d_1, d_2, d_3) . Particularly, each element of the private key is a signature on the user's identity with the short signature scheme of Boneh et al [29]. Therefore, according to the security result that the BLS signature scheme is secure against existential forgery under adaptive chosen-message attacks in the random oracle model assuming the CDH assumption holds [29], Theorem 3.2, the proposed DS-IBE-AET scheme can protect the unforgeability of private keys of users. \square

Theorem 3. *The proposed DS-IBE-AET construction in the dual-server model can guarantee the unforgeability of the private keys of both servers.*

Proof. 3 Similar to the analysis for Theorem 2, the private keys of both servers are generated by KGC using their private keys d_1 by employing the short signature scheme of Boneh et al [29]. Thus, according to the security result of [29], Theorem 3.2, the proposed DS-IBE-AET scheme can protect the unforgeability of the private keys of two servers. \square

Theorem 4. *The proposed DS-IBE-AET construction in the dual-server model can guarantee the privacy of outsourced data against the front server.*

Proof. 4 As shown in Section 4.4, the ciphertext in the proposed DS-IBE-AET scheme has a similar form as Lee et al.'s PKE-AET scheme [30]. Note that their scheme is designed in generic cyclic groups, and our DS-IBE-AET scheme is developed in bilinear groups in an identity-based setting. Moreover, the pair (c_1, c_2) can be seen as an extension of the ciphertext in Boneh and Franklin's basic IBE scheme [31], Section 4. For the second component c_2 of our ciphertext, two public parameters V_1 and V_2 are used, which would be used to enable both the front server and back server to collaboratively perform equality tests on ciphertexts; while only one public key g^α is used in producing c_2 in Lee et al.'s PKE-AET scheme [30], since their scheme is considered in the single server model. Therefore, before the front server gets authorized, the proof for the privacy of outsourced data follows [30], Theorem 4.1 and [31], Theorem 4.1, that is, the proposed DS-IBE-AET scheme offers indistinguishability against chosen ciphertext and chosen identity attacks (IND-ID-CCA security) for the front server under the CDH and CBDH assumptions. When the front server is authorized, it would get the authorization token $\tau_{i,1}$ for performing an equality test on ciphertexts. Thus, the

proposed DS-IBE-AET scheme offers one-wayness security against chosen ciphertext attacks and chosen identity attacks [31, 32] under the CDH and CBDH assumptions. \square

Theorem 5. *The proposed DS-IBE-AET construction in the dual-server model can guarantee the privacy of outsourced data against the back server.*

Proof. 5 In the proposed DS-IBE-AET scheme, the outsourced ciphertexts are only stored at the front server. When collaboratively performing equality tests on ciphertexts, only the intermediate result $\Gamma = (c_1, c_1, \omega)$ is given to the back server by the front server. Note that ω is computed from θ and θ' . As shown in equations (12) and (13), θ and θ' have a similar form of c_2 in ciphertext of Lee et al.'s scheme [30], but in an identity-based setting [31]. That is, the pairs (c_1, γ) and (c_1, γ') have a similar form of (c_1, c_2) in Lee et al.'s scheme [30] and Boneh and Franklin's basic IBE scheme [31], Section 4. Thus, the proof is similar to that in [30], Theorem 4.1, and [31], Theorem 4.1, that is, the proposed DS-IBE-AET scheme is IND-ID-CCA secure against the back server under the CDH and CBDH assumptions. \square

Theorem 6. *The proposed DS-IBE-AET construction in the dual-server model can guarantee the privacy of an authentication token.*

Proof. 6 The ciphertext authentication token in the proposed DS-IBE-AET construction is generated in a similar way as the ciphertexts in Boneh and Franklin's basic IBE scheme [31], Section 4. The difference is that t_0 is used to construct two encrypted authorization tokens $\hat{t}_{i,f} = (t_0, t_1)$ and $\hat{t}_{i,b} = (t_0, t_2)$ for two servers, respectively. Thus, the proof is similar to that in [31], Theorem 4.1, that is, the authentication token in the proposed DS-IBE-AET construction enjoys indistinguishability against chosen plaintext and chosen identity attacks under the CBDH assumption. \square

5.2. Performance Analysis. In this section, we analyze the efficiency of our DS-IBE-AET construction in each procedure and compare with Zhao et al.'s construction [9] in terms of resource-intensive operations such as exponentiation, bilinear pairing, and the map-to-point hash function. As shown in Table 2, let ℓ_E denote the evaluation cost of an exponentiation in group \mathbb{G} , ℓ_P represent the evaluation cost of a bilinear pairing $\hat{e}(\cdot, \cdot)$ and ℓ_H signify a map-to-point hash function, respectively.

Since our DS-IBE-AET construction is developed in an identity-based setting, the private keys of users and servers are generated by the trusted KGC, where the computational cost of generating a private key for a user is 3 times the cost for a server. Users and servers only need to verify the correctness of the issued private keys, respectively. Note that these private keys should be delivered via a secure channel, thus the verification process can be omitted by the respective users and servers. While in Zhao et al.'s construction [9], the private keys are produced by respective user and server,

which take 3 and 2 exponentiation operations on the bilinear group \mathbb{G} , respectively.

To facilitate the analysis of the data encryption procedure, the exponentiation operation in group \mathbb{G}_T in both schemes is converted to first computing the exponentiation operation in group \mathbb{G} and then performing the bilinear pairing operation $\hat{e}(\cdot, \cdot)$, which can enable intermediate calculated parameters to be reused and reduce computing costs. Hence, the Encrypt procedure of our DS-IBE-AET scheme takes two less exponentiation operations than that in Zhao et al.'s construction [9] when encrypting a message. Since our DS-IBE-AET scheme is designed in an identity-based setting, it takes one more bilinear pairing and map-to-point hash function evaluation than [9]. For decrypting a ciphertext, although our DS-IBE-AET scheme takes one more bilinear pairing operation than Zhao et al.'s construction [9], it only requires one exponentiation operation, whereas the latter needs to carry out 4 exponentiation operations.

In the authentication phase, our DS-IBE-AET scheme allows the user to generate different tokens for two servers. Note that these tokens have the same form and share one element t_0 . Thus, the computing cost for generating t_0 can be shared by two tokens. Also, the exponentiation operation of V_1^y can be reused in producing both t_1 and t_2 .

As shown in (17) and (18), the generation of t_1 and t_2 , respectively, requires two time-consuming map-to-point hash operations. While in Zhao et al.'s construction [9], two servers shall be authenticated with the same token; that is, the servers would recover the same token with the only difference that their respective private keys would be used during decryption. Moreover, since the authentication token is in fact an element of the user's private key, it can be validated according to the relationship with the corresponding public key.

After being authorized, the front server and back server are able to cooperatively carry out equality tests on outsourced ciphertexts. On both server sides, our DS-IBE-AET scheme is much more efficient than Zhao et al.'s construction [9], where no exponentiation operations are required in our DS-IBE-AET scheme. Specifically, to perform an equality test on one pair of ciphertexts, both servers in Zhao et al.'s construction [9] should take 4 more exponentiation operations than those in our DS-IBE-AET scheme, which is due to the fact that the private keys of these servers should be used in their respective procedures. It can be seen that the computing costs for the equality test in both schemes are linear with the number of compared ciphertexts.

Moreover, we evaluate the performance of our DS-IBE-AET scheme and compare it with Zhao et al.'s construction [9], where the experimental execution times of cryptographic operations in [33] are used. The experiments of [33] were carried out on a platform with a Windows 7 operating system, an Intel I7-4700@3.40 GHz CPU, and 4 GB of memory, where the MIRACL Cryptographic SDK [34] was run with $\log q = 512$. The exact execution times of three resource-intensive cryptographic operations are shown in Table 3.

TABLE 2: Comparison of computing costs.

Procedure		Our DS-IBE-AET construction	Zhao et al.s construction [9]
UKeyExt	KGC	$3 \ell_E + 3 \ell_H$	—
	User	$6 \ell_p + 3 \ell_H$	$3 \ell_E$
SKeyExt	KGC	$1 \ell_E + 1 \ell_H$	—
	Server	$2 \ell_p + 1 \ell_H$	$2 \ell_E$
Encrypt		$2 \ell_E + 3 \ell_p + 4 \ell_H$	$4 \ell_E + 2 \ell_p + 3 \ell_H$
Decrypt		$1 \ell_E + 3 \ell_p + 3 \ell_H$	$4 \ell_E + 2 \ell_p + 3 \ell_H$
Authen		$2 \ell_E + 2 \ell_p + 2 \ell_H$	$2 \ell_E + 1 \ell_p$
DecAuth	Decryption	$1 \ell_p + 1 \ell_H$	$1 \ell_E + 1 \ell_p$
	Verification	$2 \ell_p + 1 \ell_H$	$1 \ell_E$
EqTest _f		$2 \ell_p + 2 \ell_H$	$4 \ell_E + 2 \ell_p + 2 \ell_H$
EqTest _b		$2 \ell_p + 2 \ell_H$	$4 \ell_E + 2 \ell_p + 2 \ell_H$

TABLE 3: Execution time of cryptographic operations.

Cryptographic operation	Computing time (ms)
Bilinear pairing	4.211
Exponentiation in group \mathbb{G}	1.709
Map-to-point hash function	4.406

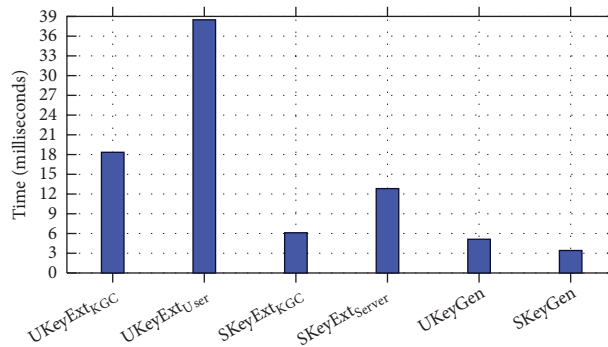


FIGURE 3: Performance of private key extraction procedures in both schemes.

The performance of private key extraction procedures of our DS-IBE-AET scheme and the key generation procedure of Zhao et al.'s construction [9] are depicted in Figure 3. It can be seen that in our DS-IBE-AET scheme, the verification procedures at both the user and server sides take more time than KGC. Note that the private keys for users and servers only need to be extracted once; the computational costs for them are affordable. In Zhao et al.'s construction [9], users and servers can generate private keys for themselves in less than 6 milliseconds, and there is no verification procedure.

The performance of other procedures is depicted in Figure 4, where the case for each procedure to be executed once is considered for both schemes. To encrypt a message, the proposed DS-IBE-AET scheme will take 5 milliseconds more than Zhao et al.'s construction [9], while our data decryption procedure is more efficient. Since the encryption of the authentication token in our scheme is designed in an identity-based setting, it would take more time to encrypt, decrypt, and validate the token than that in Zhao et al.'s construction [9]. For collaboratively performing equality test on two ciphertexts, both

the front and back servers roughly take 7 milliseconds less than Zhao et al.'s construction [9].

The comparison on communication costs between our DS-IBE-AET construction and Zhao et al.'s scheme [9] is shown in Table 4, in terms of the sizes of the user private key, server private key, ciphertext, authorization token, and internal equality test results. Since our DS-IBE-AET construction is designed in an identity-based setting, it requires KGC to issue the private keys for users and servers. As shown in Table 4, each user's private key in our scheme contains three elements in group \mathbb{G} and each server's private key contains only one element in group \mathbb{G} . While for the scheme from Zhao et al. [9], it was developed in a public key setting and the private keys can be respectively generated by each user and server. However, it is well-known that the corresponding public keys for the users and servers should be maintained through the public key infrastructure.

For the ciphertext corresponding to a message, both our DS-IBE-AET construction and Zhao et al.'s scheme [9] are composed of three elements of the same size and enjoy the same communication costs. For the authorization phase, our

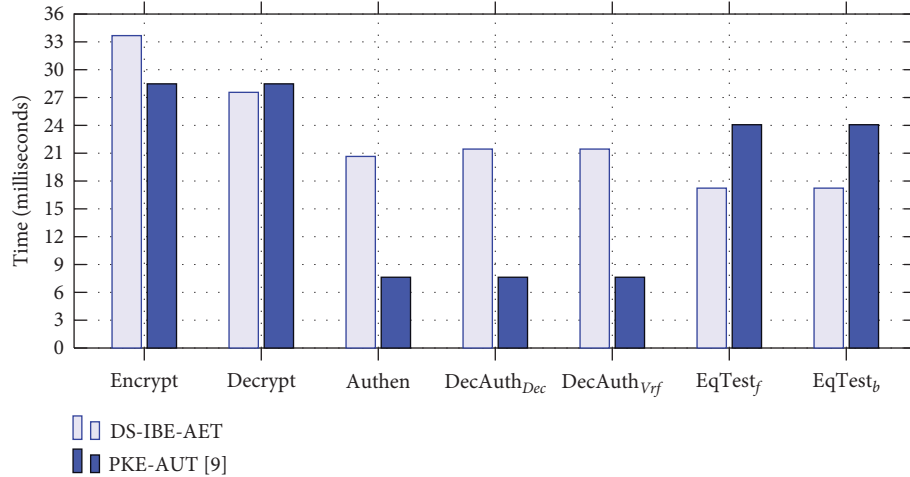


FIGURE 4: Performance of each procedure in our DS-IBE-AET scheme.

TABLE 4: Comparison of communication costs.

	Our DS-IBE-AET scheme	Zhao et al.'s scheme [9]
User private key	$3 \xi_{\mathbb{G}}$	—
Server private key	$\xi_{\mathbb{G}}$	—
Ciphertext	$2 \xi_{\mathbb{G}} + \xi_m + \log q$	$2 \xi_{\mathbb{G}} + \xi_m + \log q$
Authorization	$4 \xi_{\mathbb{G}}$	$2 \xi_{\mathbb{G}} + 2 \log q$
Internal test result	$3 \xi_{\mathbb{G}}$	$3 \xi_{\mathbb{G}}$

DS-IBE-AET construction sends different authorization tokens of the same size of $2\xi_{\mathbb{G}}$ to each server. Whereas in Zhao et al.'s scheme [9], the two servers would receive an identical authorization token containing one element in group \mathbb{G} and one element in \mathbb{Z}_q . In the equality test phase, both schemes require the front server to deliver the internal test result to the back server, which comprises three elements in group \mathbb{G} for comparing a pair of ciphertexts.

6. Conclusion

This paper proposed an identity-based encryption with authorized equality test on ciphertexts in a dual-server setting (DS-IBE-AET), which addressed the complicated certificate management problem in existing proposals supporting equality test on ciphertexts in a public key setting. Particularly, the proposed DS-IBE-AET construction can resist keyword guessing attacks on outsourced ciphertexts that are only stored on the front server side. Only after obtaining the authentication from users would the front server and back server be able to collaboratively perform equality tests on the ciphertexts of these users, where the front server generates an internal test result for further confirmation by the back server. Security analysis demonstrated that the presented DS-IBE-AET scheme can protect the privacy of outsourced ciphertexts and authentication tokens, and performance analysis showed the practicality of our DS-IBE-AET scheme.

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 article was supported by the Guangxi Natural Science Foundation under grants 2019GXNSFFA245015 and 2019GXNSFGA245004, the National Natural Science Foundation of China under projects 62162017 and 62172119, and the Peng Cheng Laboratory Project of Guangdong Province PCL2021A09, PCL2022A02, and PCL2021A03.

References

- [1] X. Fu, X. Nie, and F. Li, "Large universe attribute based access control with efficient decryption in cloud storage system," *Journal of Systems and Software*, vol. 135, pp. 157–164, 2018.
- [2] X. Fu, Y. Ding, and H. Li, "A survey of lattice based expressive attribute based encryption," *Computer Science Review*, vol. 43, Article ID 100438, 2022.
- [3] D. Boneh, G. Di Crescenzo, R. Persiano, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds., Springer, Heidelberg, Germany, pp. 506–522, 2004.
- [4] G. Yang, C. H. Tan, and D. S. Wong, "Probabilistic public key encryption with equality test," *Topics in Cryptology - CT-RSA 2010*, Springer-Verlag, in *Proceedings of the 2010 International Conference on Topics in Cryptology, CT-RSA'10*, pp. 119–131, June 2010.
- [5] L. Wu, Y. Zhang, K. Choo, and D. He, "Efficient and secure identity-based encryption scheme with equality test in cloud

- computing,” *Future Generation Computer Systems*, vol. 73, pp. 22–31, 2017.
- [6] H. Li, Q. Huang, S. Ma, and W. Susilo, “Authorized equality test on identity-based ciphertexts for secret data sharing via cloud storage,” *IEEE Access*, vol. 7, pp. 25409–25421, Article ID 25409, 2019.
 - [7] Y. Wang and H. H. Pang, “Probabilistic public key encryption for controlled equijoin in relational databases,” *The Computer Journal*, vol. 60, no. 4, pp. 600–612, 2017.
 - [8] M. Ramadan, Y. Liao, F. Li, and H. Abdalla, “IBEET-RSA: identity-based encryption with equality test over RSA for wireless body area networks,” *Mobile Networks and Applications*, vol. 25, no. 1, pp. 223–233, 2020.
 - [9] M. Zhao, Y. Ding, S. Tang, and H. Wang, “Public key encryption with authorized equality test on outsourced ciphertexts for cloud-assisted iot in dual server model,” *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–10, Article ID 4462134, 2022.
 - [10] L. Wu, Y. Zhang, and D. He, “Dual server identity-based encryption with equality test for cloud computing,” *Journal of Computer Research and Development*, vol. 54, no. 10, pp. 2232–2243, 2017.
 - [11] P. Hweehwa and X. Ding, “Privacy-preserving ad-hoc equijoin on outsourced data,” *ACM Transactions on Database Systems*, vol. 39, no. 3, pp. 1–40, October 2014.
 - [12] H. Cui, R. H. Deng, and G. Wu, “Attribute-based storage supporting secure deduplication of encrypted data in cloud,” *IEEE Transactions on Big Data*, vol. 5, no. 3, pp. 330–342, 2019.
 - [13] Y. Zheng, M. Wang, Y. Li, and V. Athanasios, “Vasilakos. “Encrypted data management with deduplication in cloud computing,”” *IEEE Cloud Computing*, vol. 3, no. 2, pp. 28–35, 2016.
 - [14] Y. Wang, H. H. Pang, and H. Robert, “Cca secure encryption supporting authorized equality test on ciphertexts in standard model and its applications,” *Information Sciences*, vol. 414, pp. 289–305, 2017.
 - [15] Y. Wang and Y. Ding, “Privacy-preserving cloud-based road condition monitoring with source authentication in vanets,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1779–1790, 2019.
 - [16] Q. Tang, “Public key encryption supporting plaintext equality test and user-specified authorization,” *Security and Communication Networks*, vol. 5, no. 12, pp. 1351–1362, 2012.
 - [17] Q. Tang, “Towards public key encryption scheme supporting equality test with fine-grained authorization,” in *Proceedings of the 16th Australasian Conference on Information Security and Privacy ACISP’11*, pp. 389–406, Springer-Verlag, Heidelberg, Germany, June 2011.
 - [18] Y. Wang, H. H. Pang, H. D. Robert, D. Yong, W. Qianhong, and Q. Bo, “Securing messaging services through efficient signcryption with designated equality test,” *Information Sciences*, vol. 490, pp. 146–165, 2019.
 - [19] H. T. Lee, San Ling, J. H. Seo, H. Wang, and Y. Y. Taek, “Public key encryption with equality test in the standard model,” *Information Sciences*, vol. 516, pp. 89–108, 2020.
 - [20] C. Li, Q. Shen, and Z. Xie, “Large universe CCA2 CP-abe with equality and validity test in the standard model,” *The Computer Journal*, vol. 64, no. 4, pp. 509–533, 07 2020.
 - [21] Y. Wang, H. H. Pang, H. D. Robert, D. Yong, W. Qianhong, and Q. Bo, “Secure server-aided data sharing clique with attestation,” *Information Sciences*, vol. 522, pp. 80–98, 2020.
 - [22] S. Ma, “Identity-based encryption with outsourced equality test in cloud computing,” *Information Sciences*, vol. 328, pp. 389–402, 2016.
 - [23] T. Wu, S. Ma, Yi Mu, and S. Zeng, “Id-based encryption with equality test against insider attack,” in *Australasian Conference on Information Security and Privacy*, pp. 168–183, Springer International Publishing, New York, NY, USA, 2017.
 - [24] H. T. Lee, H. Wang, and K. Zhang, “Security analysis and modification of id-based encryption with equality test from ACISP 2017,” vol. 10946, pp. 780–786, in *Proceedings of the Information Security and Privacy - 23rd Australasian Conference, ACISP 2018*, vol. 10946, Springer, Wollongong, NSW, Australia, July 2018.
 - [25] S. Alornyo, A. Edward Mensah, and O. A. Abraham, “Identity-based public key cryptographic primitive with delegated equality test against insider attack in cloud computing,” *International Journal on Network Security*, vol. 22, no. 5, pp. 743–751, 2020.
 - [26] Y. Ling, S. Ma, and Q. Huang, “Efficient group ID-based encryption with equality test against insider attack,” *The Computer Journal*, vol. 64, no. 4, 674 pages, 661.
 - [27] R. Chen, Yi Mu, G. Yang, F. Guo, and X. Wang, “Dual-server public-key encryption with keyword search for secure cloud storage,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 789–798, 2016.
 - [28] Q. Tang, “Public key encryption schemes supporting equality test with authorisation of different granularity,” *International Journal of Applied Cryptography*, vol. 2, no. 4, pp. 304–321, 2012.
 - [29] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” *Journal of Cryptology*, vol. 17, pp. 297–319, 2004.
 - [30] H. T. Lee, San Ling, J. H. Seo, and H. Wang, “CCA2 attack and modification of Huang et al.’s public key encryption with authorized equality test,” *The Computer Journal*, vol. 59, no. 11, pp. 1689–1694, 2016.
 - [31] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.
 - [32] S. Ma, M. Zhang, Q. Huang, and Bo Yang, “Public key encryption with delegated equality test in a multi-user setting,” *The Computer Journal*, vol. 58, no. 4, pp. 986–1002, 04 2014.
 - [33] D. He, S. Zeadally, B. Xu, and X. Huang, “An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.
 - [34] S. D. K. Miracl Cryptographic, “Multiprecision integer and rational arithmetic cryptographic library,” 2019, <https://github.com/miracl/miracl>.

Research Article

UIV-TSP: A Blockchain-Enabled Antileakage Sharing Protection Scheme for Undisclosed IIoT Vulnerabilities

Wenbo Zhang , Jing Zhang, Yifei Shi, and Jingyu Feng 

National Engineering Laboratory for Wireless Security, Xi'an University of Posts & Telecommunications, Xi'an 710121, China

Correspondence should be addressed to Jingyu Feng; fengjy@xupt.edu.cn

Received 14 May 2022; Accepted 14 September 2022; Published 10 October 2022

Academic Editor: Andrea Michienzi

Copyright © 2022 Wenbo 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.

With the large-scale deployment of industrial Internet of things (IIoT) devices in 5/6G environments, the number of vulnerabilities threatening IIoT security is growing dramatically, including a mass of undisclosed IIoT vulnerabilities that lack mitigation measures. Coordination vulnerability disclosure (CVD) is one of the most popular vulnerabilities sharing solutions, in which security workers (SWs) can develop undisclosed vulnerability patches together. However, CVD assumes that SWs are all honest and thus offering chances for dishonest SWs to internally leak undisclosed IIoT vulnerabilities. To combat such internal threats, we propose an undisclosed IIoT vulnerabilities sharing protection (UIV-TSP) scheme against internal leakage. In this paper, a dynamic token is an implicit access credential for an SW to acquire an undisclosed vulnerability message, which is only held by the system and constantly updated with the SW access. The latest updated token can be stealthily sneaked into the acquired information as the traceability token to prevent internal leakage. To quickly distinguish dishonest SWs, the feedforward neural network (FNN) is adopted to evaluate the trust value of SWs. Meanwhile, we design a blockchain-assisted continuous logs storage method to achieve the tamper-proofing of dynamic token and the transparency of undisclosed IIoT vulnerabilities sharing. The simulation results indicate that our proposed scheme is resilient to suppress dishonest SWs and protect the IIoT undisclosed vulnerabilities effectively.

1. Introduction

With the gradual deployments and applications of 5/6G, the Internet of things (IoT) technology is being applied to every part of our lives [1–4]. As a subset of IoT, industrial Internet of things (IIoT) has recently attracted attention [5]. By leveraging sensors, actuators, GPS devices, and mobile devices, the IIoT technology is being applied to advance the development of many industrial systems [6]. The industrial systems where this IIoT technology is integrated include energy [7], manufacturing [8], logistics [9], and transportation [10].

Currently, IIoT devices have been widely deployed with weak security features or a lack of security [11]. These features have made IIoT devices as a good target for attackers with malicious intentions, and in many cases, exploits using IIoT devices have been occurring [12]. There is an urgent need for a solution that provides a

lightweight and low-cost mechanism for collaborative security response of IIoT devices against emerging vulnerabilities [13].

However, the IIoT vendors generally have weak security emergency response capabilities. It is better to invite some security workers (such as organizations, institutions, or white hats) to help them mitigate the new vulnerabilities of IIoT devices. In order to standardize the process of vulnerabilities patching and accelerate the development of mitigation measures, the vulnerability disclosure policy has been presented in [14], including vulnerabilities reporting, sharing, coordinating, and patching. An IIoT vulnerability can be officially disclosed after the patch is made; otherwise, it is called an undisclosed IIoT vulnerability (uiv). According to the vulnerability disclosure policy, the IIoT vendors can report a new uiv and share it with some security workers (SWs) who develop their patches together by means of the coordination vulnerability disclosure (CVD).

Unfortunately, CVD is a set of guidelines without mandatory measures [14, 15]. CVD assumes that SWs are all honest and thus offering chances for dishonest SWs to internally leak undisclosed IIoT vulnerabilities. Due to the widespread use of IIoT devices, the leakage of an uiv message could cause a large-scale damage. Therefore, it is necessary to prevent the internal leakage of the uiv. Although a lot of works have been done in the field of threat intelligence sharing [16, 17], they focus on the sharing protection of disclosed vulnerability information. Little attention has been paid to the sharing protection of undisclosed vulnerability information.

In this paper, we propose an undisclosed IIoT vulnerabilities trusted sharing protection (UIV-TSP) scheme against internal leakage. To enable endogenously secure IIoT, our final objective is to prevent the leakage of uiv until their patches are released. The main contributions of this paper are as follows:

- (1) Introduce dynamic token as the implicit access credential and traceability clue for an SW. When uploading a new uiv, each internal SW is assigned a corresponding token called $\text{token}_{\text{access}}$, which is only held by the system and cannot be seen by anyone. Even if an SW is granted access through identify authentication, an uiv message cannot be acquired without $\text{token}_{\text{access}}$. To avoid malicious inference, $\text{token}_{\text{access}}$ should be updated dynamically. At the end of SW access, the current $\text{token}_{\text{access}}$ is revoked. A new random number is integrated into the hash generation of token to get a new $\text{token}_{\text{access}}$ as the next access credential. Meanwhile, the current $\text{token}_{\text{access}}$ and the MAC address of SWs are hashed to create the traceability token called $\text{token}_{\text{tracing}}$, which is embedded in the undisclosed IIoT acquired by an SW.
- (2) Design a blockchain-assisted method to store the continuous logs of all SWs for the UIV-TSP scheme. To ensure the tamper-proofing of dynamic token, the original token and its all-subsequent updates should be stored on the blockchain. To achieve the transparency of uiv sharing, their metadata and the related SW access records are also stored on the blockchain.
- (3) Present an internal leakage prevention method with one-step traceability. A benign logic bomb called $\text{code}_{\text{preleak}}$ is embedded into an uiv message, which checks that whether the current MAC address is the same as the preset destination address in $\text{token}_{\text{tracing}}$. Due to the confidentiality, an uiv message can only be reached by one step to the SW host that is licensed by the system. Once the uiv information leaves the SW host, $\text{code}_{\text{preleak}}$ will automatically trigger the self-destruct program to prevent leaks.
- (4) Adopt the trust mechanism based on deep learning to evaluate the trust value of SWs according to their historical behaviors in an automatic and dynamic manner. With high trust value, honest SWs would be

accepted to acquire uiv. With low trust value, dishonest SWs would be rejected. With medium trust value, it is difficult to determine the access authorities of semihonest SWs who may be suspiciously dishonest SWs. In this case, we can release a false uiv with $\text{token}_{\text{tracing}}$ to trap their external conspirators.

The architecture of this paper is as follows. In Section 2, we introduce the related works. Our UIV-TSP scheme is proposed in Section 3. In Section 4, we analyze the performance of UIV-TSP from the perspective of security and cost. We also discuss the application of UIV-TSP solution and future work in Section 5. Finally, we conclude the paper in Section 6.

2. Related Work

2.1. Vulnerability Disclosure Policy. ISO/IEC 29147 defines vulnerability disclosure as a process through which vendors and vulnerability finders may work cooperatively in finding solutions that reduce the risks associated with a vulnerability [14]. Currently, CVD [15] is a good choice to develop undisclosed vulnerability patches together among security workers. As shown in Figure 1, the CVD process is consisted of gathering, coordinating, disclosing, and patching, and more details are given in [15]. Furthermore, there are two extreme cases for vulnerability disclosure: (1) public disclosure: disclosure as soon as the vulnerability information is received. (2) private disclosure: keep the vulnerability information security. Both of them are regarded as irresponsible sharing.

These vulnerability disclosure schemes all rely on guidelines, but in practice, guidelines are not mandatory. Once a participant breaks the guidelines, the entire vulnerability disclosure process will be paralyzed, thus increasing the threat risks from the attackers.

2.2. Threat Intelligence Sharing. To prevent the leak of sensitive data in threat intelligence containing uiv, many studies have integrated cryptography primitives into their threat intelligence sharing scheme. Vakili et al. [16] designed a mechanism enables the organizations to share their cybersecurity information anonymously. Meanwhile, they proposed a new blind signature based on BBS+ to reward contributions anonymously. Badsha et al. [17] proposed a privacy preserving protocol where organizations can share their private information as an encrypted form with others and they can learn the information for future prediction without disclosing any private information. de Fuentes et al. [18] introduced PRACIS, a scheme for cybersecurity information sharing that guarantees private data forwarding and aggregation by combining STIX and homomorphic encryption primitives. Homan et al. [19] leveraged the security properties of blockchain and designed a more effective and efficient framework for cybersecurity information sharing network. Preuveneers et al. [20] employed blockchain and CP-ABE to offer fine-grained

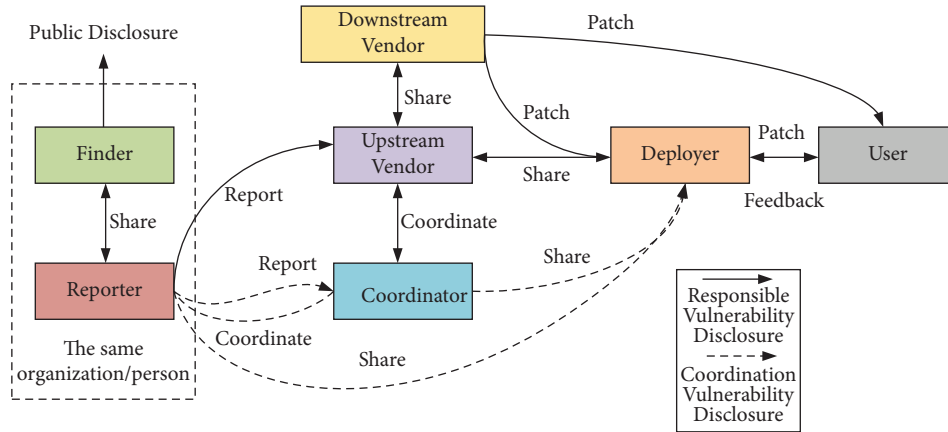


FIGURE 1: Flowchart of vulnerability disclosure policy [15].

protection and trustworthy threat intelligence sharing with the ability to audit the provenance of threat intelligence.

However, these schemes are helpful to protect the uiv information sharing, while the sharing protection of uiv information has not been involved. Without mitigation measures, the leakage of an undisclosed IIoT vulnerability could cause a large-scale damage [21–23]. Therefore, while protecting the sharing of uiv information, the responsibility of SWs should be traced back.

2.3. Data Leakage Prevention. Currently, some researchers focus on leveraging cryptography algorithms to implement an accountable and efficient data sharing in research of tracing data leakage. Mangipudi et al. [24] presented a committed receiver oblivious transfer (CROT) primitive to fairly track the traitor of leaked data by oblivious transfer (OT) protocol and zero knowledge (ZkPok). Huang et al. [25] designed an accountable and efficient data sharing scheme for industrial IoT (IIoT), named ADS/R-ADS/E-ADS, in which data receiver’s private key (i.e., evidence) is embedded in sharing data, and data owner can pursue the responsibility of a public for profits while without permission. Zhang et al. [26] proposed a fair traitor tracing scheme to secure media sharing in the encrypted cloud media center by proxy re-encryption and fair watermarking. Ning et al. [27] presented a traitor tracing with CP-ABE scheme by two kinds of noninteractive commitments. Based on signatures, Imine et al. [28] proposed a novel accountable privacy-preserving solution for public information sharing allows to trace malicious users.

The above schemes all contribute to the sharing of threat intelligence. Nevertheless, these schemes cannot deal well with the trade-off between traceability time and robustness. A lightweight traceability scheme is required to uiv sharing.

3. Our Proposed UIV-TSP Scheme

With dynamic token, we propose an undisclosed IIoT vulnerabilities sharing protection scheme called UIV-TSP to prevent uiv leakage until their patches are released. Concretely, the UIV-TSP scheme consists of four collaborative

modules: dynamic token management, blockchain-assisted continuous logs storage, internal leakage prevention with one-step traceability, and trust-based SWs distinction.

3.1. System Architecture. To prevent uiv leakage, we first present the system architecture of the UIV-TSP scheme. Figure 2 illustrates the overall system architecture of our scheme, which consists of the following entities:

- (1) Trusted authority (TA): trusted authority is responsible for processing SW’s access requests, generating and updating dynamic token, and evaluating trust value of SW_i .
- (2) Security workers (SWs): there exist some workers who access uiv information in the sharing environment to develop their mitigation measures. We describe three types of workers: (1) honest SWs who do not engage in unauthorized access; (2) semi-honest SWs who have a chance of committing malicious behavior; (3) dishonest SWs who often leak uiv information. In this paper, SWs are defined as $\{SW_1, SW_2 \dots SW_m\}$.

3.2. Dynamic Token Management. We introduce dynamic token as the implicit access credential and traceability clue for an SW. As shown in Figure 3, the lifecycle of dynamic token is consisted of generation and update.

3.2.1. Token Generation. When an SW submits a new undisclosed vulnerability vul_j , each internal SW is assigned a corresponding token called $token_{access}$, which is only held by the system and cannot be seen by anyone. TA generates the initial $token_{access}$ through a hash function. Meanwhile, we define vul_{meta} as the meta information of an uiv. The initial $token_{access}$ can be calculated as follows:

$$token_{access} = H(SW_i || vul_{meta} || tp || nonce). \quad (1)$$

Even if an SW is granted access through identify authentication, the uiv cannot be acquired without $token_{access}$. Algorithm 1 is performed to generate and update $token_{access}$.

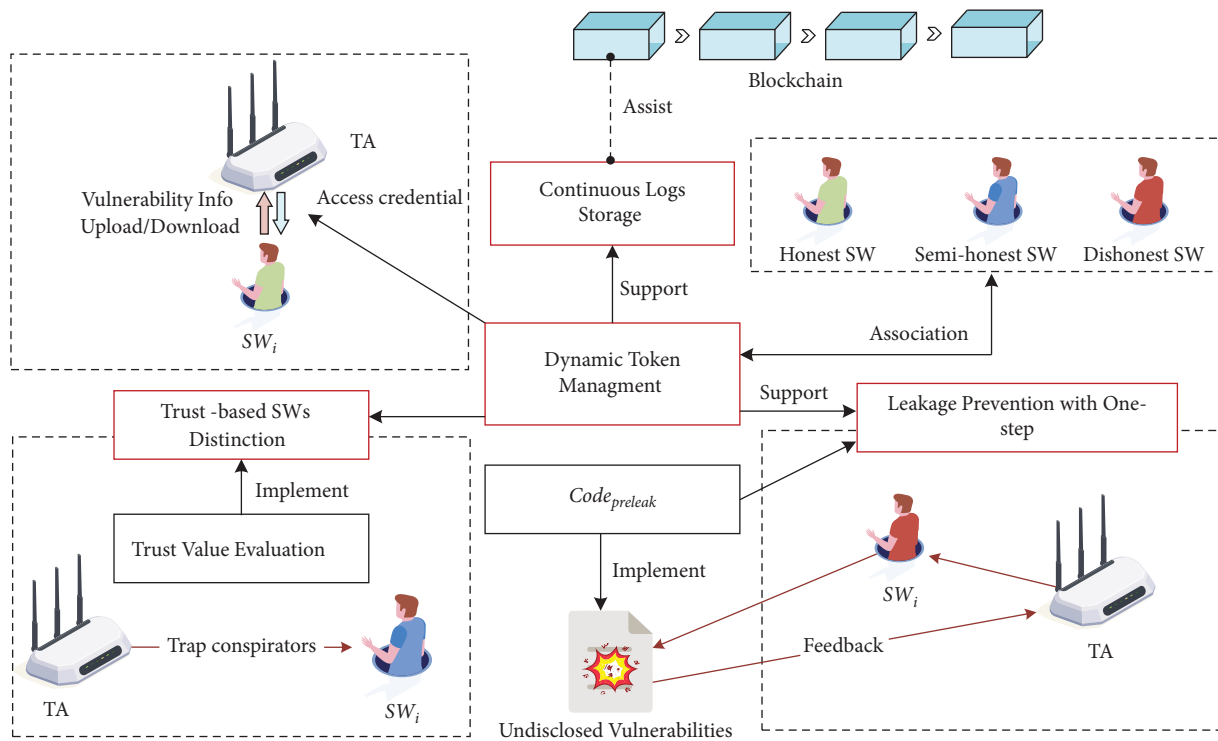


FIGURE 2: System architecture of the UIV-SP scheme.

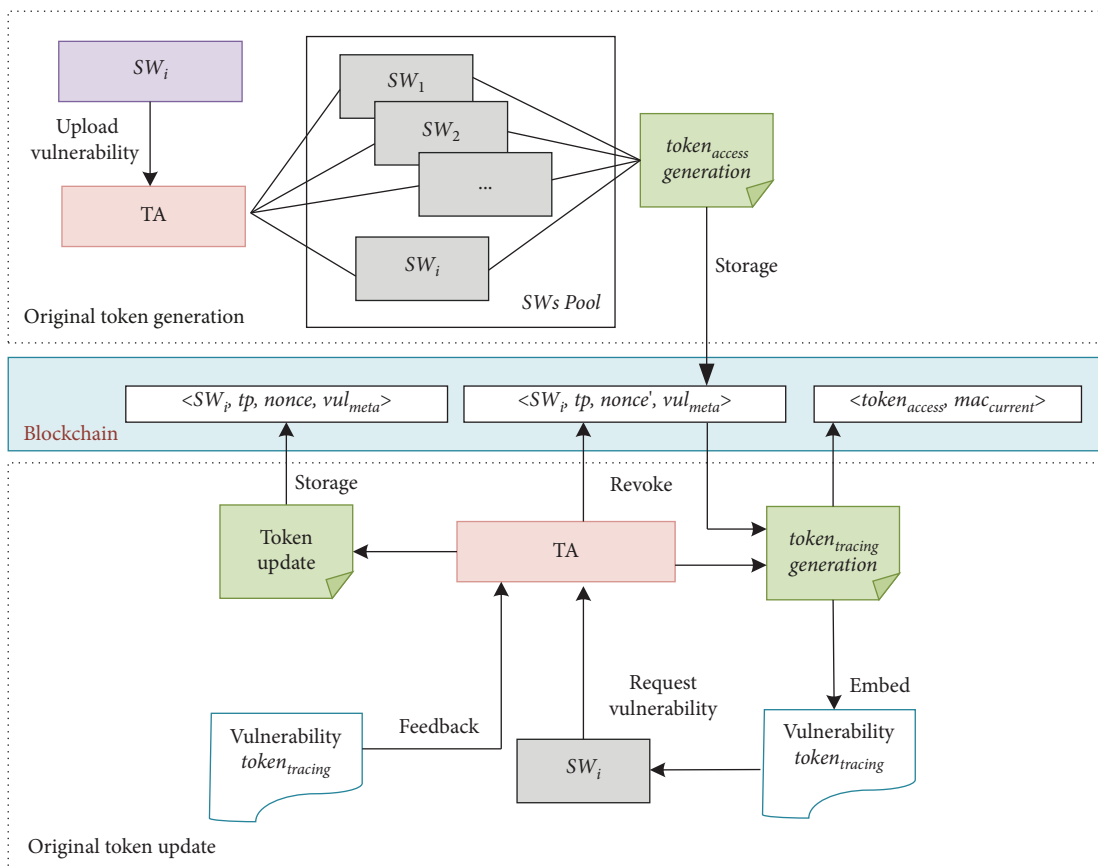


FIGURE 3: Lifecycle of dynamic tokens.

```

(i) Input:  $SW_i$ 
(ii) Output:  $token_{access}$ 
(1) If  $SW_i$  in  $SW_{pool}$  then
(2)    $token_{access} = H(SW_i || vul_{meta} || tp || nonce)$ .
(3)   If  $token_{access}$  in blockchain then
(4)     Update  $token_{access}$ ,  $token_{access} \leftrightarrow SW_i$ 
(5)   else
(6)     Store  $token_{access}$ ,  $token_{access} \leftrightarrow SW_i$ 
(7)     If end of request access, then
(8)       Revoke  $token_{access}$ 
(9)        $token_{tracing} \leftarrow H(token_{access} || mac_{current})$ 
(10)       $token_{access} \leftarrow H(SW_i || vul_{meta} || tp || nonce')$ 
(11)      Store  $token_{access}$ ,  $SW_i \leftrightarrow token_{tracing}$ 
(12)      Embed  $token_{tracing}$  to  $vul_j$ 
(13)     End If
(14)   End If
(15) End If

```

ALGORITHM 1: Pseudocode of token generation and update.

3.2.2. *Token Update.* To avoid malicious inference, we integrate a one-time random number into the hash generation of token to update token dynamically. At the end of SW access, $token_{access}$ will be update as follows:

$$token_{access} \leftarrow H(SW_i || vul_{meta} || tp || nonce'). \quad (2)$$

Meanwhile, the current $token_{access}$ and the MAC address of the SW are hashed to create the traceability token called $token_{tracing}$. $Token_{tracing}$ can be defined as follows:

$$token_{tracing} \leftarrow H(token_{access} || mac_{current}). \quad (3)$$

In our scheme, $token_{tracing}$ can be stealthily sneaked into the acquired vul_j information as the traceability credential. The execution strategies of dynamic token management can be executed with four steps, the more details are given in [29].

3.3. *Blockchain-Assisted Continuous Logs Storage.* Due to the advantages including transparency, traceability, and tamper-proofing, many studies have integrated blockchain into the prior works to implement a reliable and efficient data storage [30]. In general, there are three types of blockchain data storage patterns [31]:

- (1) Public blockchain: a public blockchain is the blockchain that can read by anyone in the world, anyone can send transactions to and expect to see them included, if they are valid, and anyone can participate in the consensus process, which determines what blocks get added to the chain and what the current state is.
- (2) Private blockchain: a private blockchain is the blockchain where can write by only one organization.

- (3) Consortium blockchain: a consortium blockchain is the blockchain where the consensus process is controlled by a preselected set of miners.

Since token can only be held by the system, the public blockchain does not meet the requirements. Hence, a private blockchain-assisted storage method is designed to centralized storage logs. To prevent attackers from tampering with data, the logs of SWs' activities in the shared process should also be continuously recorded on the blockchain. These continuous logs, including dynamic token, trust value of SW (Tr_i), and behaviors record ($R[i]$), are also only held by the system. It can be found that the private blockchain is suitable for our scheme.

In the blockchain, the block structure of continuous logs storage is shown in Figure 4.

3.3.1. *Block Head.* The block head is slightly different from the traditional structure. Except for previous hash, timestamp, Merkle root, and block ID, several new elements are integrated into the block head:

- (i) SW_i : the ID of the i -th SW who requests the undisclosed vulnerability in the sharing environment.
- (ii) Tr_i : the trust value of SW_i . In the block head, Tr_i can be quickly retrieved by TA.
- (iii) vul_{meta} : the meta information of an undisclosed IIoT vulnerability.

3.3.2. *Block Body.* In the block body, the log data of an SW (such as SW_i) are hashed to build the Merkle tree. Except for $token_{access}$ and $token_{tracing}$, the log data of SW_i contain the following elements:

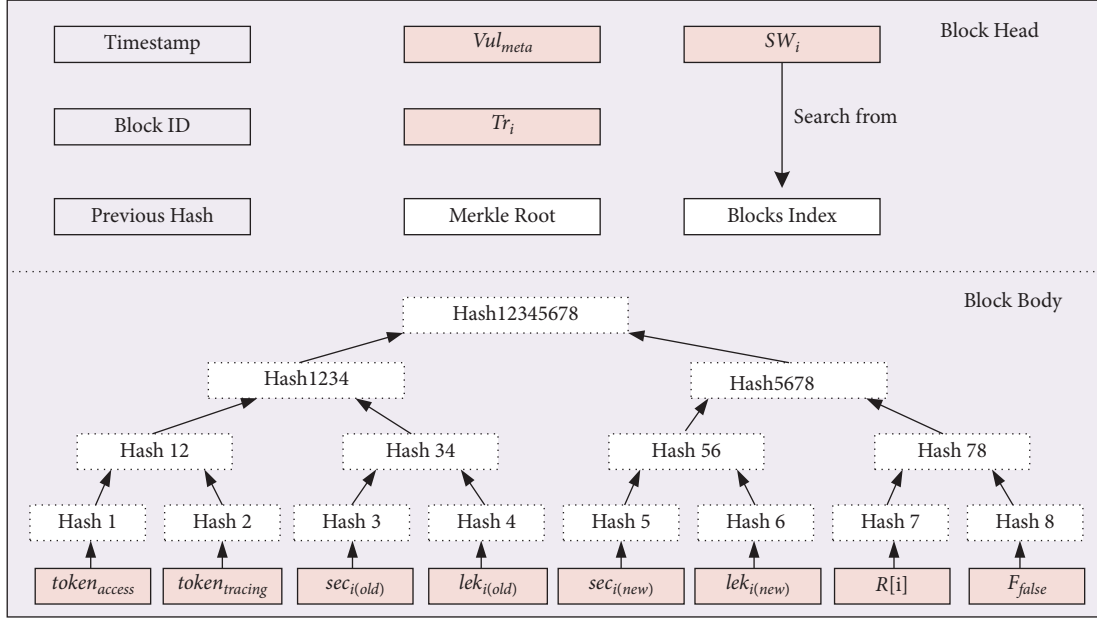


FIGURE 4: Block structure of continuous logs storage.

- (i) $(sec_{i(old)}, lek_{i(old)})$: the historical trust data of SW_i , which can be used to evaluate the trust value of SW_i before the access.
- (ii) $(sec_{i(new)}, lek_{i(new)})$: the current trust data of SW_i , which can be used to update the trust value of SW_i after the access.
- (iii) $R[i]$: the access request record of SW_i to an uiv information.
- (iv) F_{false} : the flag whether a false uiv information has been released.

3.4. Internal Leakage Prevention with One-Step Traceability. To prevent SWs leaking the acquired vul_j information, vul_j should be self-destruct when they leave the host of SWs one-step. Thus, we design a benign self-triggering logic bomb $code_{preleak}$. A logic bomb is a piece of code consisting of a trigger condition and a payload; when the trigger condition is met, the bomb is triggered (or activated) and the payload code gets executed [32].

$code_{preleak}$ is composed of trigger condition and response payloads. The trigger condition is designed to detect the access environment difference between honest SWs and dishonest SWs, so that the protection payload will be activated to destroy vul_j on the leakage side.

As shown in Figure 5, the functional structure of $code_{preleak}$ is consisted of self-checking and self-destruct.

3.4.1. Self-Check. Once the uiv information enters the SW host, $code_{preleak}$ will extract the current SW host Mac address and the revoked token to compute the verification value. Then, $code_{preleak}$ will match the verification value to $token_{tracing}$. If the result V_c is inconsistent, it will trigger self-destruct. V_c can be calculated as follows:

$$V_c \leftarrow token_{tracing} == H(token_{access}, mac_{current})?. \quad (4)$$

Algorithm 2 is performed to match verification value.

3.4.2. Self-Destruct. If $V_c = 0$, the leakage has not happened. That is, the uiv information has not left the SW host. The protection payload continues to lurk.

If $V_c = 1$, it may leak vul_j . That is, the uiv information has left the SW host. In this case, the protection payload will be activated immediately to destroy vul_j .

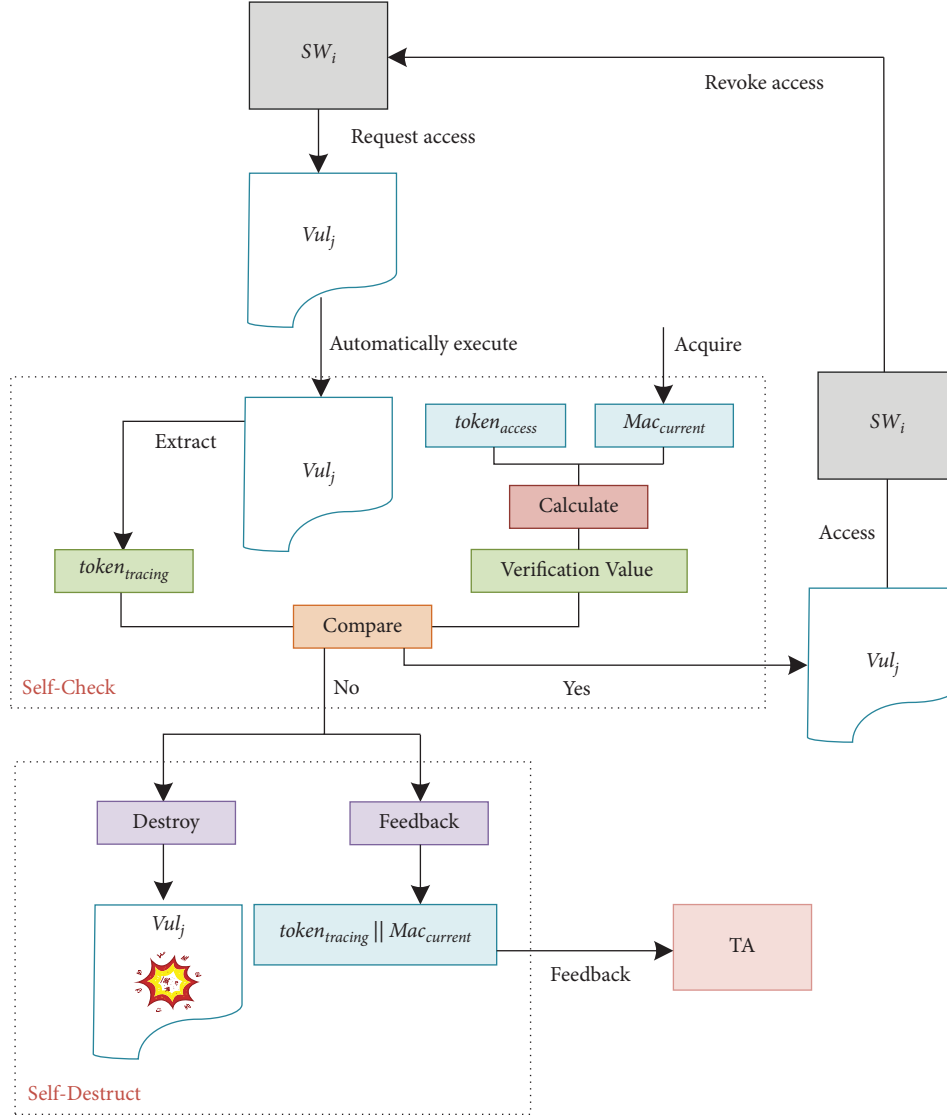
At the same time as the self-destruct event, $code_{preleak}$ can automatically send encrypted feedback $ef = \{token_{tracing}, vul_j, SW_i, mac_{current}, t_{feedback}\}$ to TA.

- (i) vul_j : the ID of undisclosed IIoT vulnerability.
- (ii) $t_{feedback}$: the time to $code_{preleak}$ send the feedback message.

Algorithm 2 is performed to activate the protection payload.

3.5. Trust-Based SWs Distinction. Trust mechanism can be adopted to evaluate the trust value of SWs according to their historical behaviors. With trust value, we can quickly distinguish honest SWs and dishonest SWs. For semi-honest SWs, we can further validate their credibility by tracing whether they have external conspirators. Different SWs will gain different access authorities to acquire uiv information.

3.5.1. Trust Value Evaluation. In the process of vulnerabilities information sharing, the behaviors of an SW are generally dualistic: secret-keeping and leakage. With trust


 FIGURE 5: Functional structure of code_{preleak}.

mechanism, if an SW often leaks information, he will get a low trust value.

To quantify these duality behaviors, the beta function is one of the most popular trust value evaluation methods. It first counts the number of secret-keeping and leakage by an SW and then calculates the trust value with beta function denoted by $\text{Beta}(\alpha, \beta)$ [33].

$$\text{Beta}(\alpha, \beta) = \frac{\Gamma(\alpha, \beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1}, \quad (5)$$

where θ is the probability of duality behaviors, $0 \leq \theta \leq 1$, $\alpha > 0, \beta > 0$.

Take SW_i as an example, sec_i and lek_i denote the number of secret-keeping and leakage in the sharing environment, respectively. The expectation value of the beta function can be calculated as follows: $E[\text{Beta}(\alpha, \beta)] = \alpha / (\alpha + \beta)$. Considering that the trust value BT_i is limited in the interval $[0, 1]$, BT_i can be described as follows:

$$BT_i = \frac{1 + sec_i}{1 + sec_i + lek_i}. \quad (6)$$

When $sec_i \geq 1$ and $lek_i = 0$, BT_i is always calculated as 1. The SW_i is completely trusted under this condition.

Obviously, the base trust value BT_i that decays too slowly will give dishonest SWs more opportunities to leak. So, it is very essential to introduce a penalty factor, which can be calculated as follows:

$$P_i = e^{-(sec_i + lek_i) / sec_i}. \quad (7)$$

With the punishment of P_i to BT_i , the trust value Tr_i of SW can be further evaluated as follows:

$$Tr_i = \begin{cases} \frac{1 + sec_i}{1 + sec_i + lek_i} \cdot e^{-(sec_i + lek_i) / sec_i}, & lek_i < sec_i, \\ 0, & lek_i \geq sec_i. \end{cases} \quad (8)$$

In the (8), the Tr_i means the comprehensive trust value of SWs, which introduces a penalty factor to cause Tr_i to decrease faster when leakage occurs. Unfortunately, the trust mechanism utilizing the beta reputation engine is suitable for scenarios with a small amount of trust data. When the trustworthiness of SW_i is not represented by duality behaviors, as in the case of the collusion clique construction, it is possible for some malicious SWs to exhibit rational behaviors to avoid the detection. That is, they can keep high trust value in an alternant process by truly sharing their uiv information or leaking uiv information to their conspirators.

To suppress such behaviors, the special reputation engine is adopted to prevent the SW trust value growth of some rational conspirators at the uiv information sharing. Once the TA receives messages from an SW regarding some uiv information requests, the trust value of the SW will calculate by applying the feedforward neural network (FNN) algorithm [34], which is depicted in Figure 6.

The input layer of FNN consists of 11 input nodes, such as the collaborators of the target SW, the average trust value of SW collaborators, and the number of SW's conspirators. The complete input feature description is shown in Table 1. There are two hidden layers with totally 16 hidden neurons. The output layer produces the trust value of SW, so the result of SW trust evaluation depends on a set of individual trust parameters rather than duality parameters. Likewise, the output of the FNN algorithm will be stored on the blockchain.

As we know, it is essential to learn which appropriate weights and biases that make FNN have good performance. The feedforward neural network based on BP (back-propagation) algorithm constructs the identification of plant and inverse controller. Formula (9) describes how the cost in a neural network is computed.

$$\text{cost} = \sum_{j=0}^{n^{L-1}} (\alpha_j^L - y_j)^2, \quad (9)$$

where α_j^L represents the j -th activation of the last neuron, and L represents the current layer. Then, the j -th activation of the previous layer is $\alpha_j^{(L-1)}$. Assuming there are L hidden layers in total, then n^{L-1} represents the neuron in the last hidden layer before the output layer. In actual calculations, costs are always finding the differences of each neuron from the expected target output minus the current output.

By introducing the feedforward neural network algorithm, the trust value of SWs can be calculated accurately and learn the potential behavior patterns among the malicious SWs. In this case, we further design the distinction rules can successfully identify which SWs are malicious.

3.5.2. Distinction Rules. SWs can be split into honest, dishonest, and semihonest based on their trust value. $(\delta_h, \delta_l, \delta_m)$ are, respectively, set as the threshold of high, low, and medium trust value. The specific distinction rules are as follows:

R1: for $Tr_i > \delta_h$, the SW_i 's access request for an uiv message will be accepted. In this situation, SW_i is classified as honest.

R2: for $Tr_i < \delta_l$, the SW_i 's access request for an uiv message will be rejected. In this situation, SW_i is classified as dishonest.

R3: for $\delta_l < Tr_i < \delta_m$, it is difficult to determine the access authorities of SW_i . In this situation, SW_i is classified as semihonest who may be suspiciously dishonest SWs.

To validate the credibility of semihonest SWs, we can trace whether they have conspirators. Let $\mu_i(\cdot)$ denote number of conspirators of SW_i .

If $\mu_i = 0$, there are no conspirators. Hence, SW_i is temporarily considered as honest.

If $\mu_i \geq 1$, SW_i may have several external conspirators. The trust value of SW_i will be set to 0. As a result, SW_i will be removed from the CVD and barred from rejoining.

3.5.3. Trap External Conspirators. If SW_i is a semihonest SW, a false uiv message can be released to trap his external conspirators. This false uiv information is set to a valid time.

Within the valid time, $\text{code}_{\text{preleak}}$ will not trigger the protection payload and provide the feedback messages, which can be employed to build a set of leak path. Once an external conspirator is trapped, SW_i can be regarded as dishonest.

After the valid time, the protection payload is activated to destroy the false uiv message, so as not to spread too widely. Algorithm 3 is performed to trap external conspirators.

4. Performance Analysis

In this section, we conduct performance analysis on the UIV-TSP scheme. We analyze the security of the proposed scheme and then perform computer simulation to further analyze the cost of the proposed scheme.

4.1. Security Analysis. In this section, we analyze how UIV-TSP can achieve the following security requirements: uiv sharing against leakage, blockchain storage against continuous logs tampering. To analyze the security better, we also compare UIV-TSP with some typical data leakage prevention (DLP) schemes in Table 2.

4.1.1. UIV Sharing against Internal Leakage. Challenge 1: Dishonest SW may disclose uiv to cause a large-scale damage.

Lemma 1. *UIV-TSP is resistant internal leakage for uiv.*

Proof. In the UIV-TSP scheme, a benign logic bomb called $\text{code}_{\text{preleak}}$ is embedded into the uiv message, which checks that whether the current MAC address is the same as the present destination address in $\text{token}_{\text{tracing}}$. The MAC address of the device is always fixed, it is impossible to be modified. Furthermore, the $\text{token}_{\text{tracing}}$ is calculated by the hash operation $h(\cdot)$, and $\text{token}_{\text{access}}$ that is dynamically updated at the end of each uiv access. The dual dynamics ensures that

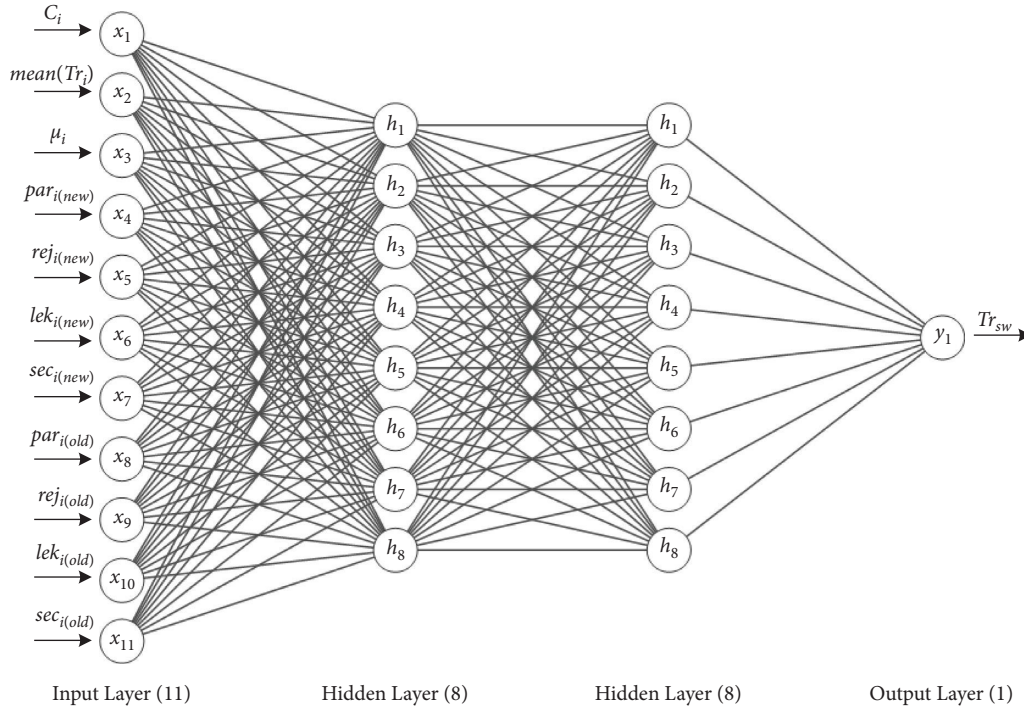


FIGURE 6: Structure of the feedforward neural network.

```

(i) Input: uiv
(ii) Output: access/deny
(1) tokentracing = extract(vulj)
(2) If tokentracing not in vulj
(3)   Go to step 15.
(4) Else
(5)   Acquire MACcurrent
(6)   Vc ← tokentracing == H(tokenaccess, maccurrent)?
(7)   If Vc = 1 then
(8)     Assign access authorities
(9)   If Extract(vulj) == Ffalse then
(10)    Go to step 8.
(11)  Else
(12)    Send an encrypted feedback message ef to TA.
(13)    Destroy the target vulj
(14)  End if
(15) End if

```

ALGORITHM 2: Pseudocode implementation of Match verification value and self-destruct.

token_{tracing} cannot be inferred. According to $V_c \leftarrow \text{token}_{\text{tracing}} == H(\text{token}_{\text{access}}, \text{mac}_{\text{current}})?$, $V_c = 1$ makes the protection payload be activated immediately to destroy vul_j , so it is believed that our UIV-TSP scheme can resist uiv leakage.

4.1.2. Blockchain Storage against Continuous Logs Tampering. Challenge 2: the trusted sharing environment with leakage-resilience construction relies on the trust value evaluation. Trust mechanisms evaluate the trust value of

SWs according to their historical behaviors. Attackers can tamper with their historical logs to disturb trust mechanism and promote their trust quickly. Therefore, it is impossible to distinguish honest SWs.

Lemma 2. *UIV-TSP is resistant to continuous logs tampering.*

Proof. In our UIV-TSP scheme, a blockchain-assisted method to store the continuous logs of all SWs for the

TABLE 1: The input features of the FNN algorithm.

Input features	Description of the input features
Number of the target SW collaborators	A collection of collaborators for the target SW, including collaborator identities. An integer representing the number of SW's collaborators $c_i \in [0, 2000]$
Average trust value of SW collaborators	A real number between 0 and 1, where 0 and 1 represent the honest collaborator and dishonest collaborator, respectively.
Number of SW's conspirators	An integer representing the number of SW's conspirators $\mu_i \in [0, 2000]$.
Current number of SW participating behaviors	The latest uiv shared and statistical behavior characteristics, represented as a real number $par_{i(new)} \in [0, 100]$
Current number of SW rejected behaviors	The latest uiv shared and statistical behavior characteristics, represented as a real number $rej_{i(new)} \in [0, 100]$
Current number of SW leak behaviors	The latest uiv shared and statistical behavior characteristics, represented as a real number $lek_{i(new)} \in [0, 50]$
Current number of SW keep-secret behaviors	The latest uiv shared and statistical behavior characteristics, represented as a real number $sec_{i(new)} \in [0, 50]$
Number of SW historical participating behaviors	Previous uiv shared and statistical behavior characteristics, represented as a real number $par_{i(old)} \in [0, 100]$
Number of SW historical rejected behaviors	Previous uiv shared and statistical behavior characteristics, represented as a real number $rej_{i(old)} \in [0, 100]$
Number of SW historical leak behaviors	Previous uiv shared and statistical behavior characteristics, represented as a real number $lek_{i(old)} \in [0, 50]$
Number of SW historical keep-secret behaviors	Previous uiv shared and statistical behavior characteristics, represented as a real number $sec_{i(old)} \in [0, 50]$

```

(i) Input:  $SW_i$ 
(ii) Output:  $path_i$ 
(1) Init trust value of SW,  $path_i$ ,  $\mu_i$  number of conspirators of SW
(2)  $Tr_i$  evaluate based on the feedback  $e_f$ , retrieves SW historical trust values  $Tr_i$ 
(3) If  $\delta_l < Tr_i < \delta_m$  then
(4)   Release false  $vul_j$  to SW to induce unauthorized behavior,  $F_{false} = true$ 
(5)    $SW_i \leftrightarrow token_{tracing}$  Extract the  $token_{tracing}$  corresponding to the malicious SW
(6)    $\mu_i + +$  increased number of SW conspirators
(7)   If valid time expire then
(8)     Invoke Self- Destruct
(9)   Else:
(10)     $mac_{current}$  not in  $path_i$ 
(11)     $path_i$  add ( $mac_{current}$ )
(12)  End if
(13) Else
(14)   Release normal  $vul_j$  to SW.
(15) End if

```

ALGORITHM 3: Pseudocode implementation of trap external conspirators.

trusted sharing environment with leakage-resilience construction. The original token and its all-subsequent updates should be stored on the blockchain, which is a sequence of blocks, which holds a complete list of transaction records like conventional public ledger. Each block points to the immediately previous block via a reference that is essentially a hash value of the previous block called parent block [35]. Just like a linked list, each block depends on its previous block. Therefore, if the continuous logs maintained in a block are tampered, its latter blocks chained on the blockchain must be modified. Obviously, the longer the blockchain created, the higher cost it takes to tamper with a block. Assuming the number of latter blocks related on Block p is l_p under the condition that there are a number of

sharing regions, the number of blocks that need to be tampered (t_p) is calculated as follows:

$$t_p = P_m + (P_m)^2 + \dots + (P_m)^{l_p} = \frac{(P_m)^{l_p+1} - P_m}{P_m - 1}. \quad (10)$$

For instance, $t_p = 1,398,100$ when $n_r = 4$ and $l_p = 10$.

Therefore, it is nearly impossible to tamper with the continuous logs maintained in Block p due to the huge resource consumption.

4.2. *Experimental Analysis.* We perform simulations to validate the performance of the UIV-TSP scheme in Python 3.7.6, combined with NumPy, pandas, matplotlib, keras,

TABLE 2: Comparison of data leak prevention scheme.

Scheme	Data leakage preventing	Transmission confidentiality	Continuous logs tampering
UIV-TSP	Yes	Yes	Yes
CROT	No	No	No
ADS	No	Yes	No
R-ADS	No	Yes	No
E-ADS	No	Yes	No

TABLE 3: Description of simulation elements.

Parameters	Description	Default
SW_i	Number of SWs	2000
per	Percentage of dishonest SWs	10%–50%
δ	Threshold of trust value	(0.2, 0.5, 0.8)
$Cycle$	Number of cycle simulation	200
ε	Embedded times of access credential	(1, 2, 3, 4)
k	The length of access credential	(256, 512, 1024)

TensorFlow, and other python libraries. The default simulation elements are shown in Table 3.

The simulations are executed in cycle-based manner. At each cycle, a certain percentage of SWs are randomly selected as dishonest SWs. The behavioral pattern of honest SW is modeled to always keep secret, while dishonest SWs may leak an uiv message sometimes. Without punishment, dishonest SWs will hinder the establishment of a trusted vulnerability message sharing environment. In this section, the performance and overload of the proposed UIV-TSP scheme are experimentally evaluated under various settings.

The simulation uses 2,000 SWs to generate 399,284 access record samples for training. Meanwhile, all the data generated are outputted to a.csv file. The proportion of benign access record samples to malicious access record samples in the training set is almost 1 : 1. Then, the data used are split to 70% training, and 30% validation.

After dividing the dataset, this paper uses the Adam optimization algorithm to obtain the optimal parameter combination, such as learning rate, epochs, and batch size. Then, the parameters obtained are used for feedforward neural network training, and the performance is evaluated by cross validation. Figure 7 shows the variation of model scores under different learning rates. The x-coordinate represents the learning rate, while the y-coordinate represents the model score. The shaded graph indicates the fluctuation range of the model score for this training. It can be seen that the training score and testing score reach the optimal value when the learning rate is 0.1, epoch = 10, and batch_size = 16.

In order to evaluate the performance of the trust mechanism based on FNN algorithm, we analyze four evaluation metrics, namely, precision (P), recall (R), accuracy (A), and F1-score ($F1$). In this paper, the precision and recall are defined as follows:

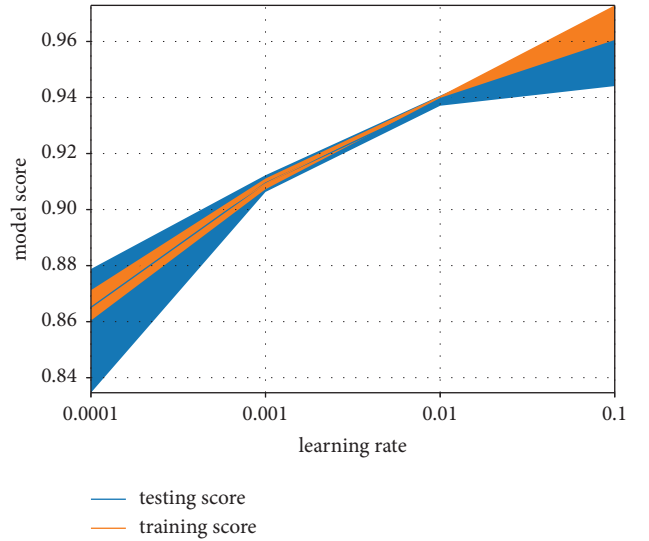


FIGURE 7: FNN learning rate score.

$$A = \frac{\text{number of truly SWs detected}}{\text{total number of SWs}} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$R = \frac{\text{number of truly malicious SWs detected}}{\text{total number of truly malicious SWs}} = \frac{TP}{TP + FN}$$

$$P = \frac{\text{number of truly malicious SWs detected}}{\text{total number of malicious SWs}} = \frac{TP}{TP + FP}$$

$$F1 = \frac{2 * P * R}{P + R} \quad (11)$$

We compare the performance of feedforward neural network (FNN) for identifying malicious SWs with two other well-known deep learning models, namely, recurrent

TABLE 4: Comparison of the performance of different algorithms.

		Accuracy	Recall	Precision	F1-score	Time (s)
FNN	Train	0.9751	0.9518	0.9546	0.9532	228
	Test	0.9736	0.9050	0.9142	0.9096	
CNN	Train	0.7428	0.8370	0.6486	0.7306	337
	Test	0.7422	0.8366	0.6481	0.7305	
RNN	Train	0.8804	0.8455	0.8447	0.8452	410
	Test	0.8780	0.8454	0.8447	0.8449	

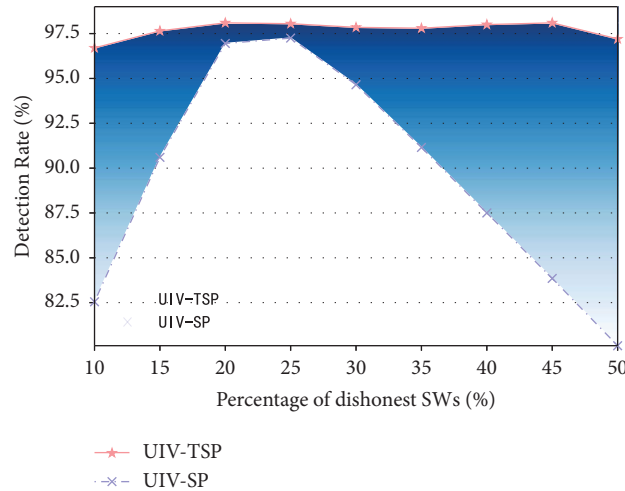


FIGURE 8: Detection rate under the percentage of dishonest SWs.

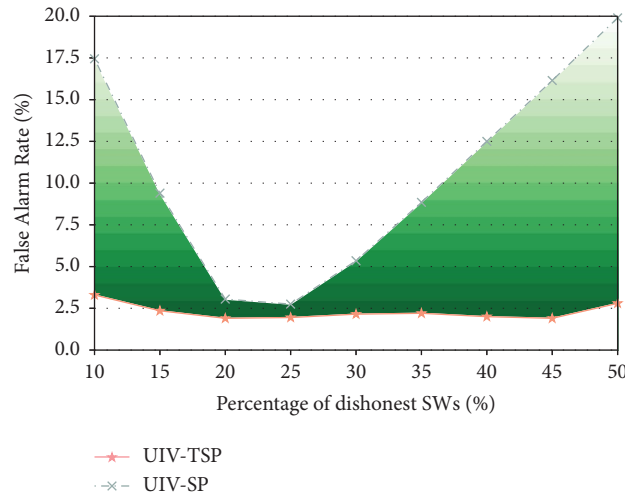


FIGURE 9: Alarm rate under the percentage of dishonest SWs.

neural network (RNN) and convolutional neural network (CNN). Table 4 shows that the FNN performs best in terms of the accuracy, recall, precision, and $F1$ -score. In addition, FNN clearly lower than the other algorithms in terms of the training time costs. That is because the FNN algorithm has obvious advantages in network structure, and owns simpler node connections than RNN. Due to the convolution operation, CNN will require more computation. Therefore, we conclude that FNN is a suitable deep learning algorithm for SW trust evaluation.

Then, we analyze the detection and false alarm rate of our UIV-TSP scheme (i.e., probability of successful detection leaker) in Figures 8 and 9. To analyze the effectiveness better, we compare UIV-TSP with the undisclosed IIoT vulnerabilities sharing protection (UIV-SP) scheme without trust mechanism.

In this simulation, the detection rate of UIV-TSP is better than UIV-SP in Figure 8, while the false alarm rate of UIV-TSP is lower than UIV-SP in Figure 9. Therefore, our designed trust mechanism can improve the performance of

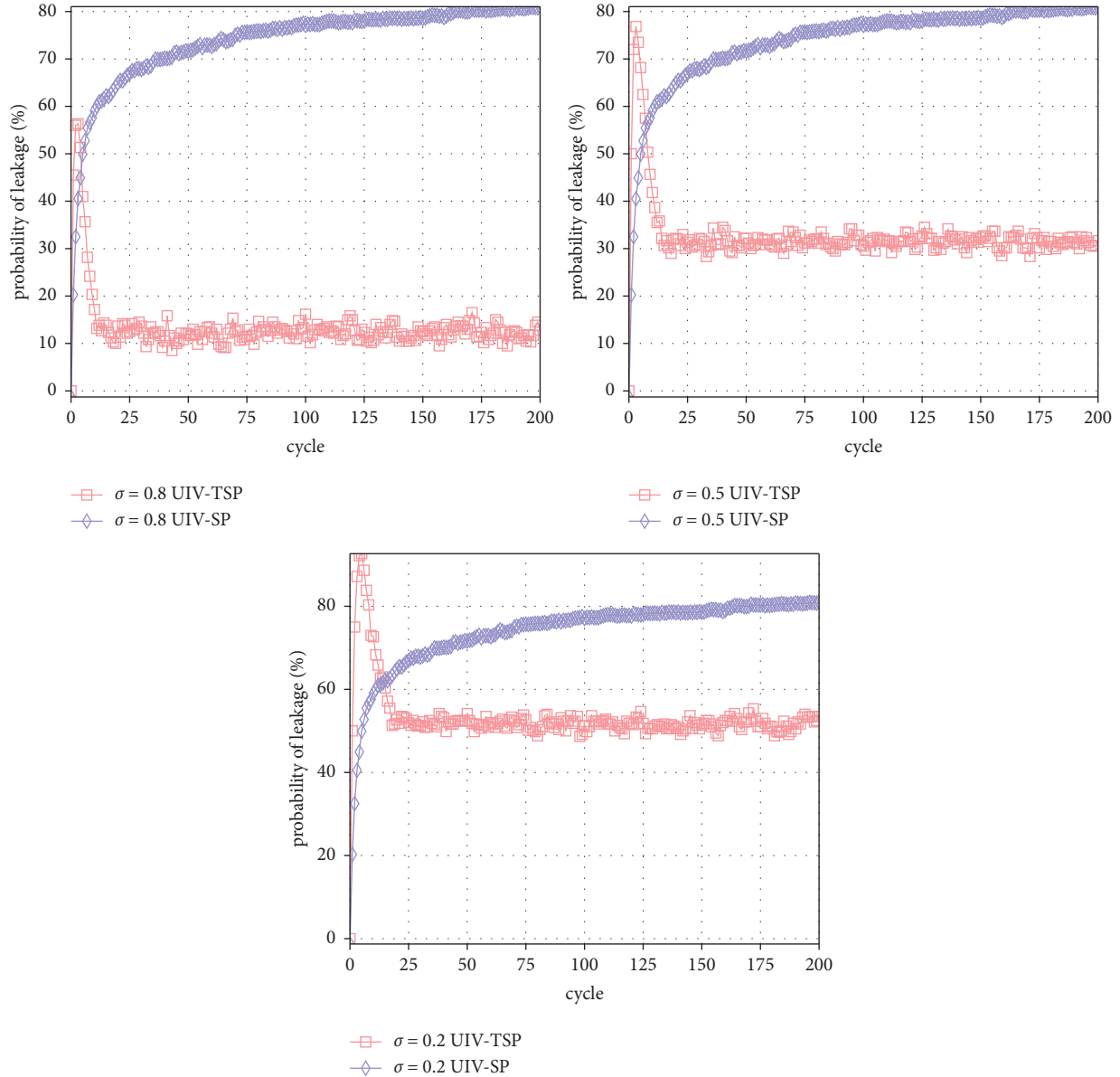


FIGURE 10: Suppressing leakage behaviors.

UIV-SP distinctly in the construction of trusted sharing environment.

Then, we validate the performance of our UIV-TSP scheme against dishonest SWs, in terms of suppressing leakage behaviors. Dishonest SWs will leak uiv in the sharing environment. Consequently, some leakage behaviors may generate in each cycle, which may cause unnecessary waste of network resources. So, the key performance indicator of UIV-TSP is to suppress these leakage behaviors. As shown in Figure 10, it is obvious that UIV-TSP is better than UIV-SP in suppressing leakage behaviors. This means that the trust mechanism plays a key role in the detection of dishonest SWs. In this simulation, δ is set as (0.2, 0.5, 0.8), respectively.

We also analyze the uiv leakage prevention performance of UIV-TSP in terms of leakage probability. In this

simulation, the percentage of dishonest SWs is set as 30%, respectively. As shown in Figure 11, the uiv leakage probability of UIV-TSP is also lower than UIV-SP.

Finally, we evaluate the traceability performance of UIV-TSP in terms of computational complexity. We count the number of time-consuming operations such as the symmetric-key encryption/decryption (SKE), public-key encryption/decryption (PKE), cryptographic hash function (HASH), and exponential operation (EXP) in G_q multiplicative operation (MUL) in G_q . As shown in Table 5, we compare UIV-TSP with three types of traditional schemes.

It can be found that UIV-TSP cannot require any exponential operation or public-key encryption/decryption. Moreover, the requirements for hash and symmetric key operations are limited in UIV-TSP.

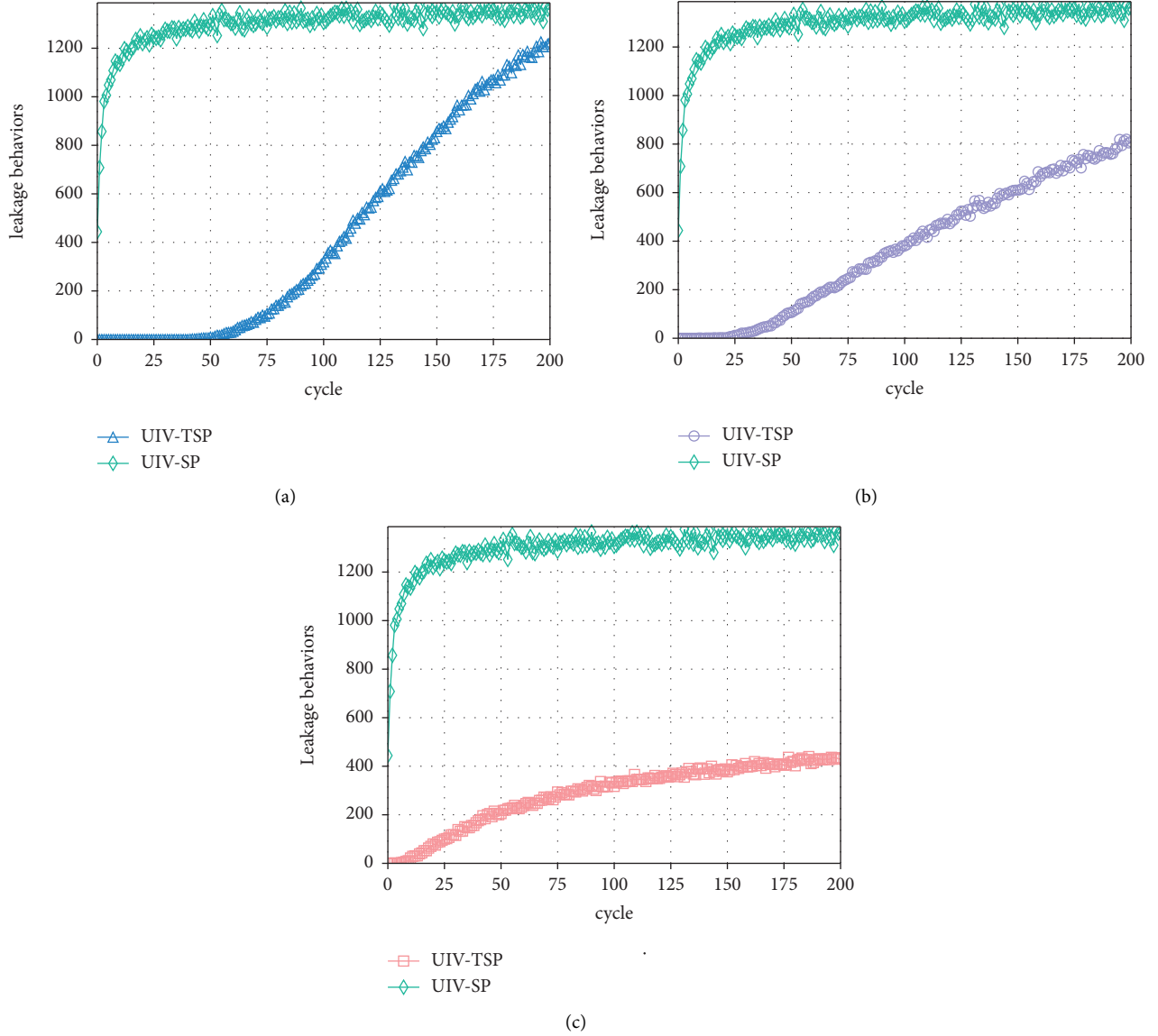


FIGURE 11: Probability of leakage at different δ . (a) $\delta = 0.8$. (b) $\delta = 0.5$. (c) $\delta = 0.2$.

TABLE 5: Computational complexity.

	SKE	HASH	PKE	EXP	MUL
CROT	$4t$	$8t$	$3t$	$6t + 3$	$2t + 1$
ADS	$2t$	$2t$	N/A	N/A	N/A
R-ADS	$2t$	$2t$	N/A	N/A	$4kt + 2t$
Ours work	N/A	$3t$	N/A	N/A	N/A

To further evaluate the traceability performance of UIV-TSP in terms of computational complexity, we can observe the traceability delay of UIV-TSP and these traditional schemes.

We run 200 rounds of experiments and obtain their average traceability delay as the result. We define k as the length of access credential. In our UIV-TSP scheme, the access credential is a dynamic token. In the traditional

schemes, the access credential is the private key of a user. A sufficient length of k can contribute to the collision resistance generated by hashing. As the length of k increases, the user capacity will be improved. Of course, with the increase of k and the embedded times of access credential, the traceability delay grows as well. As shown in Figure 12, UIV-TSP is more computationally efficient than ADS and CROT. The reason is that the sharing data in UIV-TSP need not to

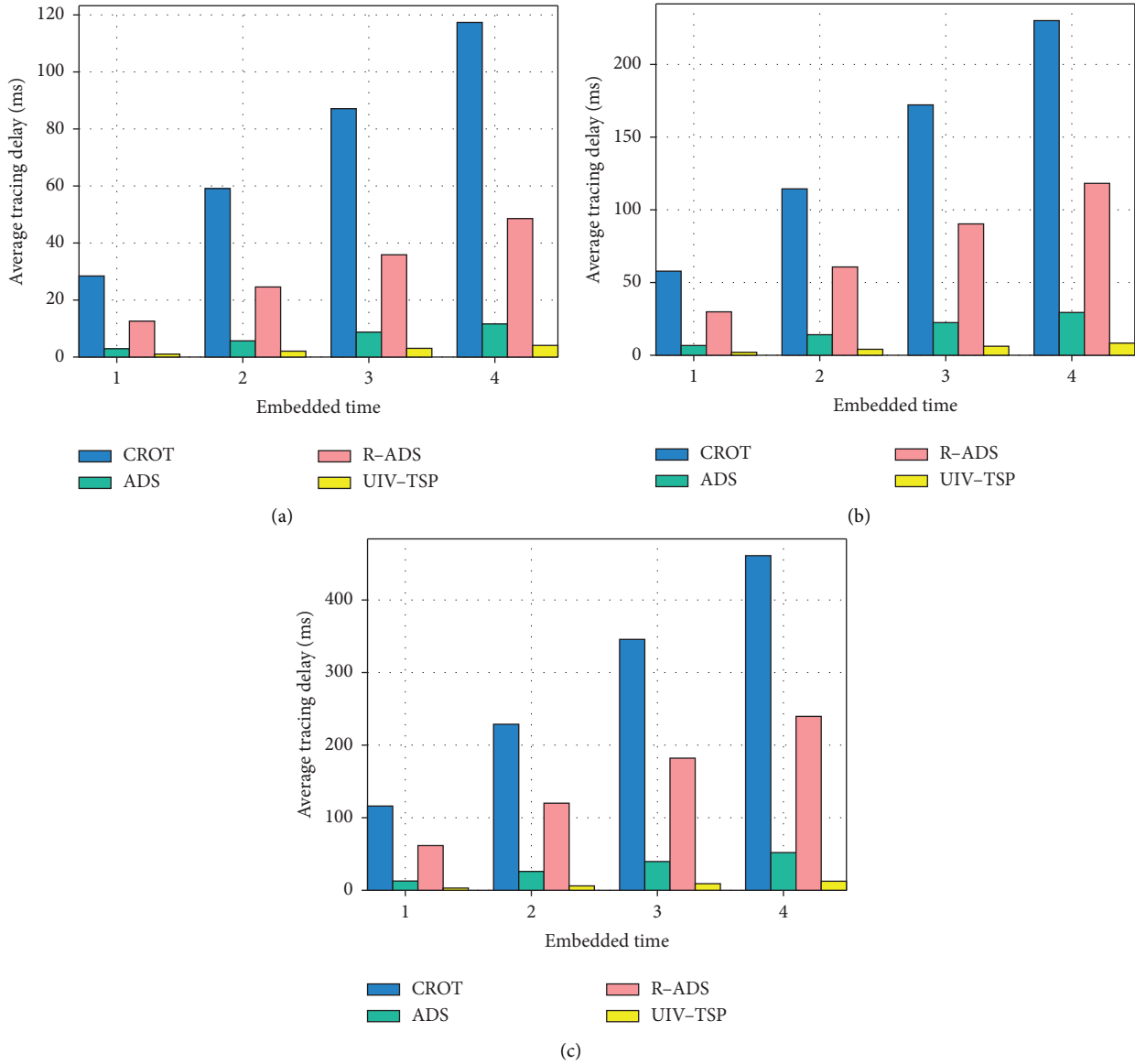


FIGURE 12: Average tracing delay. (a) $k=256$. (b) $k=512$. (c) $k=1024$.

perform oblivious transfer (OT) and zero-knowledge proof. Once the embedded times vary from 1 to 4, the number of OT increases linearly.

In summary, our UIV-TSP scheme can prevent the uiv information leakage effectively, which merely requires limited traceability delay caused by multiple shared SWs.

5. Industrial Applications Discussion

Since the IIoT vendors generally have weak security emergency response capabilities, some SWs can be invited to help them path a new uiv by means of CVD. Our UIV-TSP scheme can prevent the uiv leakage effectively in CVD and trace the dishonest SWs with limited traceability delay. As shown in Figure 13, UIV-TSP can be applied to several IIoT scenarios, such as energy, logistics, manufacturing, and transportation.

Take the IIoT manufacturing as an example. Once an IIoT manufacturing vendor reports a new uiv in CVD, he can select several SWs. Then, TA will assign $\text{token}_{\text{access}}$ to each of them and store $\text{token}_{\text{access}}$ on the blockchain. Without the implicit access credential, the unselected SWs cannot acquire uiv. With the implicit access credential, a selected SW can only acquire uiv on the basis of his high trust value. In this way, the uiv sharing can be restricted within the scope of permission, and the leakage problem of dishonest SWs can be avoided in advance.

In our UIV-TSP scheme, the metadata of uiv and the related SW access records are also stored on the blockchain, which can make the access logs of the uiv sharing as tamper-resistant. After the access, $\text{token}_{\text{tracing}}$ and $\text{code}_{\text{preleak}}$ can be stealthily sneaked into the acquired uiv information. Once the uiv information is one step away from the selected SW host, $\text{code}_{\text{preleak}}$ will destroy the uiv information and sends

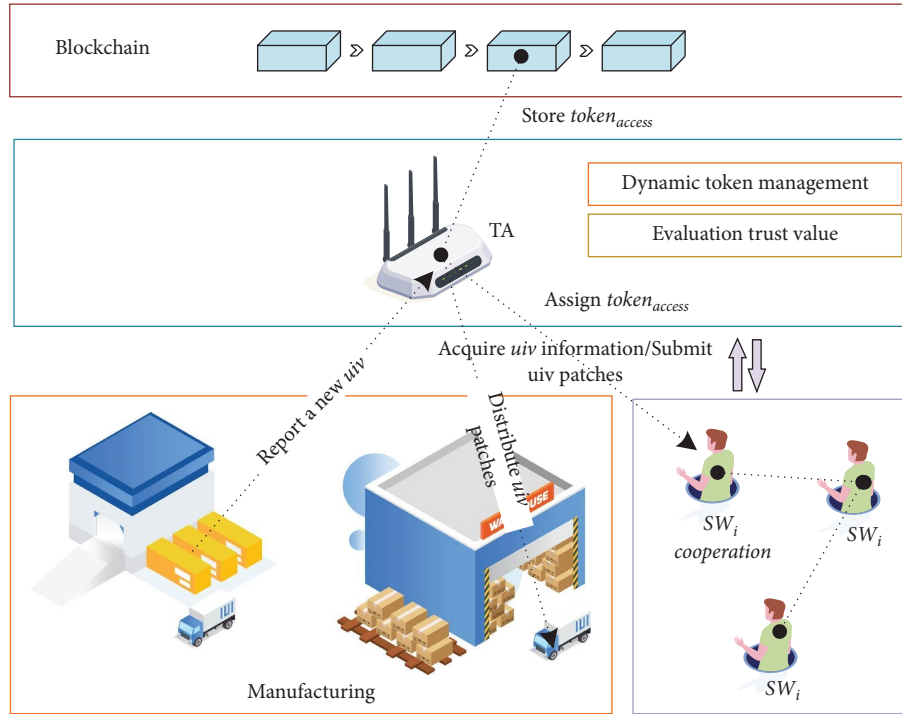


FIGURE 13: Industrial application case of UIV-TSP.

back feedback containing the token, thus avoiding a widespread damage to the users of the IIoT manufacturing devices after the uiv leakage. When the mitigation measures are developed, the uiv patch will be accurately distributed to the target IIoT manufacturing devices. Under the circumstances, uiv can be made public.

6. Conclusion and Future Works

In this paper, we propose an undisclosed IIoT vulnerabilities trusted sharing protection scheme against internal leakage. To facilitate the detection of leakage behaviors, we design a dynamic token as the implicit access credential and traceability clue. Assisted by blockchain, the continuous access logs of SWs can be securely stored. To prevent the leakage of a vulnerability, we present a benign logic bomb called $\text{code}_{\text{preleak}}$, which is embedded into the undisclosed IIoT vulnerability information. A trust management system based on deep learning is adopted to evaluate the trust value of SWs, which can quickly distinguish SWs. Simulation results indicate that our proposed scheme is resilient to suppress dishonest SWs, and merely require limited traceability delay.

For future works, we will investigate on the selfish SWs and motivate them to develop the mitigation measures of undisclosed IIoT vulnerabilities under the protection of UIV-TSP.

Data Availability

The data required for simulation are generated through experiments.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The study was supported by the National Natural Science Foundation of China (No. 61802302) and the Natural Science Basic Research Program of Shaanxi (No. 2019JM-442).

References

- [1] J. Gui, L. Hui, N. N. Xiong, and J. Wu, "Improving spectrum efficiency of cell-edge devices by incentive architecture applications with dynamic charging," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 795–808, 2021.
- [2] D. Cecchinato, T. Erseghe, and M. Rossi, "Elastic and predictive allocation of computing tasks in energy harvesting IoT edge networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1772–1788, 2021.
- [3] F. Shamieh, X. Wang, and A. R. Hussein, "Transaction throughput provisioning technique for blockchain-based industrial IoT networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 3122–3134, 2020.
- [4] W. Lu, S. Hu, X. Liu, C. He, and Y. Gong, "Incentive mechanism based cooperative spectrum sharing for OFDM cognitive IoT network," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 662–672, 2020.
- [5] R. Zhou, X. Wang, J. Wan, and N. Xiong, "EDM-fuzzy: an euclidean distance based multiscale fuzzy entropy technology for diagnosing faults of industrial systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4046–4054, 2021.

- [6] L. D. Xu, W. He, and S. Li, "Internet of Things in industries: a survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, Nov 2014.
- [7] Z. Zhou, C. Zhang, C. Xu, F. Xiong, Y. Zhang, and T. Umer, "Energy-efficient industrial Internet of UAVs for power line inspection in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2705–2714, 2018.
- [8] C. Zhang, G. Zhou, H. Li, and Y. Cao, "Manufacturing blockchain of Things for the configuration of a data- and knowledge-driven digital twin manufacturing cell," *IEEE Internet of Things Journal*, vol. 7, no. 12, pp. 11884–11894, 2020.
- [9] Y. Zhang, Z. Guo, J. Lv, and Y. Liu, "A framework for smart production-logistics systems based on CPS and industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4019–4032, 2018.
- [10] J. Feng, Y. Wang, J. Wang, and F. Ren, "Blockchain-based data management and edge-assisted trusted cloaking area construction for location privacy protection in vehicular networks," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2087–2101, 2021.
- [11] J. M. Mcginthy and A. J. Michaels, "Secure industrial Internet of Things critical infrastructure node design," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8021–8037, Oct 2019.
- [12] Y. Shah and S. Sengupta, "A survey on classification of cyber-attacks on IoT and IIoT devices," in *Proceedings of the 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0406–0413, New York, NY, USA, October 2020.
- [13] W. Zhao and G. White, "A Collaborative Information Sharing Framework for Community Cyber Security," in *Proceedings of the 2012 IEEE Conference on Technologies for Homeland Security (HST)*, pp. 457–462, Waltham, MA, USA, November 2012.
- [14] "ISO/IEC 29147:2018 Information technology -Security techniques- Vulnerability disclosure," 2018, <https://www.iso.org/standard/72311.html>.
- [15] D. Allen, "The CERT guide to coordinated vulnerability disclosure," 2019, <https://vuls.cert.org/conf luence/display/CVD/The+CERT+Guide+to+Coordinated+Vulnerability+Disclosure.2019-12-12>.
- [16] I. Vakili, D. K. Tosh, and S. Sengupta, "Privacy-preserving cybersecurity information exchange mechanism," in *Proceedings of the 2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pp. 1–7, Seattle, WA, USA, July 2017.
- [17] S. Badsha, I. Vakili, and S. Sengupta, "Privacy preserving cyber threat information sharing and learning for cyber defense," in *Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0708–0714, Las Vegas, NV, USA, January 2019.
- [18] J. M. de Fuentes, L. Gonzalez-Manzano, J. Tapiador, and P. Peris-Lopez, "PRACIS: privacy-preserving and aggregatable cybersecurity information sharing," *Computers & Security*, vol. 69, pp. 127–141, 2017.
- [19] D. Homan, I. Shiel, and C. Thorpe, "A new network model for cyber threat intelligence sharing using blockchain technology," in *Proceedings of the 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–6, Canary Islands, Spain, June 2019.
- [20] D. Preuveneers, W. Joosen, J. Bernal Bernabe, and A. Skarmeta, "Distributed Security Framework for Reliable Threat Intelligence Sharing," *Security and Communication Networks*, vol. 2020, pp. 1–15, Article ID 8833765, 2020.
- [21] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6822–6834, Aug 2019.
- [22] V. Sharma, G. Choudhary, Y. Ko, and I. You, "Behavior and vulnerability assessment of drones-enabled industrial Internet of Things (IIoT)," *IEEE Access*, vol. 6, pp. 43368–43383, 2018.
- [23] F. Xiao, L.-T. Sha, Z.-P. Yuan, and R.-C. Wang, "VulHunter: a discovery for unknown bugs based on analysis for known patches in industry Internet of Things," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 267–279, 2020.
- [24] E. V. Mangipudi, K. Rao, J. Clark, and A. Kate, "Towards automatically penalizing multimedia breaches (extended abstract)," in *Proceedings of the 2019 IEEE European Symposium on Security and Privacy Workshops (EuroSec&PW)*, pp. 340–346, Stockholm, Sweden, June 2019.
- [25] C. Huang, D. Liu, J. Ni, R. Lu, and X. Shen, "Achieving accountable and efficient data sharing in industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1416–1427, 2021.
- [26] L. Y. Zhang, Y. Zheng, J. Weng, C. Wang, Z. Shan, and K. Ren, "You can access but you cannot leak: defending against illegal content redistribution in encrypted cloud media center," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 6, pp. 1218–1231, 2020.
- [27] J. Ning, Z. Cao, X. Dong, and L. Wei, "White-box traceable CP-abe for cloud storage service: how to catch people leaking their access credentials effectively," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 883–897, 2018.
- [28] Y. Imine, A. Lounis, and A. Bouabdallah, "An accountable privacy-preserving scheme for public information sharing systems," *Computers & Security*, vol. 93, Article ID 101786, 2020.
- [29] W. Zhang, J. Zhang, Y. Shi, and J. Feng, "Blockchain-assisted undisclosed IIoT vulnerabilities trusted sharing protection with dynamic token," 2021, <https://arxiv.org/abs/2103.08908>.
- [30] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.
- [31] V. Buterin, "On Public and Private Blockchains," 2015, <https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains>.
- [32] Q. Zeng, L. Luo, Z. Qian et al., "Resilient user-side android application repackaging and tampering detection using cryptographically obfuscated logic bombs," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, pp. 2582–2600, 2021.
- [33] A. Jøfsang and R. Ismail, "The beta reputation system," *Proc. The 15th Bled Electronic Commerce Conference*, vol. 5, pp. 2502–2511, 2002.
- [34] C. Zhang, W. Li, Y. Luo, and Y. Hu, "AIT: an AI-enabled trust management system for vehicular networks using blockchain technology," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3157–3169, 2021.
- [35] H. Wang, Z. Zheng, S. Xie, H. N. Dai, and X. Chen, "Blockchain challenges and opportunities: a survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

Research Article

DeepGuard: Backdoor Attack Detection and Identification Schemes in Privacy-Preserving Deep Neural Networks

Congcong Chen ¹, Lifei Wei ², Lei Zhang ¹, Ya Peng ¹ and Jianting Ning ³

¹College of Information Technology, Shanghai Ocean University, Shanghai 201306, China

²College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China

³Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China

Correspondence should be addressed to Lifei Wei; lfwei@shou.edu.cn

Received 22 June 2022; Accepted 9 September 2022; Published 10 October 2022

Academic Editor: Jinguang Han

Copyright © 2022 Congcong Chen 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.

Deep neural networks (DNNs) have profoundly changed our lifeways in recent years. The cost of training a complicated DNN model is always overwhelming for most users with limited computation and storage resources. Consequently, an increasing number of people are considering to resort to a cloud for an outsourced DNN model training. However, the DNN models training process outsourced to the cloud faces privacy and security issues due to the semi-honest and malicious cloud environments. To preserve the privacy of the data and the parameters in DNN models during the outsourced training and to detect whether the models are injected with backdoors, this paper presents DeepGuard, a framework of privacy-preserving backdoor detection and identification in an outsourced cloud environment for multi-participant computation. In particular, we design a privacy-preserving reverse engineering algorithm for recovering the triggers and detecting the backdoor attacks among three cooperative but non-collusion servers. Moreover, we propose a backdoor identification algorithm adapting to single-label and multi-label attack detection. Finally, extensive experiments on the prevailing datasets such as MNIST, SVHN, and GTSRB confirm the effectiveness and efficiency of backdoor detection and identification in a privacy-preserving DNN model.

1. Introduction

Deep neural networks (DNNs) have made outstanding achievements in many fields and the DNNs-based applications are profoundly changing the aspects of our lives, such as medical diagnosis [1], autonomous driving [2], and image processing [3]. Most of the DNN models are generally obtained by training or refining existing models. In order to obtain a more accurate DNN model, it is often necessary to use a large amount of data for the model training [4]. Due to the limited capability of personal computers in model training, it is difficult for individual users to complete DNN training on personal computers. Thus, the users always outsource the model training to a cloud. However, outsourcing the training process of DNN models to cloud servers also risks privacy and security issues. It is well known

that the training process of the DNN model consists of several steps, such as data collection, data processing, data training, storage, and use of model parameters. The more steps the training process of DNN models involves, the more significant privacy and security risks arise [5].

According to the privacy risks, the data may not be collected or used smoothly during data collection and data training due to sensitive data such as identity ID, health, property, and other information, or laws and regulations like the European Union General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA). In addition, the parameters of the trained DNN models are a great asset to the model owner after spending lots of time and money without privacy revealing [6]. According to the security risks, some recent studies have shown that the outsourced DNN models may be injected with backdoors

[7–9]. The DNN models injected with backdoors appear normal with a high probability to classify the clean data (i.e., no effect on classification accuracy). However, it misclassifies the poisoned data (clean data with triggers attached) as the prespecified label (target attack) or any incorrect label (untargeted attack). Backdoor attacks can be extremely damaging by identifying anyone as a specific person in the security guard systems. Moreover, backdoor attacks can also lead to significant risks in areas such as autonomous driving, voice recognition, and text recognition [10–12].

Many defense strategies have been proposed against backdoor attacks. There are two major types of defense strategies, model-based and data-based [9]. The former checks whether the model has been injected with a backdoor [13, 14] and the latter detects whether the input data has contained triggers [15]. These defense strategies have proven to be effective in detecting, identifying, and mitigating backdoor attacks in the plaintext domain, that is, directly in the data. However, these defense strategies may not work well in the ciphertext domain due to privacy-preserving requirements in the data and the DNN models. Secure multi-party computation (SMPC) seems to provide a possible solution for preserving data privacy, which originates from Yao’s millionaire problem [16], allowing multiple participants to cooperatively calculate arbitrary functions without revealing the private input of the participants. Much previous work has focused on this area such as SecureML [17], Chameleon [18], ABY3 [19], Falcon [20], MP-SPDZ [21]. However, most of the schemes focus on the computation’s privacy without considering backdoor defense strategies.

Therefore, there is a big gap between the backdoor attack in the plaintext and ciphertext domains. Figure 1 illustrates the scenarios between the backdoor attacks in different domains. In the plaintext domain, all computations are visible. While in the ciphertext domain, all computations are encrypted, this paper uses a three-participant secret sharing technique to preserve data and model privacy, where each participant holds a fragment of the data. Since the data is invisible, it brings many difficulties for backdoor defense.

Motivated by the above discussions, in this paper, we present, DeepGuard, a framework for privacy-preserving backdoor detection and identification in an outsourced cloud environment for multi-participant computation. We propose a model-based defense strategy that can detect backdoors in the ciphertext domain. In addition, most of the existing backdoor attack defense strategies are designed for single-label attacks and might be invalid for multi-label attacks. To detect these various attacks, we propose a novel backdoor identification algorithm. In summary, the contributions of this paper are summarized as follows:

- (i) A defense strategy for backdoor attacks that works effectively in the ciphertext domain. We propose a backdoor attack detection algorithm based on MP-SPDZ [21] and NC [13] that can work in the ciphertext domain, which performs detection of backdoor attacks and ensures that the privacy of training data and DNN models are not compromised.

- (ii) A backdoor identification algorithm for single-label and multi-label attacks. We propose a novel identification algorithm for single-label and multi-label attacks based on the forementioned backdoor detection results, which can effectively identify the specific attacked label.
- (iii) Evaluating the effectiveness and efficiency of the proposed schemes. We validate the effectiveness and efficiency of our proposed schemes against both single-label and multi-label attacks by conducting extensive experiments on DNNs and state-of-the-art attack methods on the prevailing datasets such as MNIST, SVHN, and GTSRB.

2. Related Work

2.1. Backdoor Attacks. To the best of our knowledge, two advanced backdoor attack approaches are widely used to inject backdoors into target models. These two backdoor attack schemes are BadNets [7] and Trojan Attack [11]. Gu et al. [7] proposed BadNets to inject backdoor attacks by poisoning a training dataset. BadNets constructs a poisoned training dataset by adding triggers to randomly selected clean data and modifying its label to the target label. The Trojan Attack proposed by Liu et al. [11] differs from BadNets, i.e., the Trojan Attack does not allow the attacker to access clean datasets but can access pretrained DNN models. The Trojan Attack generates trojan triggers and training data by reverse engineering pretrained DNN models, and then uses the generated trojan triggers and training data to retrain the DNN model to inject the backdoor.

Besides, some more advanced backdoor attack approaches have been proposed in the past few years, Saha et al. [8] proposed a hidden random backdoor trigger injection technique. However, they required a larger trigger size to achieve a better attack effect. Gong et al. [9] proposed a backdoor attack approach that can resist four advanced defense strategies in an outsourced cloud environment. Bagdasaryan and Shmatikov [22] proposed blind backdoors that require neither access to the training data nor the model. Shokri [23] designed an adaptive adversarial training algorithm that optimizes the raw loss function of the model and maximizes indistinguishability of the poisoned data and clean data. Liu et al. [24] proposed a backdoor attack on DNN models with a high success rate by using mathematical modeling of the physical reflection model. Salem et al. [25] proposed a triggerless backdoor attack against deep neural networks based on the dropout technique, i.e., the attacker does not need to modify the input that triggers the backdoor. Yao et al. [26] consider backdoor attacks in transfer learning, in which all “student” models can inherit backdoors hidden in the “teacher” model, which poses a significant security threat.

2.2. Backdoor Defenses. Liu et al. [27] proposed the first effective defense scheme for DNN backdoor attacks, Fine-Pruning, which successfully defended against backdoor attacks using a combination of pruning and fine-tuning. In

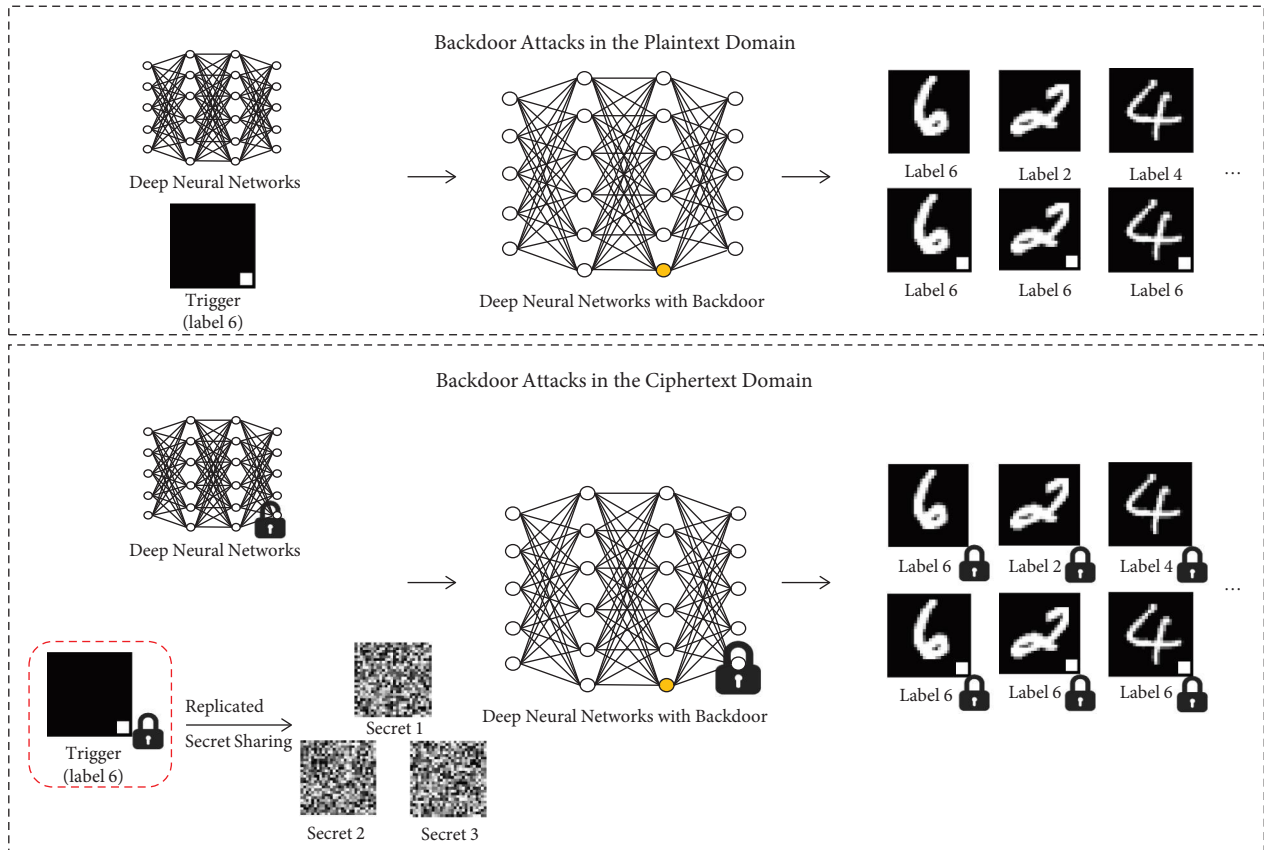


FIGURE 1: The difference between the backdoor attacks in the plaintext and ciphertext domains.

2019, Wang et al. [13] constructed triggers by reverse engineering methods and analyzed their outliers through the triggers and eventually pruned neurons based on this constructed trigger to reduce the success rate of backdoor attacks. Liu et al. [14] analyzed internal activation values by introducing different levels of stimulation to neurons to detect whether the DNN model is being attacked by a backdoor. Gao et al. [15] detected triggers by overlaying input samples and measuring the entropy distribution of output results. Chen et al. [28] used a conditional generative model to learn the probability distribution of potential triggers from the model to detect and defend against backdoor attacks. Du et al. [29] demonstrated the effectiveness of differential privacy for outlier detection and extended the technique to backdoor attack detection. Qiu et al. [30] applied a data augmentation policy to eliminate the effectiveness of backdoor attacks and an augmentation policy to preprocess the input samples to invalidate the triggers in the inference phase. Shen et al. [31] found that the defense complexity of existing methods is quadratic with the number of class labels. They propose a more efficient scheme that reduces the complexity of the defense method by the K -Arm optimization method, allowing to handle models with many classes. Wang et al. [32] found that the representations of authentic and poisoned data against the target class are embedded in different linear subspaces. Therefore, based on the coherence optimization problem, they proposed the PiDAn algorithm to detect backdoors.

2.3. *Privacy-Preserving Machine Learning.* Mohassel and Zhang [17] present SecureML, the first privacy-preserving machine learning training scheme in a two-party computation model. Chameleon [18] used a semi-honest third party to replace Beaver triples to reduce the amount of communication. In addition, they greatly improved the practicality and scalability of ABY [33]. ABY3 [19] proposes a new scheme in semi-honest and malicious environments that allows arithmetic sharing, Boolean sharing, and Yao sharing to be efficiently converted among the three cooperative and non-collusion participants. Zhang et al. [34] proposed DeepPAR and DeepDPA protocols to preserve the input privacy and model parameter privacy of models in deep learning. MP-SPDZ [21] proposes an SMPC framework that greatly simplifies the cost of comparing different protocols and security models. Its underlying cryptographic primitives include secret sharing, oblivious transfer, homomorphic encryption, and garbled circuits. Subsequently, a series of efficient approaches such as Falcon [20], ASTRA [35], FLASH [36], Trident [37], and ABY2 [38] have focused on privacy-preserving machine learning.

2.4. *Comparison with Other Defense Strategies.* A comparison with other defense strategies against backdoor attacks is shown in Table 1. FLGUARD [44] requires submodel clustering to exclude submodels with high attack rates, while Safe Learning [45] excludes anomalous submodels by user random combination. They are partially compliant with the white-box item.

TABLE 1: Comparison with other backdoor defense strategies.

Approaches	Privacy protection approach	Model access		Privacy		Backdoor detection type		Backdoor attack defense types	
		Black-box	White-box	Data privacy	Model privacy	Model-based	Data-based	Single-label attacks	Multi-label attacks
SentiNet [39]	None	●	○	○	◐	○	●	●	●
Strip [15]	None	●	○	○	◐	○	●	●	●
NIC [40]	None	○	●	○	○	○	●	●	–
AC [41]	None	○	●	○	○	●	○	●	–
NC [13]	None	●	○	○	◐	●	○	●	◐
ABS [14]	None	○	●	○	○	●	○	●	◐
DeepInspect [28]	None	●	○	◐	◐	●	○	●	◐
TABOR [42]	None	●	○	○	◐	●	○	●	●
Auror [43]	Differential privacy, Federal learning	●	○	●	●	○	●	●	–
FLGUARD [44]	Secret sharing, Federal learning	○	◐	●	●	●	○	●	●
SAFElearning [45]	Secret sharing, Federal learning	○	◐	●	●	●	○	●	–
CRFL [46]	Differential privacy, Federal learning	○	●	●	●	●	○	●	–
MP-BADNet [47]	Replicated secret sharing, SMPC	●	○	●	●	●	○	●	○
Ours	Replicated secret sharing, SMPC	●	○	●	●	●	○	●	●

Note. ‘●’ indicates that the item meets, ‘○’ indicates that the item does not meet, ‘◐’ indicates that the item partially meets, and ‘–’ means that the item has not been experimentally proven and it is not possible to determine whether the item meets. The model access indicates the level of access to the model parameters of the backdoor attack detection approach. The privacy indicates the level of privacy preservation of the backdoor attack detection approaches. The backdoor detection type indicates whether it is data-based or model-based, with the former detecting whether the data contains triggers and the latter detecting whether the model is injected with backdoors. And the backdoor attack defense types indicate whether the approach is resistant to single- or multi-label attacks.

DeepInspect [28] is used to invert a subset of the training data through the model inversion, which can be seen as partially preserving the privacy of the data. SentiNet [39], Strip [15], NC [13], DeepInspect [28], and TABOR [42] can be seen as partially preserving model privacy because they are black-box access. NC [13] may not succeed for multiple target labels. ABS [14] and DeepInspect [28] have shown that a small number of multi-label attack detection may fail. Auror [43] and CRFL [46] are considered for backdoor defense in federated learning but are not applicable in privacy-preserving outsourced machine learning. MP-BADNet [47] can resist backdoor attacks against privacy-preserving DNN models but only supports single-label attacks and cannot effectively defend against multi-label attacks. Besides, an additional secondary server is required to participate in the computation.

3. Overview

3.1. Key Ideas. It is known from the previous discussion that many backdoor defense strategies and privacy-preserving machine learning schemes have been proposed, respectively. To better explain the motivation for this paper and the challenges in the ciphertext domain, we give the answers to the following questions for further elaboration.

- (1) Why do we need to preserve privacy? Nowadays, the data is always distributed among different data

owners, such as different companies, organizations, and individuals, who need to train a model together for practical reasons. However, with the increasing privacy awareness, the data owners are reluctant to disclose their sensitive data or restrict sharing their sensitive data due to the laws and regulations (e.g., GDPR and CCPA). Moreover, these data owners often have limited computing and storage resources for economic reasons and prefer to outsource their models to cloud servers for training. As a result, the ownership and management of the data are separated. Meanwhile, cloud service providers are usually untrustworthy and try to find out private data. Therefore, how to use private data in the DNNs model training while keeping the data available and invisible has become an urgent problem.

- (2) Why choose SMPC? Generally speaking, the techniques such as homomorphic encryption (HE) and SMPC are always used to preserve the privacy of data and models in an outsourcing environment. However, HE is unsuitable for training DNN models due to its vast computational burdens. That is why this paper prefers to use the SMPC technique for the privacy preservation of data and models. Apart from smaller computation, SMPC is naturally suitable for multiuser participation scenarios.

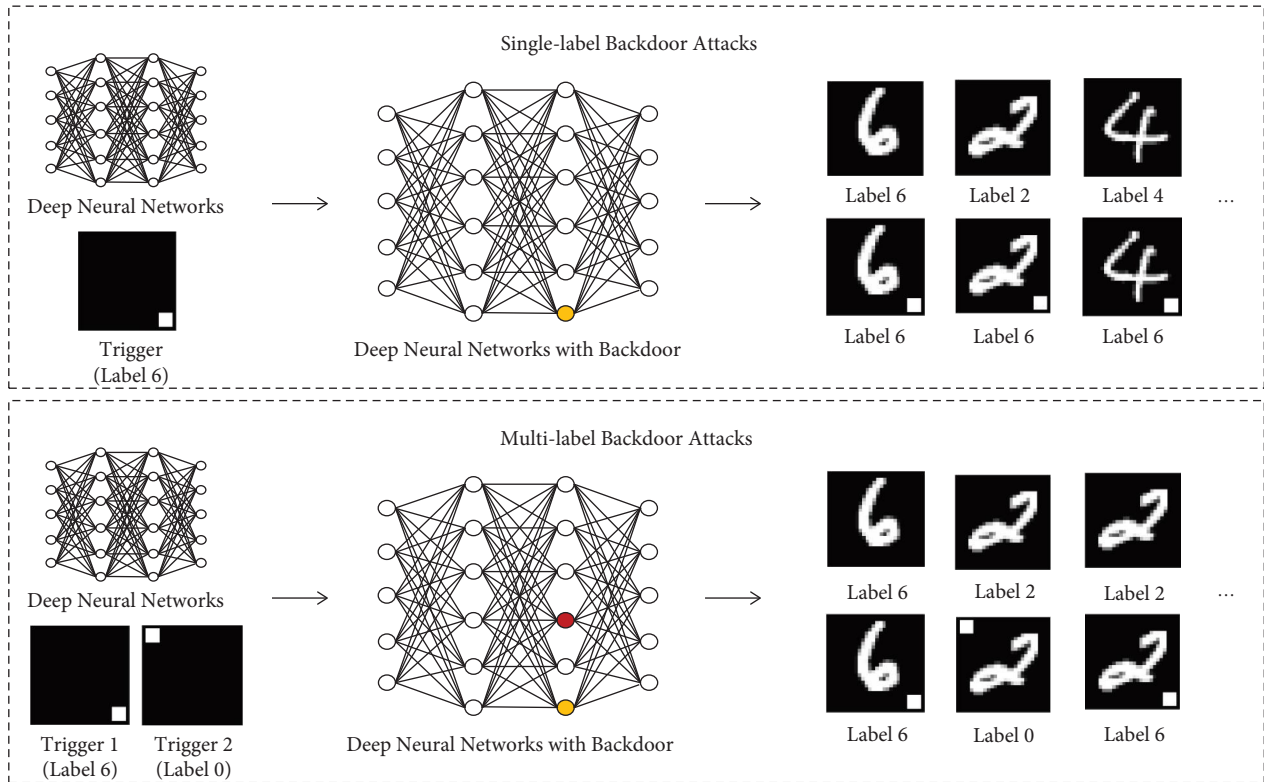


FIGURE 2: The single-label and multi-label backdoor attacks. The single-label attacks can classify arbitrary images into target labels. The multi-label attacks can hide multiple backdoors in the complicated DNN models, with different triggers corresponding to different target labels.

- (3) Why choose the SMPC technique with three participants? The SMPC technique between two participants is more suitable for two data owners to jointly train models without compromising privacy due to the underlying technology and is not suitable for outsourced computation scenarios. In addition, the time and communication overhead over three participants grows linearly with the number of participants. The SMPC technique with three participants is the best choice for our outsourced computing.

3.2. Threat Model. In this paper, we assume that three participants cooperatively train a DNN model using SMPC based on replicated secret sharing technique [48]. We consider the passive adversary model, which means that the corrupted participant is semi-honest and will not actively deviate from the protocol and launch a maliciously active attack but might try to snoop and obtain the private data from the other participants. It is a common adversarial setting considered in previous SMPC schemes [17–19]. Moreover, we assume that the participants do not collude since if the collusion appears, the colluding participants could recover private data through the fragments they hold.

For the backdoor attack, we assume that malicious users can modify their data before training the model. However, malicious users cannot modify other users’ data and cannot manipulate the DNN training process. The goal of the malicious user is to embed a backdoor in the DNNs model

when it is trained. A DNNs model with an embedded backdoor will output normally for clean data but will output malicious behavior specified by the malicious user for poisoned data. The defenders (the three participants in this paper) can access the data fragments and model fragments in the ciphertext domain, but not the complete data and model.

The experiments in [9] demonstrate that a certain ratio of data is sufficient for the trained DNN model to be backdoor attacked. Our basic attack model, named as single-label attacks, is consistent with [7, 13] in which it uses a white square as a trigger attached to the bottom right corner in a clean image to produce a poisoned data (the clean image is attached with a trigger) as shown in Figure 2. We believe that a successful backdoor attack does not affect the classification accuracy of the clean data, but it has a high probability to misclassify for the poisoned data. In the advanced attack model, named as multi-label attacks, we implement in this paper considering that the triggers appear at different locations in one image as shown in Figure 2.

3.3. Framework of DeepGuard. As shown in Figure 3, we assume that users want to outsource training the DNN model to the cloud servers due to the limitation of time or economic cost, and at the same time, do not want the cloud servers to learn the private data and parameters from the DNN model.

It is assumed that data are from different users or belongs to different data owners. Some of the data might be poisoned

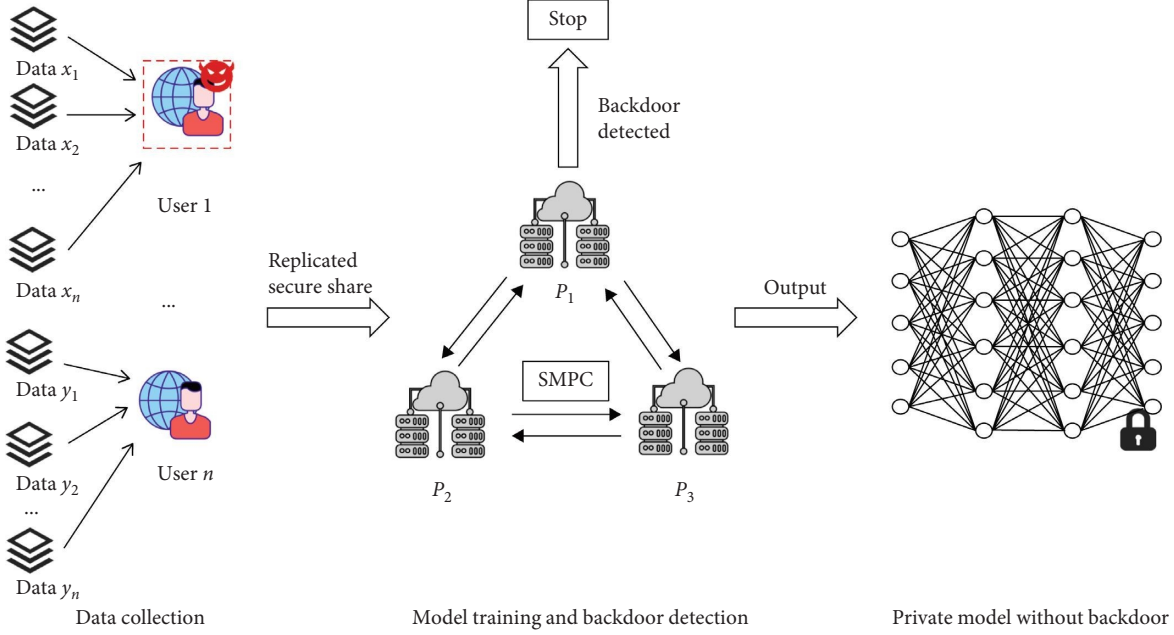


FIGURE 3: The framework of DeepGuard. The data are from different users or belong to different data owners. The triggers may be attached to the original data, which is trained in an SMPC-based DNN model through replicated secret sharing in three servers. When a backdoor attack is detected, the training is stopped, and the user is warned. Otherwise, a privacy-preserving DNN model is output.

TABLE 2: List of symbols.

Symbol	Description
Δ	When the target label corresponds to a backdoor, the minimum perturbation is required to map the other dimensional space to the target dimensional space.
δ	When the target label corresponds to a nonbackdoor, the minimum perturbation is required to map the other dimensional space to the target dimensional space.
π	The trigger was used to construct the poisoned data.
α	The attack threshold, i.e., the backdoor attack success rate, takes the value from 0 to 1.
η	The parameter MinSamples in the DBSCAN algorithm.
ϵ	The parameter epsilon in the DBSCAN algorithm.
β	The threshold is used to adjust the larger mean value.

in the source by malicious users and used in the model training to inject a backdoor into the model.

The cloud server consists of P_1 , P_2 , and P_3 , referred to as the participants in this paper. Each of these three participants holds the private data shares by the replicated secret sharing from the users and wants to collaborate to train a DNN model. In the trained DNN model, no specific parameters are accessible to any parties.

As the backdoored model has a high classification accuracy for clean datasets, it is difficult for the users to detect that the model is subject to a backdoor attack only by themselves. According to the privacy-preserving requirements, the honest users need to turn to all three participants simultaneously to perform the detection of the backdoor attack in the ciphertext domain to detect whether the DNN model is injected with a backdoor.

4. Concrete Construction

4.1. List of Symbols. Table 2 describes the symbols used in this section.

4.2. Overall Design. We define the poisoning data as follows:

$$\begin{aligned} \|x^*\| &= T(\|x\|, \|\text{mask}\|, \|\text{pattern}\|) \\ &= (1 - \|\text{mask}\|) \cdot \|x\| + \|\text{mask}\| \cdot \|\text{pattern}\|, \end{aligned} \quad (1)$$

where $\|\cdot\|$ denotes that the private data are held by three participants through replicated secret sharing technique [48] and thus single party is not able to access the private data. x^* denotes poisoned data, and $T(\cdot)$ denotes a function that attaches a trigger to the original image x . We use mask to denote a 2D matrix that determines the size and position of the trigger over the original image, which takes values in the range $(0, 1)$. We use pattern to represent the trigger pattern, which is a matrix of the same size as the original image. The parameters $\text{mask}_{i,j}$ and $\text{pattern}_{i,j}$ represent the mask and pattern values of row i and column j , respectively. When $\text{mask}_{i,j}$ equals to 1, it means that the value of row i and column j in the original image are completely covered by those row i and column j in the pattern. When $\text{mask}_{i,j}$ equals 0, it means that the original image is not covered at all.

As in [13], we consider the classification problem as the creation of partitions in a multidimensional space in ciphertext, where each dimension contains certain features, i.e., each dimension can be considered to represent a label. Embedding backdoors in DNN models is to create a “shortcut” from other dimensional spaces to the target dimensional space. Detecting backdoors in a DNN model can be thought of as detecting the minimum perturbation needed to get from the other dimensional space to the target dimensional space. If the target label corresponds to a backdoor, then let the minimum perturbation required to map the other dimensional space to the target dimensional space corresponding to the backdoor be Δ . If the target label corresponds to a no backdoor, then let the minimum perturbation required to map the other dimensional space to the corresponding target dimensional space be δ .

Let the trigger used to construct the poisoned data be π . Since the trigger needs to be as inconspicuous as possible (e.g., a white square in the corner of the image) in order to avoid being easily detected, it is natural to have that $\text{size}(\pi) \ll \text{size}(\delta)$. For single-label attacks, since Δ is the smallest perturbation, i.e., the corresponding valid part of the trigger, we can derive $\text{size}(\Delta) < \text{size}(\pi)$. For multi-label attacks, we can draw similar conclusions. Thus, we only need to analyze $\text{size}(\Delta)$ to know whether the model is attacked by a backdoor.

4.3. The Reverse Engineering Algorithm. To obtain Δ for each label, we designed an SMPC-based reverse engineering algorithm for obtaining triggers based on a clean dataset. There are three targets in this algorithm, that is, (1) to preserve data and model parameter privacy, (2) to find the shape of the trigger by backdoor attacks, i.e., (mask, pattern) that can misclassify other labels as the target label, and (3) to find the smallest possible trigger, i.e., to find the smallest (mask, pattern) that can produce a misclassification. To further achieve the third goal, we use the L_1 - norm of the mask to measure the size of the potential trigger. The objective function can be defined formally as follows:

$$h(\|x\|) = \min_{(\|\text{mask}\|, \|\text{pattern}\|)} L(\|y\|), f(\|x\|), (\|\text{mask}\|, \|\text{pattern}\|) + \text{cost} \cdot |\text{mask}|, \quad (2)$$

where $L(\cdot)$ denotes the loss function, which is the cross-entropy loss function used in this paper. y is the ground truth, $f(\cdot)$ denotes the DNN model, and cost denotes the coefficient of the third objective of the optimization. To speed up the convergence of reverse engineering, we use the Adam Optimizer [49] to obtain the optimal mask and pattern.

Algorithm 1 describes the process of recovering potential triggers. Step 1 is the initialization operation. Steps 2–18 are to reverse the model using clean dataset to obtain mask and pattern. Steps 4–8 are executed by splitting the clean dataset into several batches, which number is *miniBatch*. Step 5 is to construct the poisoned data, and step 6 is to take the poisoning data into the model to calculate the loss and the accuracy. Step 10 is to update the learning rate of the Adam

optimizer, and steps 12–15 to determine whether it achieves the optimal for (mask, pattern). Step 16 detects whether the algorithm stops, which is determined by the accuracy of the algorithm reaches the threshold several times.

In Algorithm 1, $\|\cdot\|$ indicates that the value is in the ciphertext domain. When α is set to 1, it means that the trigger is capable to cause all clean images to be misclassified as target label. α affects the effectiveness of the backdoor attack and the final size of the recovered trigger. *lossBest* denotes the optimal value of equation (2). When the trigger is recovered for a potential label, the value *cost* is first initialized to 0. If the recovered trigger causes other labels to be classified as the potential label with a success rate higher than the threshold α , the *cost* is adjusted upward to reduce the size of the trigger. Otherwise, the *cost* value is adjusted downward to enlarge the size of the trigger. In short, the larger *cost*, the lower the success rate of the attack and the smaller the trigger. The algorithm stops until the success rate of classifying other labels as potential labels are relatively stable. In this paper, the values mask and pattern are initialized with random values, and then optimized according to equation (2). To enable the reverse engineering algorithm to accommodate a certain amount of error, α is set to 0.99. *lossBest* is initialized to infinity because the loss needs to be updated according to the calculation of Equation (2).

4.4. The Backdoor Identification Algorithm. In backdoor identification, we analyze the size of the trigger by the L_1 - norm of mask. The idea is to use an efficient clustering algorithm to classify the labels into two categories: benign and poisoned, since the potential triggers corresponding to the labels might be benign or poisoned. Naturally, we first try to use the K -means clustering algorithm for classification. It was found through experiments that the K -means clustering algorithm could not identify clean models (i.e., only one class of benign labels) and could not correctly classify scenarios in which the sizes of the triggers corresponding to the labels differed significantly. Since the recovered trigger sizes are relatively dense, we consider using the density-based classification algorithm. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [50] is one of the typical algorithms based on density clustering. The DBSCAN algorithm assumes that the closeness of the sample distribution determines the categories and does not require prespecifying the cluster size for clustering, which performs better than K -means in backdoor identification.

Since the value mask computed by the reverse engineering algorithm is relatively dense, we chose to use the DBSCAN algorithm for outlier analysis. The parameters (epsilon, MinSamples) in the DBSCAN algorithm describe the closeness of the sample distribution in the neighborhood. Where epsilon describes the neighborhood distance threshold for a given sample and MinSamples describes the threshold for the number of samples in the neighborhood for a given sample with distance epsilon. Thus, it is crucial to determine the parameters (epsilon, MinSamples). In the

```

Input:  $\|x\|, \|\text{target}\|$ 
Output:  $\min(\|\text{mask}\|, \|\text{pattern}\|)$ 
(1) Initialize  $\|\text{mask}\|, \|\text{pattern}\|$ , let  $\alpha = 0.99$ ,  $\text{lossBest} = \infty$ .
(2) while 1 do
(3)   Initialize LossList = [], LossAcc = [];
(4)   for  $i$  in mini Batch do
(5)      $\|\text{TrojanInput}_i\| = (1 - \|\text{mask}\|) \times \|x_i\| + \|\text{mask}\| \times \|\text{pattern}\|$ ;
(6)     Calculate  $\|\text{loss}\|$  and  $\|\text{acc}\|$  based on  $\|\text{target}\|$  and  $\|\text{TrojanInput}_i\|$ ;
(7)     Update  $\|\text{mask}\|, \|\text{pattern}\|$ ;
(8)     LossList.append ( $\|\text{loss}\|$ ), Loss Acc.append ( $\|\text{acc}\|$ );
(9)   end for
(10)  Update Adam learning rate  $LR$ ;
(11)   $\|\text{accAvg}\| = \text{mean}(\text{LossAcc})$ ,  $\|\text{lossAvg}\| = \text{mean}(\text{LossList})$ ;
(12)  if ( $\|\text{accAvg}\| > \alpha$ ) and ( $\|\text{lossAvg}\| < \text{lossBest}$ ) then
(13)     $\min(\|\text{mask}\|, \|\text{pattern}\|) = (\|\text{mask}\|, \|\text{pattern}\|)$ ;
(14)     $\text{lossBest} = \|\text{lossAvg}\|$ ;
(15)  end if
(16)  Check whether the algorithm ends early;
(17)  Update cost;
(18) end while
(19) Return  $\min(\|\text{mask}\|, \|\text{pattern}\|)$ .

```

ALGORITHM 1: Reverse engineering algorithm.

```

Input: List Mask, Labels
Output: Targetlabel
(1) Initialize  $\epsilon, \eta, \beta = 4 * \epsilon$ , TargetLabel = [];
(2) Calculate mean = mean(ListMask);
(3) if mean  $> \beta$  then
(4)   for  $i$  in Labels do
(5)     while ListMask[ $i$ ]/ $\beta > 1.1$  do
(6)       ListMask[ $i$ ] = List Mask[ $i$ ]/1.1;
(7)     end while
(8)   end for
(9) else
(10)  for  $i$  in Labels do
(11)    while ListMask[ $i$ ]  $> \beta$  do
(12)      List Mask[ $i$ ] = ListMask[ $i$ ]/1.1;
(13)    end while
(14)  end for
(15) end if
(16) outlier_detection = DBSCAN(min_samples =  $\eta$ , eps =  $\epsilon$ );
(17) clusters = outlier_detection.fit_predict (ListMask);
(18)  $a = [], b = []$ ;
(19) for  $i$  in Labels do
(20)   if clusters[ $i$ ] == 0 then
(21)      $a.append(i, \text{ListMask}[i])$ ;
(22)   else
(23)      $b.append(i, \text{ListMask}[i])$ ;
(24)   end if
(25) end for
(26) if len( $a$ )  $> 0$  and len( $b$ )  $> 0$  then
(27)   TargetLabel = mean( $a$ )  $<$  mean( $b$ )?  $a$ :  $b$ ;
(28) end if
(29) Return TargetLabel.

```

ALGORITHM 2: Backdoor identification algorithm.

following, we use ϵ to denote epsilon and η to denote MinSamples for short.

Algorithm 2 is the backdoor identification algorithm. Step 1 is to initialize the parameters (ϵ, η) and the threshold β . Step 2 is to calculate the mean value of the mask. Steps 3–15 are to prevent larger mask values from being identified as outliers. If the mean value is greater than β , the values mask greater than the mean value are reduced to around the mean value to increase the density of the data. Steps 16–17 are used for clustering analysis using the DBSCAN algorithm. Steps 18–25 correspond to the clustering result analysis. The clustering results are generally classified into $(-1, 0)$, $(0, 1)$, and $(-1, 0, 1)$, where -1 represents the anomaly, 0 represents the first category of clustering results, and 1 represents the second category of clustering results. When the backdoor attack is a kind of single-label attack or multi-label attack with a relatively small number of labels, the classification result is generally $(-1, 0)$. When the multi-label attacks have a large number of labels, the classification result is generally $(0, 1)$ or $(-1, 0, 1)$. Therefore, in steps 26–28, we analyze the specific attacked labels by comparing the mean of the two categories of classification results.

4.5. Security Analysis

4.5.1. Channel Security. We assume that a secure channel is established among the three participants by each other to ensure that the transmitted information is not corrupted. The communication private key in this secure channel is securely stored and cannot be easily disclosed.

4.5.2. Privacy-Preserving. In this scheme, we assume that the participants (i.e., the cloud servers providing computing services) are semi-honest, and do not actively deviate from the protocol. Different users or data owners provide the datasets used for model training and backdoor detection. They send the datasets to the participants via replicated secret sharing technique. Eventually, each participant only holds fragments and is assumed not to collude with others to recover the data. Therefore, the different users or data owners cannot have access to the private data of other users and the participants cannot have access to the complete private data. Moreover, the privacy-preserving DNN model is trained by the interaction between participants using replicated secret sharing technique, and does not reveal the privacy of the model parameters.

5. Experiment and Evaluation

We evaluate the effectiveness and efficiency of the proposed framework by training a DNN model and embedding two types of backdoor attacks (single-label and multi-label attacks). The experiment performed all the benchmarks on a server with two 16-core 2.10 GHz Intel(R) Xeon(R) Gold 6130 CPUs, 256 GB RAM, and Ubuntu 16.04.4 LTS.

As shown in Figure 4, in the single-label attacks scenario, a trigger is targeted at one label and thus there exists only one trigger in the DNN model. The model makes classification

normally for a dataset with all clean data. However, for the poisoned data (those data with a trigger), the model always classifies it to the prespecified label regardless of the original label. In the multi-label attacks, a DNN model is attacked by a backdoor containing multiple attacked labels representing the backdoor attacks against different labels at different locations. The trigger is squares of 3×3 pixels or 4×4 pixels. To make the multi-label attacks effective, the poisoning rate of each poisoned label is about 25%. The following equation defines the proportion R_{poison} of the total poisoned data to the dataset.

$$R_{\text{poison}} = \frac{R_{\text{label}} \times N_{\text{label}}}{R_{\text{label}} \times N_{\text{label}} + 1}, \quad (3)$$

where R_{label} denotes the poisoning rate of each poisoning label and N_{label} denotes the number of poisoning labels. Based on the above settings, the steps of each epoch when training the DNN model is as follows:

$$S_{\text{train}} = \frac{1}{1 - R_{\text{poison}}} \times |D|, \quad (4)$$

where S_{train} denotes the number of training steps per epoch during the DNN model training and $|D|$ denotes the size of the dataset.

5.1. Backdoor Detection in the Ciphertext Domain

5.1.1. Experimental Setup. The experimental is built on the primitive provided by the MP-SPDZ [21] library, and all arithmetic sharing of secret data is performed on modulo 2^{128} . For the ciphertext, we use three different processes to simulate three participants, and they perform calculations through the replicated secret sharing technique. In this section, we only consider using the MNIST dataset [51] and single-label attacks (supposing that the target label is 6). The architecture of the DNN model we evaluated is consistent with [17], which is a simple network consisting of 3 fully connected layers with a Rectified Linear Unit (ReLU) activation function. Finally, the softmax function is used to calculate the output probability of the labels. During training, the basic learning rate is set to 0.1, the number of epochs is set to 15, the batch size is 128, and the stochastic gradient descent (SGD) momentum is set to 0.9. The accuracy of the clean model is approximately 97%, both in the ciphertext and plaintext domains. Furthermore, in this network structure, we use triggers that are 4×4 pixel squares, as shown in Figure 4(a).

5.1.2. The Effectiveness of Backdoor Detection. Figure 5 shows the trigger sizes calculated by the reverse engineering algorithm from the different models, such as (a) the clean model and (b) the injected backdoor model, with Plaintext and Ciphertext representing the trigger sizes in plaintext (NC [13]) and ciphertext (ours), respectively. The sizes of the plaintext and ciphertext datasets used in our reverse

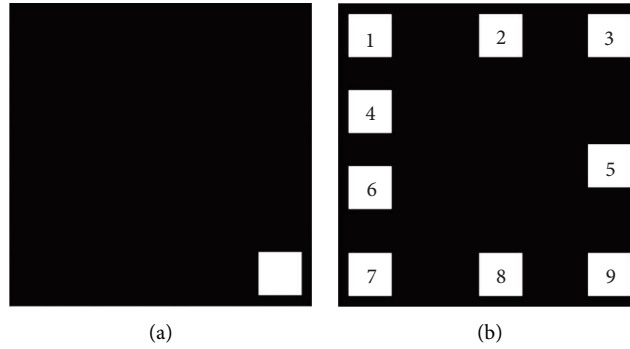


FIGURE 4: The trigger used in the paper. (a) Denotes the trigger in single-label attacks located at the bottom right corner and (b) denotes the possible attack locations of the trigger in multi-label attacks.

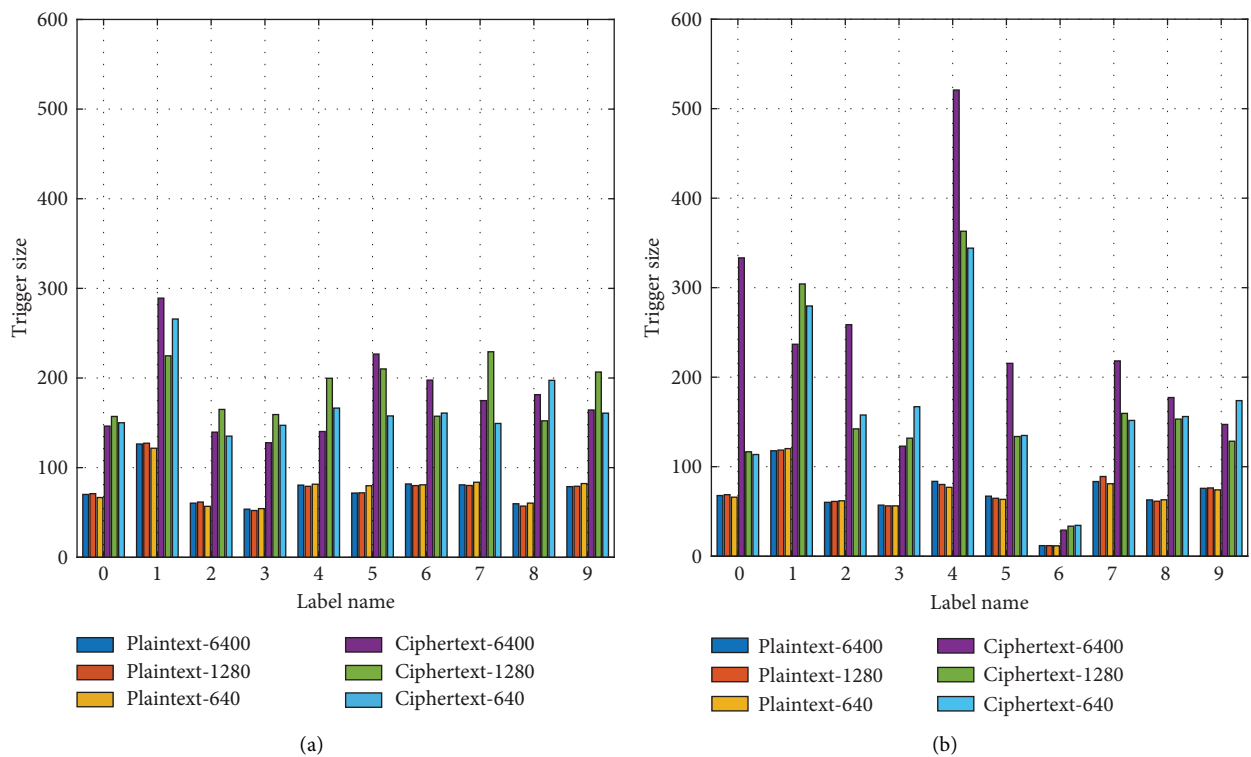


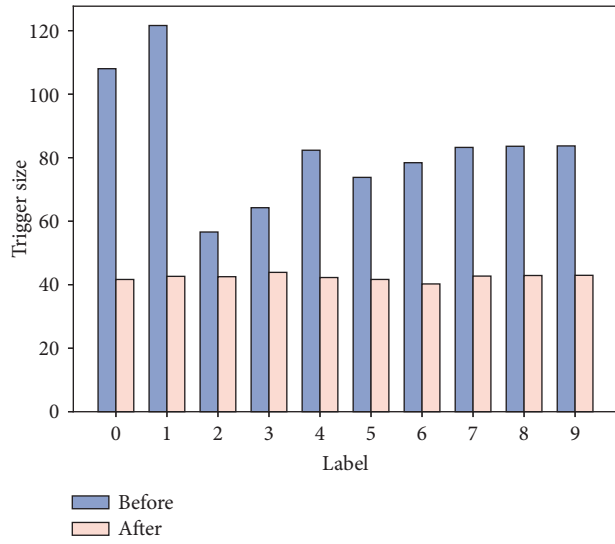
FIGURE 5: Trigger sizes calculated by reverse engineering algorithms. (a) Denotes the trigger size in the clean model and (b) denotes the trigger size in the poisoned model (assuming the target label is 6).

TABLE 3: The running time for reverse engineering algorithm in the ciphertext domain.

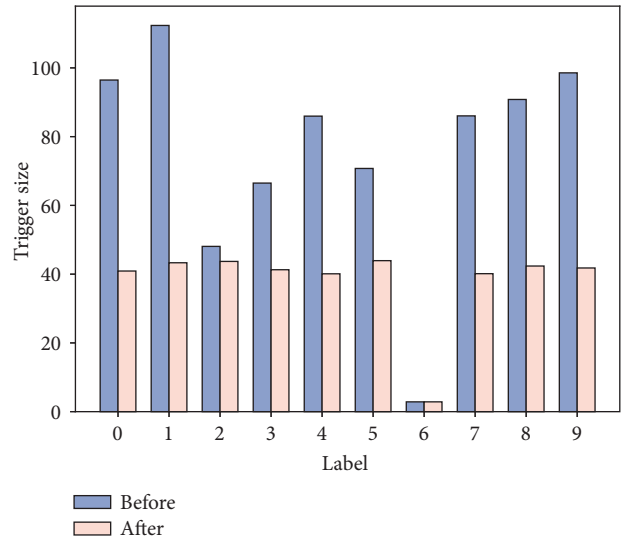
Time (s)	Reverse engineering					
	Clean model		Poisoned model		640	
	6400	1280	640	6400	1280	640
Average	27952.00	6041.90	3015.50	31886.80	6638.60	2896.40

engineering experiments are 6400, 1280, and 640, respectively. In this section, we only show the results of single-label attacks (the target label is 6) for comparison. The experimental results show that the trigger size of label 6 can be

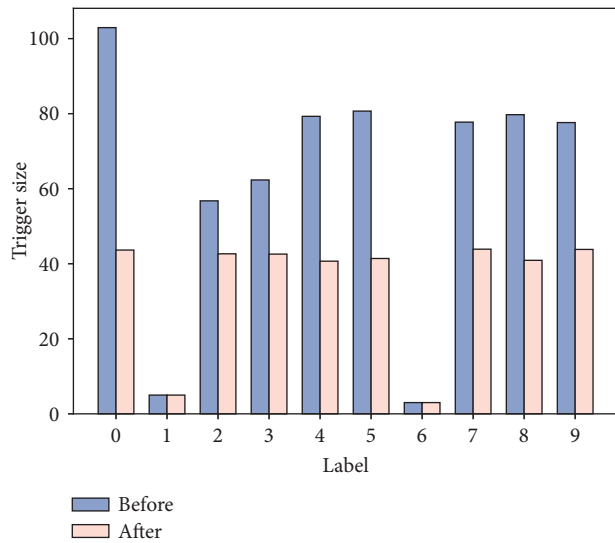
found to be significantly smaller than the other labels, although the difference between the trigger sizes calculated by the reverse engineering algorithm from the plaintext and ciphertext is relatively large.



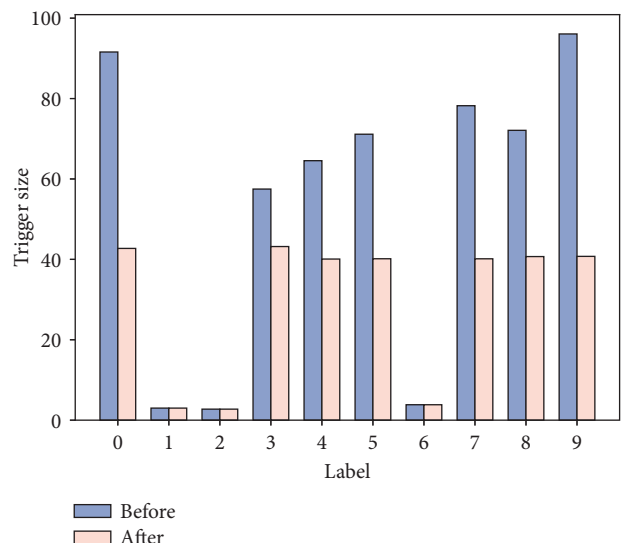
(a)



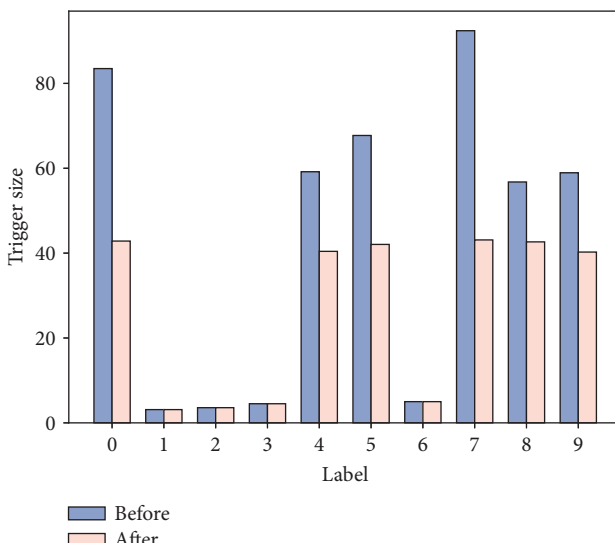
(b)



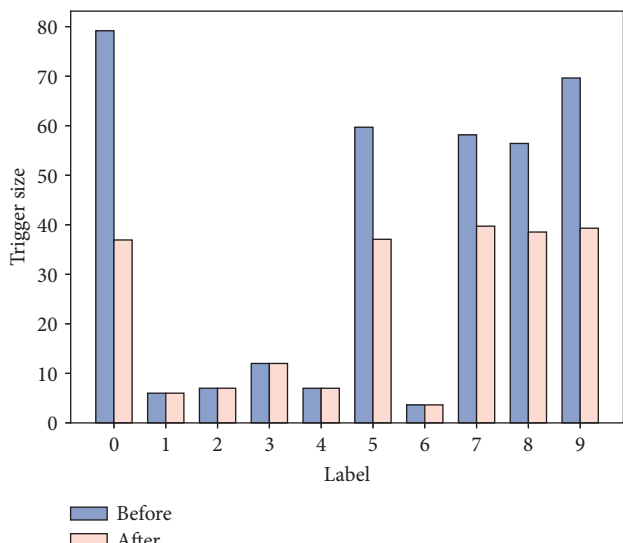
(c)



(d)



(e)



(f)

FIGURE 6: Continued.

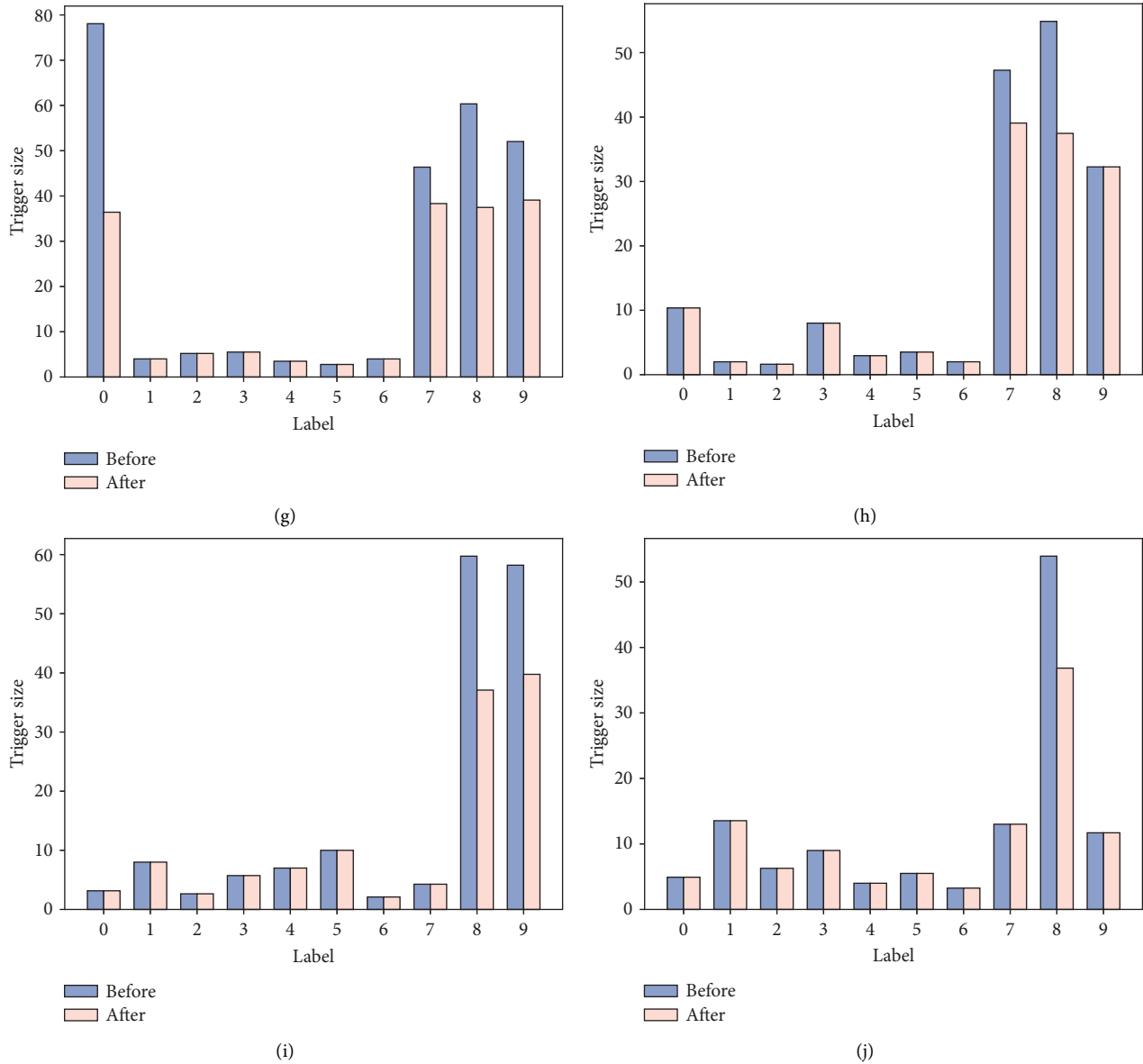
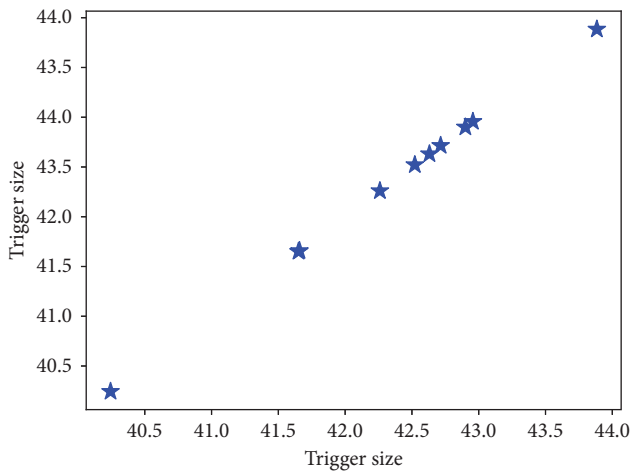


FIGURE 6: The trigger size before/after the backdoor identification algorithm on the MNIST dataset. (a) The number of attacked labels is 0. (b) The number of attacked labels is 1. (c) The number of attacked labels is 2. (d) The number of attacked labels is 3. (e) The number of attacked labels is 4. (f) The number of attacked labels is 5. (g) The number of attacked labels is 6. (h) The number of attacked labels is 7. (i) The number of attacked labels is 8. (j) The number of attacked labels is 9.

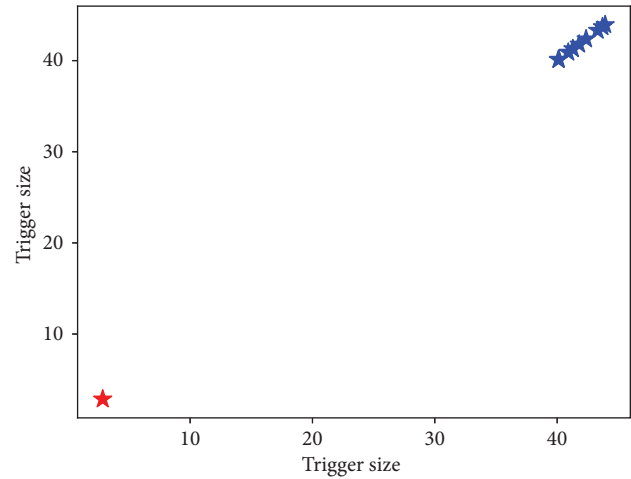
5.1.3. The Efficiency of Backdoor Detection. Table 3 compares the time cost of the reverse engineering algorithm in the ciphertext domain with different clean MNIST dataset sizes 6400, 1280, and 640 for reverse engineering, respectively. Compared with the plaintext [13], the time cost of performing reverse engineering algorithms in the ciphertext domain is greater according to the privacy requirement. Moreover, there is a significant communication cost in the ciphertext domain due to the three-party interaction required.

5.2. The Backdoor Identification

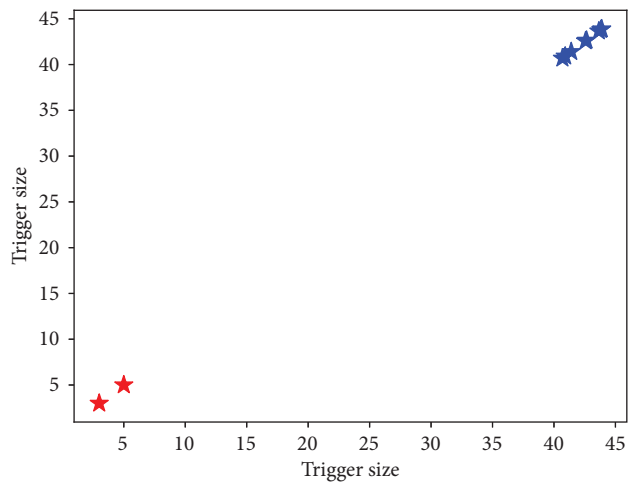
5.2.1. Experiment Setup. According to experience, we set η to 3 in both plaintext and ciphertext domains. The value ϵ depends on the dataset used. In the plaintext domain, when the mean value of the triggers computed by the reverse engineering algorithm is greater than 50, we set ϵ to 10. Otherwise, ϵ is 5. Meanwhile, the value ϵ in the ciphertext domain is generally three times higher than that in the plaintext domain. Moreover, we consider the three of the



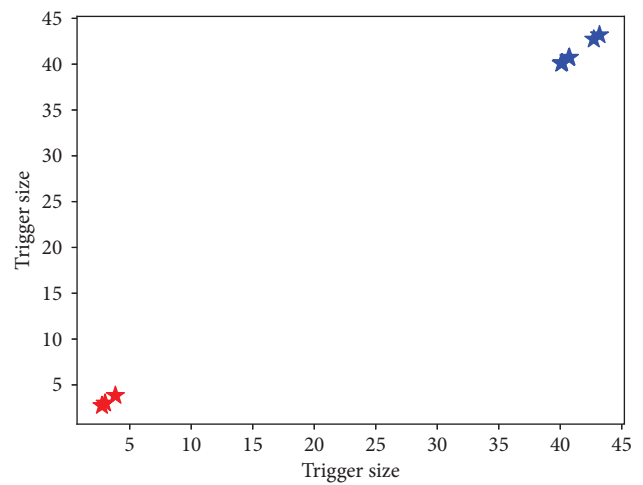
(a)



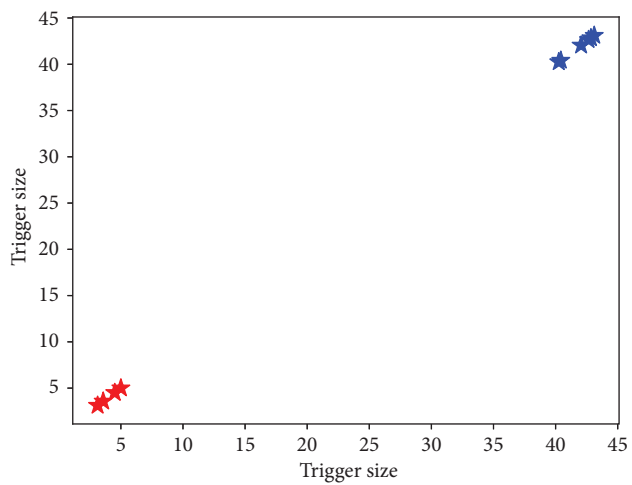
(b)



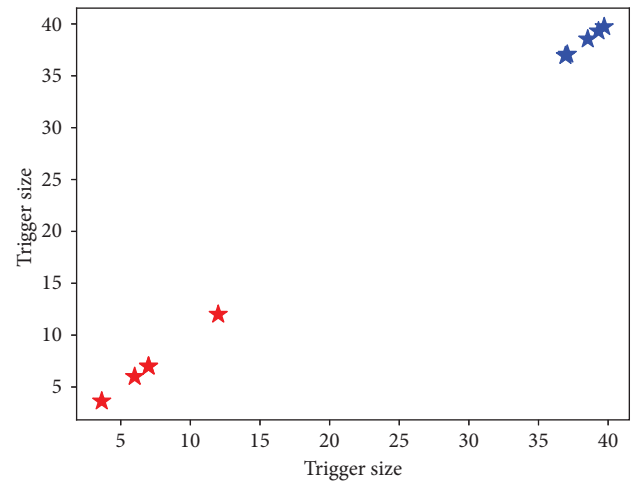
(c)



(d)



(e)



(f)

FIGURE 7: Continued.

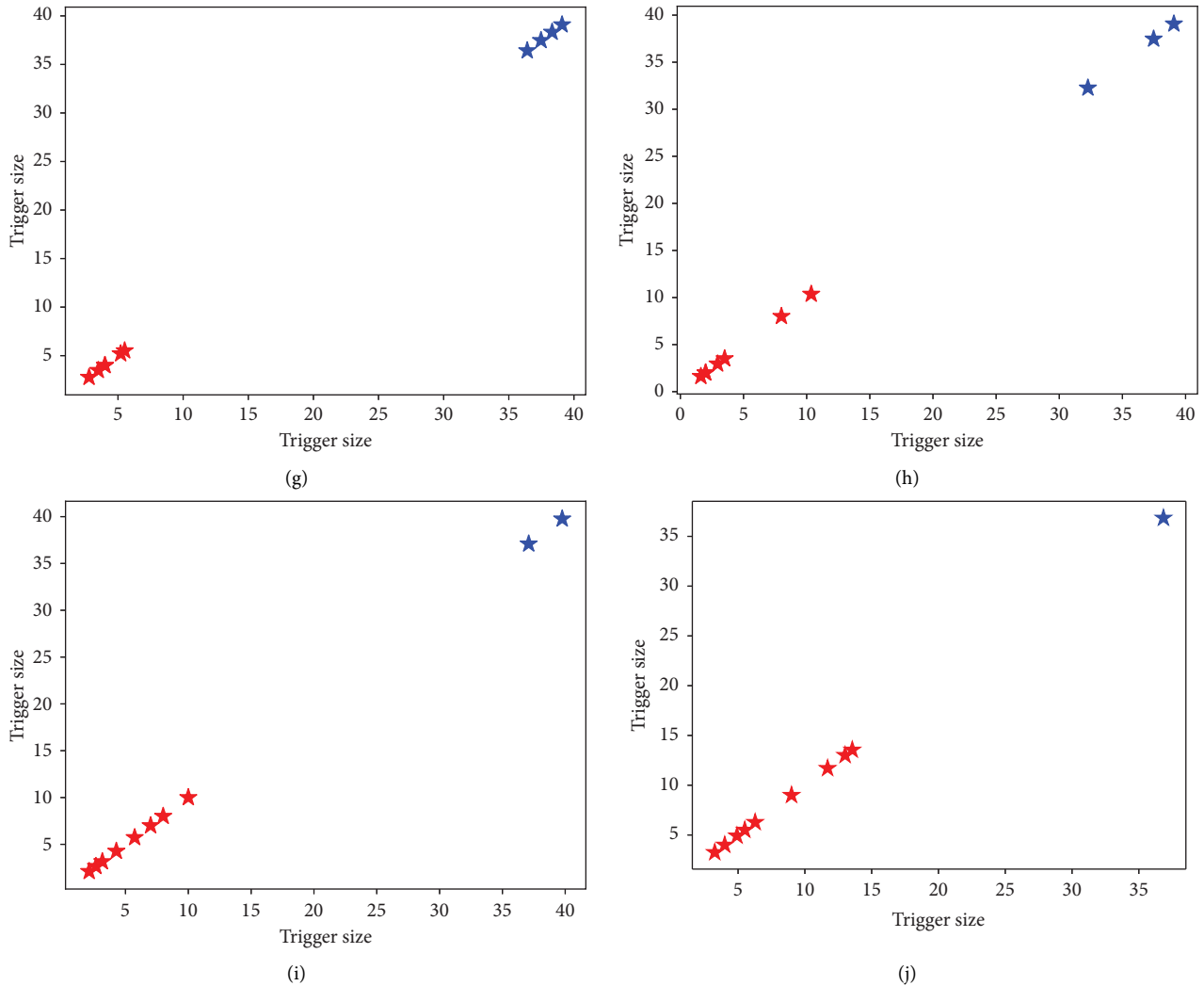


FIGURE 7: The identification results for different number of attacked labels. If the DNN model is not embedded with backdoors, the identification result is only one category (one color in the figure). Otherwise, the backdoors are embedded (two colors in the figure). (a) The number of attacked labels is 0. (b) The number of attacked labels is 1. (c) The number of attacked labels is 2. (d) The number of attacked labels is 3. (e) The number of attacked labels is 4. (f) The number of attacked labels is 5. (g) The number of attacked labels is 6. (h) The number of attacked labels is 7. (i) The number of attacked labels is 8. (j) The number of attacked labels is 9.

most common datasets to measure the effectiveness of backdoor identification algorithms. The network structure settings for these datasets are described as follows:

- (i) MNIST. The DNN structure used for MNIST dataset is shown in Table 4. In the training phase, the base learning rate is set as 0.001, epochs are 10, the batch size is 32, and we use the Adam optimizer to optimize with a learning rate decay of 1×10^{-5} . The trained clean model achieved an accuracy of 99.25%. In this scenario, we use triggers that are 3×3 pixel squares.
- (ii) SVHN [52]. We train a street view house number classifier on the DNN structure as in Table 5. We trained the model using the Adam optimizer with 15 epochs. The base learning rate is 0.001, the batch size is 32, and the learning rate decay is 1×10^{-5} . The final accuracy achieved for the trained clean model was 93.53%. Moreover, the

trigger we used for this dataset is a 3×3 pixel square.

- (iii) GTSRB [53]. We assume that the user specifies the structure of the trained deep neural network consists of 6 convolution layers and 2 dense layers, as shown in Table 6. We trained a traffic sign classifier using the same Adam classifier with 10 epochs. The base learning rate is 0.001, the batch size is 32, and the learning rate decay is 1×10^{-5} . The obtained clean model has a prediction accuracy of 96.53% on the test dataset. Moreover, the trigger we used for this dataset is a 3×3 pixel square.

5.2.2. *The Effectiveness of Backdoor Identification.* Figure 6 shows the trigger sizes for the different numbers of attacked labels before/after the backdoor identification

TABLE 4: The DNN structure used for MNIST dataset.

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	16	3×3	1	ReLU
MaxPool	16	2×2	2	—
Conv	32	3×3	1	ReLU
MaxPool	32	2×2	2	—
FC	512	—	—	ReLU
FC	10	—	—	Softmax

TABLE 5: The DNN structure used for SVHN dataset.

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	32	3×3	1	ReLU
MaxPool	32	2×2	2	—
Conv	64	3×3	1	ReLU
MaxPool	64	2×2	2	—
FC	512	—	—	ReLU
FC	10	—	—	Softmax

TABLE 6: The DNN structure used for GTSRB dataset.

Layer Type	# of Channels	Filter Size	Stride	Activation
Conv	32	3×3	1	ReLU
Conv	32	3×3	1	ReLU
MaxPool	32	2×2	2	—
Conv	64	3×3	1	ReLU
Conv	64	3×3	1	ReLU
MaxPool	64	2×2	2	—
Conv	128	3×3	1	ReLU
Conv	128	3×3	1	ReLU
MaxPool	128	2×2	2	—
FC	512	—	—	ReLU
FC	43	—	—	Softmax

algorithm. In this section, we only show the trigger size calculated by the reverse engineering algorithm in plaintext for comparison. The location of the backdoor attack is shown in Figure 4(b). Experimental results show that the trigger size is smoother after our backdoor identification algorithm, thus greatly reducing the probability of the DBSCAN algorithm identifying the labels with large trigger sizes as outliers. In multi-label attacks, our backdoor identification algorithm does not process the trigger size of the normal label to be about the same size as the trigger of the attacked label, as shown in Figures 6(i) and 6(j).

The identification results for different numbers of attacked labels are depicted in Figure 7. The backdoor identification algorithm classifies normal labels and target labels into two categories. It can identify the specific label being attacked by analyzing the values of the two categories. For the clean model, i.e., Figure 7(a), the classification result is for only one category. For the poisoned model, Algorithm 2 can classify the results into two categories, as shown in Figures 7(b)–7(j), where the blue color represents the normal label, while the red color represents the poisoned label. The

experimental results show that our backdoor identification algorithm can effectively detect whether the model is injected with a backdoor and can identify the specific attacked labels.

5.2.3. Comparison with NC. To further compare with the most related work [13], Figure 8 gives the results after recovering triggers in the plaintext domain (NC [13]) and in the ciphertext domain (this paper) on the MNIST dataset [13]. We can observe from Figure 8 that NC [13] and our algorithm can detect the target labels in the plaintext and ciphertext domains, respectively. Figure 8(g) shows the trigger corresponding to the target label 6 recovered by the reverse engineering algorithm, where the result in the plaintext domain is shown on the left, and the result in the ciphertext domain is shown on the right. Although the triggers recovered from the ciphertext domain are more complicated than those recovered from the plaintext domain, we can get the same results. That is, the size of the recovered trigger corresponding to the target label is much smaller than that of other normal labels.

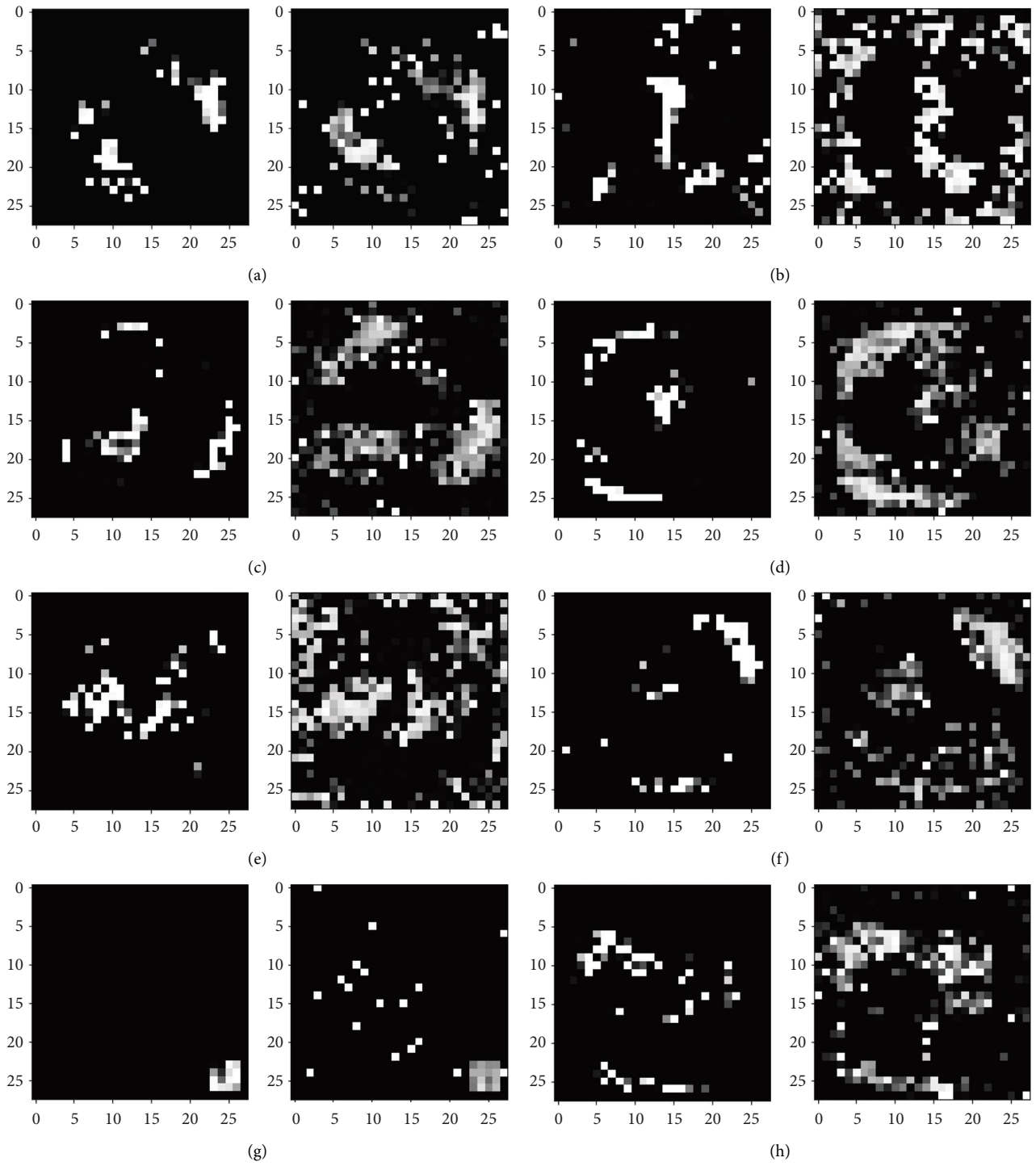


FIGURE 8: Continued.

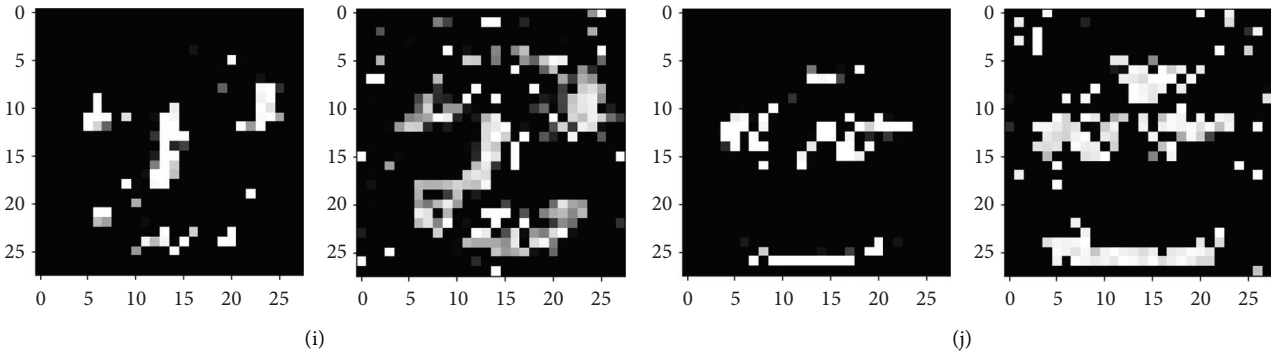


FIGURE 8: Triggers recovered by reverse engineering algorithm on MNIST data. The left side of each subgraph is recovered in the plaintext domain (NC [13]), while the right side of each subgraph is recovered in the ciphertext domain (this paper), which is close to NC [13]. Label 6 is the target label. (a) Label 0. (b) Label 1. (c) Label 2. (d) Label 3. (e) Label 4. (f) Label 5. (g) Label 6. (h) Label 7. (i) Label 8. (j) Label 9.

TABLE 7: Comparison on MNIST, SVHN, and GTSRB datasets.

Dataset	Target labels	Attack location	Classification accuracy (%)	Attac success rate (%)	Backdoor detection and identification	
					NC [13]	Ours
MNIST	—	—	99.25%	—	2	—
	6	9	9.30%	100.00%	6	6
	6, 1	9, 1	99.35%	100.00%	6, 1	6, 1
	6, 1, 2	9, 1, 7	99.38%	100.00%	6, 1, 2	6, 1, 2
	6, 1, 2, 3	9, 1, 7, 6	99.34%	100.00%	—	6, 1, 2, 3
	6, 1, 2, 3, 4	9, 1, 7, 6, 2	99.31%	100.00%	—	6, 1, 2, 3, 4
	6, 1, 2, 3, 4, 5	9, 1, 7, 6, 2, 3	99.27%	100.00%	—	6, 1, 2, 3, 4, 5
	6, 1, 2, 3, 4, 5, 0	9, 1, 7, 6, 2, 3, 5	99.27%	100.00%	—	6, 1, 2, 3, 4, 5, 0
	6, 1, 2, 3, 4, 5, 0, 7	9, 1, 7, 6, 2, 3, 5, 8	99.29%	100.00%	—	6, 1, 2, 3, 4, 5, 0, 7
	6, 1, 2, 3, 4, 5, 0, 7, 9	9, 1, 7, 6, 2, 3, 5, 8, 4	99.20%	100.00%	—	6, 1, 2, 3, 4, 5, 0, 7, 9
SVHN	—	—	93.53%	—	—	—
	0	—	94.56%	98.67%	0	0
	0, 1	1, 2	94.25%	98.97%	0, 1	0, 1
	0, 1, 2	1, 2, 3	94.26%	99.03%	—	0, 1, 2
	0, 1, 2, 3	1, 2, 3, 4	94.11%	99.39%	—	0, 1, 2, 3
	0, 1, 2, 3, 4	1, 2, 3, 4, 5	94.25%	99.16%	—	0, 1, 2, 3, 4
	0, 1, 2, 3, 4, 5	1, 2, 3, 4, 5, 6	94.17%	98.98%	—	0, 1, 2, 3, 4, 5
	0, 1, 2, 3, 4, 5, 6	1, 2, 3, 4, 5, 6, 7	94.11%	99.25%	—	0, 1, 2, 3, 4, 5, 6
	0, 1, 2, 3, 4, 5, 6, 8	1, 2, 3, 4, 5, 6, 7, 8	93.78%	99.36%	—	0, 1, 2, 3, 4, 5, 6, 8
	0, 1, 2, 3, 4, 5, 6, 8, 9	1, 2, 3, 4, 5, 6, 7, 8, 9	93.83%	99.25%	—	0, 1, 2, 3, 4, 5, 6, 8, 9
GTSRB	—	—	96.53%	—	—	—
	28	1	96.54%	92.64%	28	28
	28, 33	1, 2	97.09%	96.00%	28, 33, 20	28, 33
	28, 33, 1	1, 2, 3	96.63%	95.36%	28, 33, 1	28, 33, 1
	28, 33, 1, 7	1, 2, 3, 4	95.75%	95.97%	28, 33, 1, 7	28, 33, 1, 7
	28, 33, 1, 7, 11	1, 2, 3, 4, 5	95.61%	95.50%	28, 33, 1, 7, 11	28, 33, 1, 7, 11
	28, 33, 1, 7, 11, 16	1, 2, 3, 4, 5, 6	96.04%	96.27%	28, 33, 1, 7, 11, 16	28, 33, 1, 7, 11, 16
	28, 33, 1, 7, 11, 16, 23	1, 2, 3, 4, 5, 6, 7	96.15%	96.03%	28, 33, 1, 7, 11, 16, 23	28, 33, 1, 7, 11, 16, 23
	28, 33, 1, 7, 11, 16, 23, 39	1, 2, 3, 4, 5, 6, 7, 8	95.39%	97.08%	28, 33, 1, 7, 11, 16, 23, 39	28, 33, 1, 7, 11, 16, 23, 39
	28, 33, 1, 7, 11, 16, 23, 39, 42	1, 2, 3, 4, 5, 6, 7, 8, 9	96.05%	97.59%	28, 33, 1, 7, 11, 16, 23, 39, 42	28, 33, 1, 7, 11, 16, 23, 39, 42

Note. Target labels denote the target labels of the attack. Attack location denotes the trigger location of the attack target labels, as shown in Figure 4(b). Classification accuracy and Attack success rate are the classification accuracies for the clean and poisoned images. The last two columns are the target labels detected and identified by related work NC [13] and our algorithm.

Table 7 demonstrates the effectiveness of the backdoor identification algorithm on the MNIST, SVHN, and GTSRB datasets. We suppose to attack each of the 9 different labels of the above datasets. The experiments show that the classification accuracy and the attack success rate are at a high level. Moreover, NC [13] and our algorithm both detect and identify the clean model and the poisoned model by single-label attacks, whereas NC [13] fails in the multi-label attacks by regarding the poisoned model as a clean model.

6. Conclusions

This paper presents a framework, DeepGuard, which considers both privacy-preserving and backdoor defense for DNNs in an outsourced cloud environment. We design an effective and efficient reverse engineering algorithm that enables to keep the confidentiality of the data and model parameters in the DNN training. We also propose a practical backdoor identification algorithm that achieves to detect both single-label and multi-label attacks. Finally, the extensive experiments on the various datasets validate the effectiveness and efficiency of our backdoor detection and identification algorithm.

In future, we will pay attention to the backdoor mitigation in the ciphertext domain, which is quite different from that in plaintext due to privacy-preserving DNNs. It is challenging to implement the existing backdoor mitigation scheme for the plaintext domain directly to those in the ciphertext domain. We believe that the efficient privacy-preserving backdoor detection, identification, and mitigation framework in ciphertext will be the future work in the research direction.

Data Availability

We use the public datasets such as MNIST, SVHN, and GTSRB.

Conflicts of Interest

The authors declare that they do not have any conflicts of interest.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China under Grant 61972241 and 61972094, in part by Natural Science Foundation of Shanghai under Grant 22ZR1427100 and 18ZR1417300, and in part by Luo-Zhaorao College Student Science and Technology Innovation Foundation of Shanghai Ocean University.

References








- [1] A. H. Ribeiro, M. H. Ribeiro, G. M. M. Paixão et al., "Automatic diagnosis of the 12-lead ecg using a deep neural network," *Nature Communications*, vol. 11, no. 1, pp. 1760–1769, 2020.
- [2] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *Proceedings of the 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 132–142, IEEE, Montpellier, France, September 2018.
- [3] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: deep hypersphere embedding for face recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 212–220, HI, USA, July 2017.
- [4] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan, "Do we need more training data?" *International Journal of Computer Vision*, vol. 119, no. 1, pp. 76–92, 2016.
- [5] Y. Li, B. Wu, Y. Jiang, Z. Li, and S. Xia, "Backdoor learning: a survey," *CoRR*, vol. abs/2007, Article ID 08745, 2020.
- [6] W. Lifei, C. Congcong, Z. Lei, L. Mengsi, C. Yujiao, and W. Qin, "Security issues and privacy preserving in machine learning," *Journal of Computer Research and Development*, vol. 57, no. 10, p. 2066, 2020.
- [7] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47230–47244, 2019.
- [8] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, pp. 11957–11965, 2020.
- [9] X. Gong, Y. Chen, Q. Wang et al., "Defense-resistant backdoor attacks against deep neural networks in outsourced cloud environment," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2617–2631, 2021.
- [10] S. Koffas, J. Xu, M. Conti, and S. Picek, "Can you hear it? backdoor attacks via ultrasonic triggers," *CoRR*, vol. abs/2107, Article ID 14569, 2021.
- [11] Y. Liu, S. Ma, Y. Aafer, W. C. Lee, and X. Zhang, "Trojaning attack on neural networks," in *Network and Distributed System Security Symposium*, 2017.
- [12] X. Chen, A. Salem, M. Backes, S. Ma, and Y. Zhang, "Badnl: backdoor attacks against nlp models," in *Proceedings of the ICML 2021 Workshop on Adversarial Machine Learning*, NY, USA, December 2021.
- [13] B. Wang, Y. Yao, S. Shan et al., "Neural cleanse: identifying and mitigating backdoor attacks in neural networks," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723, IEEE, CA, USA, May 2019.
- [14] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: scanning neural networks for backdoors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1265–1282, London, UK, November 2019.
- [15] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: a defence against trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 113–125, San Juan Puerto Rico USA, December 2019.
- [16] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science (Sfcs 1982)*, pp. 160–164, IEEE, 1982.
- [17] P. Mohassel and Y. Zhang, "Secureml a system for scalable privacy-preserving machine learning," in *Proceedings of the 2017 IEEE symposium on security and privacy SP*, pp. 19–38, IEEE, CA, USA, May 2017.
- [18] M. S. Riaz, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: a hybrid secure computation framework for machine learning applications," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 707–721, Incheon Republic of Korea, June 2018.

- [19] P. Mohassel and P. Rindal, "Aby3: a mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 35–52, Toronto Canada, October 2018.
- [20] S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin, "Falcon: honest-majority maliciously secure framework for private deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 1, pp. 188–208, 2021.
- [21] M. Keller, "Mp-spdz: a versatile framework for multi-party computation," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1575–1590, USA, October 2020.
- [22] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," in *Proceedings of the USENIX Security Symposium*, pp. 1505–1521, USENIX Association, Vancouver, Canada, August 2021.
- [23] R. Shokri, "Bypassing backdoor detection algorithms in deep learning," in *Proceedings of the 2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 175–183, IEEE, Genoa, Italy, September 2020.
- [24] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: a natural backdoor attack on deep neural networks," in *European Conference on Computer Vision*, pp. 182–199, Springer, 2020.
- [25] A. Salem, M. Backes, and Y. Zhang, "Don't trigger me! A triggerless backdoor attack against deep neural networks," *CoRR*, vol. abs/2010, Article ID 03282, 2020.
- [26] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2041–2055, London, UK, November 2019.
- [27] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: defending against backdooring attacks on deep neural networks," in *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294, Springer, Heraklion, Crete, Greece, September 2018.
- [28] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, *Deepinspect: A Black-Box Trojan Detection and Mitigation Framework for Deep Neural Networks*, pp. 4658–4664, IJCAI, Vienna, Austria, 2019.
- [29] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," 2019, <https://arxiv.org/abs/1911.07116>.
- [30] H. Qiu, Y. Zeng, S. Guo, T. Zhang, M. Qiu, and B. Thuraisingham, "Deepsweep: an evaluation framework for mitigating dnn backdoor attacks using data augmentation," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pp. 363–377, Hongkong, China, June 2021.
- [31] G. Shen, Y. Liu, G. Tao et al., "Backdoor scanning for deep neural networks through k-arm optimization," in *Proceedings of the International Conference on Machine Learning. PMLR*, pp. 9525–9536, Baltimore MD, February 2021.
- [32] Y. Wang, W. Li, E. Sarkar, M. Shafique, M. Maniatakos, and S. E. Jabari, "Pidan: a coherence optimization approach for backdoor attack detection and mitigation in deep neural networks," 2022, <https://arxiv.org/abs/2203.09289>.
- [33] D. Demmler, T. Schneider, and M. Zohner, *Aby-a Framework for Efficient Mixed-Protocol Secure Two-Party Computation* NDSS, Chennai, India, 2015.
- [34] X. Zhang, X. Chen, J. K. Liu, and Y. Xiang, "Deeppar and deepdpa: privacy preserving and asynchronous deep learning for industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2081–2090, 2020.
- [35] H. Chaudhari, A. Choudhury, A. Patra, and A. Suresh, "Astra: high throughput 3pc over rings with application to secure prediction," in *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pp. 81–92, London, UK, November 2019.
- [36] M. Byali, H. Chaudhari, A. Patra, and A. Suresh, "Flash: fast and robust framework for privacy-preserving machine learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 2, pp. 459–480, 2020.
- [37] H. Chaudhari, R. Rachuri, and A. Suresh, "Trident: efficient 4pc framework for privacy preserving machine learning," NDSS. The Internet Society, Chennai, India, 2020.
- [38] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "Aby2.0: improved mixed-protocol secure two-party computation," in *Proceedings of the 30th {USENIX} Security Symposium {USENIX} Security*, Canada, USA, November 2021.
- [39] E. Chou, F. Tramèr, and G. Pellegrino, *Sentinet: Detecting Localized Universal Attacks against Deep Learning Systems*, S. P. Workshops, Ed., pp. 48–54, IEEE, 2020.
- [40] S. Ma and Y. Liu, "Nic: detecting adversarial samples with neural network invariant checking," in *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS 2019)*, China, July 2019.
- [41] B. Chen, W. Carvalho, N. Baracaldo et al., "Detecting backdoor attacks on deep neural networks by activation clustering," in *SafeAI@AAAI, ser. CEUR Workshop Proceedings* vol. 2301, 2019.
- [42] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "TABOR: a highly accurate approach to inspecting and restoring trojan backdoors in AI systems," *CoRR*, vol. abs/1908, Article ID 01763, 2019.
- [43] S. Shen, S. Tople, and P. Saxena, "Auror: defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pp. 508–519, USA, December 2016.
- [44] T. D. Nguyen, P. Rieger, H. Yalame et al., "FLGUARD: secure and private federated learning," *CoRR*, vol. abs/2101, Article ID 02281, 2021.
- [45] Z. Zhang, J. Li, S. Yu, and C. Makaya, "Safelearning: enable backdoor detectability in federated learning with secure aggregation," *CoRR*, vol. abs/2102, Article ID 02402, 2021.
- [46] C. Xie, M. Chen, P. Chen, and B. Li, "CRFL: certifiably robust federated learning against backdoor attacks," in *ICML, ser. Proceedings of Machine Learning Research* vol. 139, pp. 11372–11382, PMLR, 2021.
- [47] C. Chen, L. Wei, L. Zhang, and J. Ning, "Mp-badnet: a backdoor-attack detection and identification protocol among multi-participants in private deep neural networks," in *Proceedings of the ACM Turing Award Celebration Conference-China (ACM TURC 2021)*, pp. 104–109, China, July 2021.
- [48] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party

- computation with an honest majority,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 805–817, France, April 2016.
- [49] D. P. Kingma and J. Ba, *Adam: a method for stochastic optimization* ICLR, ND, India, 2015.
- [50] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” *Kdd*, vol. 96, no. 34, pp. 226–231, 1996.
- [51] L. Deng, “The mnist database of handwritten digit images for machine learning research [best of the web],” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
- [52] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, vol. 2011, 2011, http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.
- [53] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition,” *Neural Networks*, vol. 32, pp. 323–332, 2012, <https://www.sciencedirect.com/science/article/pii/S0893608012000457>.

Research Article

TFPPASV: A Three-Factor Privacy Preserving Authentication Scheme for VANETs

Zongtao Duan ¹, Jabar Mahmood ¹, Yun Yang ¹, Michael Abebe Berwo ¹,
Abd al Kader Ahmed Yassin ², Muhammad Nasir Mumtaz Bhutta ³,
and Shehzad Ashraf Chaudhry ^{3,4}

¹School of Information and Engineering, Chang'an University, Xi'an 710064, China

²Hatay Mustafa Kemal University (MKU) Turkish-Hatay/Hassa-MYO Girne, 79 Sokak, Hassa 31700, Turkey

³Department of Computer Science and Information Technology, College of Engineering, Abu Dhabi University, Abu Dhabi, UAE

⁴Department of Computer Engineering, Faculty of Engineering and Architecture, Nisantasi University, Istanbul 34398, Turkey

Correspondence should be addressed to Yun Yang; yangyun@chd.edu.cn

Received 25 May 2022; Revised 6 August 2022; Accepted 12 August 2022; Published 30 September 2022

Academic Editor: AnMin Fu

Copyright © 2022 Zongtao Duan 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.

A vehicular ad hoc network (VANET) is essential for the autonomous vehicle industry, and with the advancement in VANET technology, security threats are increasing rapidly. Mitigation of these threats needs an intelligent security protocol that provides unbreakable security. In recent times, various three-factor authentication solutions for VANET were introduced that adopt the centralized Trusted Authority (TA), which is responsible for assigning authentication parameters during vehicle registration, and the authentication process depends on these parameters. This article first explains the vulnerabilities of the recent three-factor (3F) authentication scheme presented by Xu et al. Our analysis proves that if an RSU is dishonest, it can easily bypass the TA and can create a session with OBU. Furthermore, this paper puts forward a new scheme that provides the 3F authentication for VANETs (TFPPASV) to resist RSU from bypassing the TA and to offer user privacy. The proposed scheme fulfills the security and performance requirements of the VANET. We use BAN-Logic analysis to perform a formal security analysis of the proposed scheme, in addition to the informal security feature discussion. Finally, we compare the security and performance of the proposed TFPPASV with some recent and related schemes.

1. Introduction

Due to its dynamic structure and related advantages including the realization of autonomous cars, increased road safety, congestion avoidance, and so on, the vehicular ad hoc networks (VANETs) are getting more popularity and are being considered as the only vehicular network structure of the future. In recent years, the road travel safety is also being considered as most important factor for transportation industry and accordingly several technologies are being developed. A general model of vehicular ad hoc network (VANET) [1–3] is given in Figure 1. VANET is a subbranch of MANETs; intelligent transportation system (ITS) [4] provides support to manage transportation efficiently on

roads. VANET consists of three parts [5]. (i) On-board unit (OBU) [6]: OBU is installed inside the vehicle at the time of manufacture from the company side. The OBU stores the information related to vehicle identity, vehicle password, and other parameters necessary for registration and communication; without this confidential information, the vehicle cannot communicate to other OBUs or road side unit (RSU) [2]. OBU communicates to other OBUs or RSUs on the road using the dedicated short range communication (DSRC) protocol [7–9]. (ii) RSU is fixed alongside the road; RSU has more computational and communication power than OBU. RSU provides the facilities to OBUs to communicate with other OBUs or to communicate with RSU via DSRC. In addition, OBU wants to communicate with

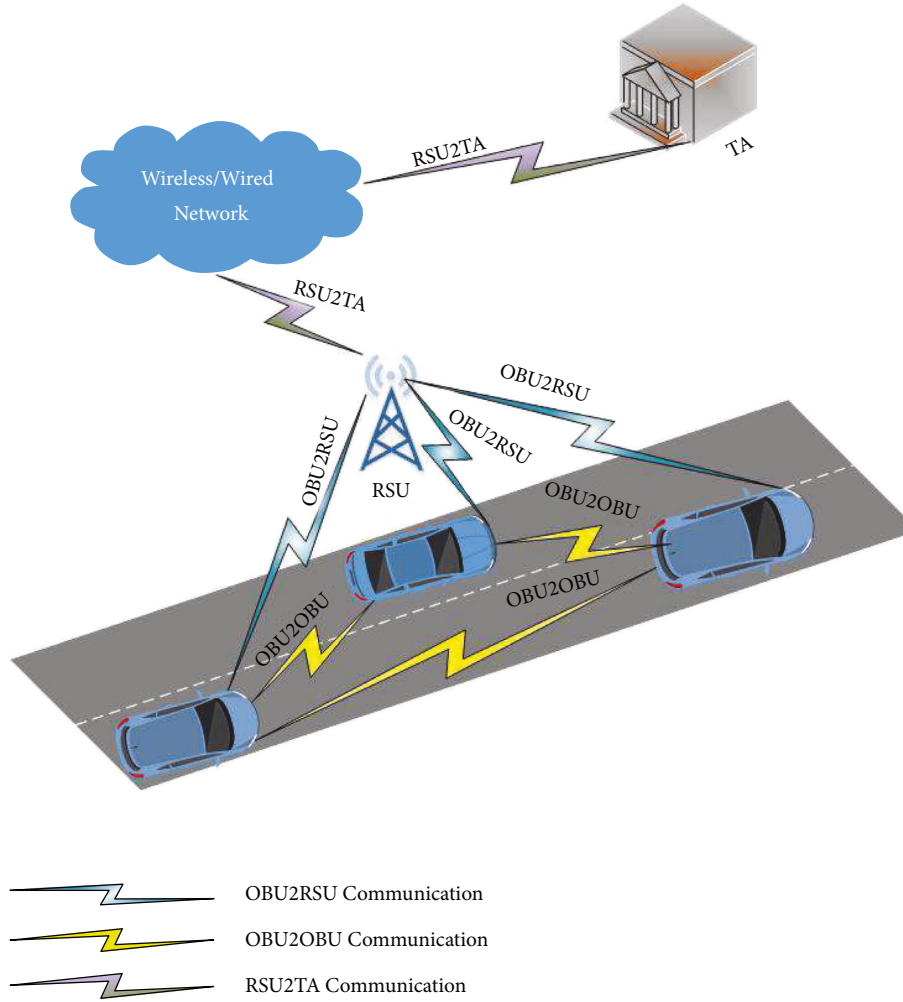


FIGURE 1: General model of VANETs.

Trusted Authority (TA) [2]. RSU acts as a mediator between OBU and TA , where the communication among RSU and TA is carried over some wired or wireless channel. (iii) TA provides authentication parameters to facilitate communication among various entities in a VANET. TA is responsible for completing all node authentication. VANETs provide more comfortable and reliable facilities to passengers and drivers on the road, such as infotainment, weather conditions, location information, traffic congestion, and so on. These services aim to provide a safe drive and secure human life on the road and proper energy resource utilization. Due to VANET's openness characteristics, many security threats are faced during communication. Avoiding these security threats needs a secure authentication scheme that provides resilience against all such threats.

1.1. Motivation. In recent past, many researchers proposed various authentication schemes for VANETs, but many of these schemes do not fulfill the security requirements and are having insecurities against various threats. In addition, some of these schemes have high computational and communication costs. Due to these limitations, we propose a

three-factor authentication scheme and key agreement for VANETs. In our scheme, RSU and TA perform authentication processes. RSU reduces the TA computational and communication cost and performs the authentication. In the proposed scheme, TA hands over a smart card (SC) to each registering vehicle. Inside the SC, TA stores confidential information such as the biological information of the vehicle to provide better security. The proposed scheme provides the facilities to identify malicious vehicle in a multi-drive environment.

1.2. Contributions. The contributions of this study are as follows:

- (1) Firstly, we reviewed and revealed that Xu et al.'s authentication scheme for IoV is insecure against TA bypassing attack. Additionally, an improved scheme titled "TFPPASV: A Three-Factor Privacy Preserving Authentication Scheme for VANETs" is proposed.
- (2) Secondly, the security of the proposed TFPPASV scheme is proved using BAN-Logic in addition to the

informal discussion on critical security feature provision of the proposed TFPPASV scheme.

- (3) We also provided a comparative security and performance analysis of the proposed TFPPASV with some related and recent authentication schemes.

1.3. Organization. The remaining structure of the paper is organized as follows. Section 2 describes the preliminaries such as elliptic curve cryptography, fuzzy extractor, network model, and attack model. Section 3 provides the summary of the related work, and Section 4 details the previously published Xu et al.'s scheme [10]. Section 5 summarizes the weaknesses of Xu et al.'s scheme. In Section 6, the proposed TFPPASV is explained briefly. Section 7 analyzes the BAN-Logic-based security proof of the proposed TFPPASV, in addition to the security feature discussion under various attacks. In Section 8, we conduct security and performance comparisons with related schemes. Finally, a conclusion is provided in Section 9.

2. Preliminaries

This section describes the elliptic curve cryptography (ECC), fuzzy extraction, network model, and attack model used in the proposed TFPPASV. Moreover, Table 1 provides the notation used in this paper.

2.1. Elliptic Curve Cryptography. The concept of elliptic curve cryptography (ECC) was presented by Miller and Koblitz in 1985 [11]. ECC is an asymmetric cryptography technique and the following are details related to ECC.

Characteristics of ECC:

- (i) In ECC, the key generation time is faster than other cryptographic techniques.
- (ii) The size of the ECC key is small and provides the same security, for example, RSA key size is 1024-bit and ECC key size is 160-bit.

Currently, ECC is used in various authentication schemes, devices, and applications such as VANETs, wireless sensor networks, mobiles, RFID devices, bitcoin, and safe web browsers through SSL/TLS due to its small key size. In this paper, we also used the ECC for a secure scheme. Here, we describe the basics of ECC.

The ECC equation $E: y^2 = x^3 + ix + j \pmod{p}$ is used to describe the mathematical operations, where $i, j \in \mathbb{Z}_p^*$ and $4i^3 + 27j^2 \pmod{p} \neq 0$ such that p is a large prime number ($|P| \geq 2^{160}$). Here, we discuss two computationally intensive problems along with a trapdoor function (TF) role in ECC.

- (i) TF is defined as a function that is a one-way function easy to compute in one direction but if computing in the reverse direction is computationally difficult, every public key cryptography has its TF.
- (ii) Elliptic curve discrete logarithm problem (ECDLP): Let $U = kV$ and $k < n$, if V and k are known, U can be computed easily, whereas, it is computationally

difficult to compute k such that $k \in \mathbb{Z}_p^*$, if U and V are known.

- (iii) Elliptic curve computational Diffie–Hellman problem (ECCDHP): let $U = aP$ and $V = bP$ be two points on E and $\{a, b\} \in \mathbb{Z}_p^*$. It is computationally hard to calculate the $W = abP$ point, provided that a, b are unknown.

2.2. Fuzzy Extractor. Authentication through complex passwords is not a better idea for secure registration on an insecure channel. A good technique for secure registration is biometric template, for example, heartbeat, fingerprint, and iris templates are usually used for authentication.

The characteristics of the biometric key are given below:

- (i) Biometric keys are unique and these are not easy to replicate.
- (ii) No need to store or memorize because it comes from the user's body.
- (iii) No duplicate keys are generated.
- (iv) Cannot be estimated or guessed.
- (v) Challenging to reprint and distribute.

Biometrics using raw data are not safe, and thus the biometric data must be stored safely in the system. Various security methods are developed to save the biometric information, such as fuzzy extractor and bio-hash function. They mostly used the fuzzy extractor because the bio-hash function faces the denial of service attack.

The fuzzy extractor has been widely used in an authentication scheme for extracting the biometric key.

The fuzzy extractor has two processes with the following parameters (W, l, t) where W is the input string.

- (i) $\text{Gen}(\cdot)$ is a probability generation procedure. In this procedure, input W is the biometric information from the user, α is a random secret key of the length of l , and β is a public string extracted from the input W , and (1) describes the procedure of generation key.

$$\text{Gen}(W) = (\alpha, \beta). \quad (1)$$

- (ii) $\text{Rep}(\cdot)$ is the process of reproduction and in this procedure, and R can be retrieved as per biometric information W' close to W and β . (2) describes the procedure of reproduction key. For all W, W' , if $d(W, W') \geq t$, there is (2) under precondition (1), where $d(W, W') \geq t$ represent the distance between W and W' which should not be greater than l .

$$\alpha = \text{Rep}(W', \beta). \quad (2)$$

Here, we define the fuzzy extractor.

- (iii) In (3), there is a high probability that the distance between two biometric values W and W' generated from the same entity is low, which can be described as

TABLE 1: List of notations.

Notations	Description of notations
TA, OBU	Trusted authority, on-board unit
RSU	Road side unit
$V2V, V2I$	Vehicle to vehicle, vehicle to infrastructure
IoV	Internet of vehicle
x	RSU and TA private key
P_{pub}	TA public key
A	An adversary
TPD	Temper proof device
W_i	Biometric Information of U_i
α	The random biometric secret key of U_i
β	The public reproduction parameter of U_i
G	An elliptic curve cycle additive group
P	A generator of G
P	Order of G
SC	Smart card
U_i	i_i h users
t_0, t_1, t_2	Timestamp
y_i, r, k	Random number
ID_i	The identity of vehicle/user (U_i)
PW_i	The password of vehicle/user (U_i)
A_i, C_i, D_i, E_i	TA -generated U_i parameters
$h(.)$	One-way cryptography hash function
\oplus	XOR operation

$$P_r [\text{dis}(W, W') < t] \geq 1 - \epsilon_{f_n}, \quad (3)$$

where t is the predetermined tolerance threshold and “false negative” probability is ϵ_{f_n} .

- (iv) There is a high probability that the distance between two biometric values, W_1, W_2 , for two entities is high, which is described in the following equation:

$$P_r [\text{dis}(W, W') < t] \geq 1 - \epsilon_{f_n}, \quad (4)$$

where $t' < t$ and ϵ_{f_p} is the probability of “false positive.”

2.3. Network Model. The network model of the proposed security scheme is presented in Figure 2.

TA: TA is an autonomous or fully trusted entity in VANET responsible for system initialization and registration of a vehicle or a user. TA has more resources in the shape of communication and computational cost. It knows about all RSUs’ locations and identities. It issues the parameters to the nodes in VANET and transmits via a secure channel to each node.

RSU: RSU is fixed alongside the road and is equipped with temper proof device (TPD). TPD is responsible for storing data and performing encryption operations on data. RSU communicates with TA via wired or wireless channels and OBU via DSCR protocol. RSU holds information about all registered vehicles in the range of RSU. In addition, RSU shares information with authenticated vehicles via a session key created during authentication.

OBU: each vehicle has its OBU device fixed inside it and stores all confidential information integral for OBU to prove

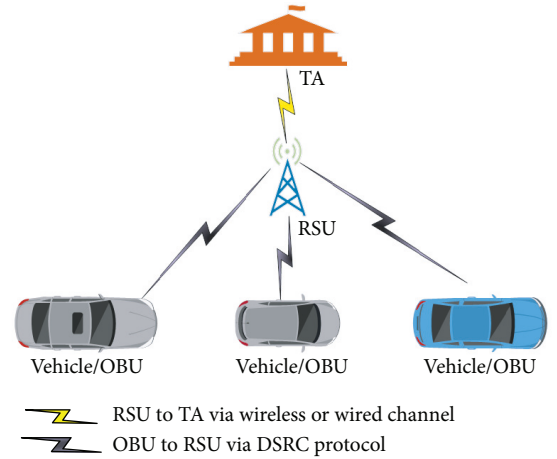


FIGURE 2: Proposed scheme network model.

its authenticity. OBU links to RSU via DSRC protocol. Before communication, OBU proves their authenticity; if OBU proves that it is authenticated, then it communicates with RSU; otherwise, it stops the session key generation.

2.4. Attack Model. In this paper, we consider the common DY adversarial model with following description:

- (1) An adversary (A) plays the role of an eavesdropper, who easily eavesdrops on the insecure communication link and can modify/change or replay the message or send a new message on the link. A can also stop/remove a message from the communication link.

- (2) If A gets the vehicle smart card (SC), he can quickly get all the confidential information stored in SC.
- (3) TA is assumed to be secure. Precisely, except the private key of the TA , rest of the parameters stored on TA could be exposed to A .
- (4) TPD is an important temper proof device because the authenticated data of RSU are stored inside the TPD. Suppose E captures the TPD; it cannot extract the data from the TPD.

3. Related Work

Due to dynamicity of VANETs environment, communication process deviates from other networks. VANET communication in smart cities faces various security threats such as eavesdropping, tracking, and positioning. Security and anonymity provisions are required to avoid these issues. Zheng et al. [12] proposed a VANETs authentication scheme for smart cities. Zhang et al.'s scheme uses certificateless group signature and the Elliptic curve scalar multiplication operations. Zheng et al. [12] proved that overhead cost of their scheme is less than Chen et al.'s [13] and Zhao et al.'s [14] schemes. However, they failed to provide the security analysis of the proposed scheme.

Two-factor security authentication protocols in VANETs are mainly accepted and used for authentication between $V2V$ and $V2I$ on the insecure communication channel. In recent years, various two-factor authentication schemes were proposed, but most of these schemes are vulnerable to one or more weaknesses including SC loss, impersonation assaults, and offline password guessing assaults. Qu and Tan. [15] proposed a password based remote user authentication with key agreement scheme using ECC. Qu and Tan [15] proved that their proposed scheme provides security against various known security threats, but they did not provide the communication cost, running time, and overhead cost of their scheme.

Nandy et al. [16] proposed an authentication scheme using ECC. Nandy et al. [16] proved through security analysis that the proposed scheme provides security against several VANET security attacks. Nevertheless, Chaudhry [17] proved that the ECC techniques used by Nandy et al. involve a faulty operation and their scheme cannot compute the private key of the vehicles. Therefore, their scheme cannot complete the authentication process in their described manner.

Chuang and Lee [18] proposed a security scheme called TEAM in 2013 for $V2V$ secure communication. In TEAM, TA is only for initialization and vehicle registration, which reduces the computational cost of TA . However, Zhou et al. [19] in 2017 highlighted the weakness of the Chuang and Lee's scheme [18] and proved that it cannot perform against inside assaults such as impersonation assaults. Thus, Zhou et al. [19] proposed an authentication scheme that removes the weakness of Chuang and Lee's scheme [18]. In 2019, Wu et al. [20] revealed the weakness of Zhou et al.'s scheme [19]

and proved that it cannot perform against impersonation assault, identity guessing assault, and vehicle anonymity. Wu et al. [20] proposed a scheme for $V2V$ secure communication through mutual authentication.

In 2020, Vasudev et al. [21] proposed a security scheme related to mutual authentication between $V2V$ of IoV and proved that it worked against various VANET attacks through informal security analysis. However, they did not provide a formal security analysis of the scheme. In 2021, Mahmood et al. [22] highlighted its weakness and proved that it does not work in dense environments if more than one vehicle is registered. Thus, Mahmood et al. [22] proposed a new scheme that removes the weakness of Vasudev et al. [21] and proved it through formal analysis and informal analysis.

The main issue faced in VANETs is the provision of security to the user on the road because the nature of VANETs is different from the other communication networks. Therefore, more focus on the secure and authentication process is mandatory to avoid the VANET threats. In 2016, Jiang et al. [23] proposed a scheme related to WSN and implemented the three-factor authentication mechanism and proved that it works better than other schemes. However, in 2017, Li et al. [24] pointed out the functional and security flaws in Jiang et al.'s [23] scheme and proposed a new scheme for WSN. Li et al. [24] removed the flaws of the Jiang et al.'s scheme and proved through formal and informal security analysis that their proposed scheme provides correctness and incurs less computation and communication cost than other schemes. However, they did not provide the running time of the proposed scheme.

Wang et al. [25] proposed a two-factor authentication scheme for vehicular ad hoc networks. The scheme aims to provide lightweight authentication and parallel security against various security threats such as denial of service attacks that cause traffic jamming. Wang et al.'s [25] scheme provides biometric security to vehicles; thus, adversaries cannot track and trace the vehicle's location and identity. However, the authors [25] did not provide a formal security analysis of the scheme.

In 2010, Paruchuri and Durresi [26] proposed a protocol called PAAVE. In that protocol, the smart card generated a key for authentication between the vehicle and RSU. Paruchuri and Durresi [26] provided security comparison but did not provide formal and informal security analysis.

In 2017, Ying and Nayak [27] proposed lightweight authentication for VANETs; the authors [27] focused on efficiency and anonymity. The proposed protocol reduces 50% computation and communication cost compared to other protocols. The scheme of Ying and Nayak [27] provides password change feature without involvement of TA . In 2019, Chen et al. [28] discovered some weaknesses in Ying and Nayak's scheme and proved that the scheme does not perform securely against location spoofing, offline identity guessing, and replay attack. In addition, it takes more time for authentication; after that, Chen et al. [28] also proposed a protocol to remove these vulnerabilities from the

TABLE 2: Summary of authentication schemes in VANETs.

Authors	Cryptography technique	Advantage	Disadvantage
Zheng et al. [12]	ECC	Less storage cost, suitable for OBU and RSU in sense of less computing and limited storage	Missing formal security analysis
Qu and Tan [15]	ECC	Provides mutual authentication and key agreement, resists against impersonation attack, stolen smart card, inside attack, and sever spoofing attack, provides user anonymity	Missing communication cost, running time, and overhead cost
Nandy et al. [16]	ECC and symmetric key operation-based authentication	Lightweight and provides vehicle to vehicle secure communication	Faulty design
Vasudev et al. [21]	XOR operation, one-way hash functions	Resists against impersonation attack, stolen smart card, offline password guessing, and man-in-the-middle attacks and provides anonymity	Missing formal security analysis
Mahmood et al. [22]	XOR operations, one-way hash functions	Proved that Vasudev et al.'s scheme [21] is incorrect and proposed new scheme for V2V secure communication. Resists against impersonation attack, stolen smart card, offline password guessing, man-in-the-middle attacks, and DOS attack and provides anonymity and untraceability.	—
Li et al. [24]	XOR operation, hash function, biometric authentication	Improves the functional and security flaws of Jiang et al.'s scheme [23], communication and computation cost is less than that of other schemes	Running time of scheme is missing
Wang et al. [25]	Using multiple hashing functions, biological password-based authentication	Reduces the communication, overhead, and computation cost	Formal security analysis is missing
Paruchuri and Durrezi [26]	Smart card-based key generation	Provides anonymous authentication, less space for key storage in smart card	Informal and formal security analysis is missing

scheme presented in [27]. Table 2 provides the bird's eye view of the previous related works such as cryptography techniques, and their advantages and disadvantages are listed in the table.

4. Summary of Xu Et Al.'s Scheme

This section provides a detailed review of Xu et al.'s scheme [10]. The scheme is divided into six phases and three entities are participating in this scheme. First of all, we explain the entities and then phases of the scheme. User U_i or OBU acts as a vehicle or node that wants to communicate with other OBUs or RSUs. The second entity is RSU which plays the role of an intermediate node between U_i and TA . U_i communicates with RSU via DSRC protocol and RSU communicates with TA via a wired or wireless channel. The last entity of this scheme is TA , and it is responsible for user authentication and making sure users have been authenticated. These three entities perform activities in six phases such as (1) system initialization, (2) registration, (3) user login, (4) user authentication, (5) malicious user tacking, and (6) password and biometric key exchange phase.

4.1. System Initialization. In system initialization phase, TA performs the following steps:

- (i) TA selects G , which is a cyclic additive group having order p and $E: y^2 = X^3 + iX + j \pmod p$ where $i, j \in_R Z_p^*$. The TA further generates x as the primary/private key and computes the $P_{pub} = x.P$ as the public key; after generation, the public key is published by

TA . Through secure channel, TA loads the private key x in the RSUs and TPD.

4.2. User Registration. Under this phase, U_i approaches the TA for the completion of the registration process. The following are the steps involved in user registration phase:

- (i) U_i puts W_i (his biometric information) on the reader to get $\{\alpha_i, \beta_i\} = \text{GEN}(W)$ via FE and provides his original identity ID_i and password PW_i to the TA . TA generates $y_i: \{i = 1, 2 \dots n\}$ randomly for each U_i . Moreover, TA computes $PID_i = h(ID_i, y_i)$, $A_i^* = h(ID_i, x) \oplus \alpha_i$, $C_i = h(ID_i, PW_i, \alpha_i) \oplus y_i$, $D_i = h(PW_i, y_i, \alpha_i)$, $E_i = PID_i \oplus A_i$. After that, TA forwards the SC to U_i with engraved information of the tuple $\langle G, p, P, \beta_i, C_i, D_i, h \rangle$ and stores the tuple $\{PID_i, \alpha_i\}$ in a verifier table.

4.3. User Login. For user login, the OBU checks and verifies the legitimacy of users via execution of the following steps:

- (i) User U_i inserts the SC into OBU and enters the ID_i^* and PW_i^* and imprints biometric information W_i^* . The SC extracts $\alpha_i^* = \text{Rep}(W^l, \beta_i)$. The SC computes $y_i^* = C_i \oplus (ID_i^*, PW_i^*, \alpha_i^*)$. The SC verifies $D_i \stackrel{?}{=} h(PW_i^*, y_i^*, \alpha_i^*)$. If the information is true, login is successful. The SC computes $A_i^* = E_i \oplus (ID_i^*, y_i^*)$; after that, user attenuation will start. Otherwise, SC terminates the registration process. If U_i repeatedly enters wrong information and exceeds the threshold value, it will not accept inputs from U_i .

4.4. User Authentication. Under this phase, OBU and RSU perform mutual authentication and produce secret key for data communication through authentication process. Figure 3 describes the whole process of user authentication of Xu et al.'s scheme, and the following steps are involved:

Step 1. OBU \rightarrow RSU : $\{DID_i, M_0, M_1, R_0, t_0\}$.

- (i) OBU generates a random number r and computes $R_0 = rP, R_1 = rP_{pub}$. After that, OBU computes the dynamic identity $DID_i = h(ID_i^*, y_i^*) \oplus h(R_1, t_0)$ and t_0 (timestamp). Now, OBU computes $M_0 = A_i^* \oplus h(R_1, t_0)$ and $M_1 = h(ID_i^*, DID_i, A_i^*, R_1, t_0)$. OBU sends $DID_i, M_0, M_1, R_0, t_0$ to RSU through insecure channel.

Step 2. RSU \rightarrow TA : $\{DID_i, AID_i, t_0\}$.

- (ii) After receiving the message from the OBU, the RSU checks the freshness of t_0 and verifies whether the message has expired or not. If $t_{r1} - t_0 \geq \Delta t$, RSU immediately stops the process; otherwise, continue. RSU computes $R_1 = xR_0$. RSU computes $PID_i^* = DID_i \oplus h(R_1, t_0)$, $A_1^* = M_0 \oplus h(R_1, t_0)$. RSU computes $AID_i = h(DID_i, x) \oplus PID_i^*$. Now, RSU sends the message $\{DID_i, AID_i, t_0\}$ to the TA.

Step 3. TA \rightarrow RSU : $\{BID_i, t_1\}$.

- (iii) When TA receives the message form the RSU, it checks the freshness of t_1 and verifies whether the message has expired or not. On success, TA computes $PID_i^* = AID_i \oplus h(DID_i, x)$. TA searches the legitimate table of U_i based on PID_i^* . If table is not found, TA stops the process; otherwise, it continues the process. TA computes $BID_i = h(DID_i, x) \oplus ID_i$. After computing BID_i , TA sends message $\{BID_i, t_1\}$ to RSU.

Step 4. RSU \rightarrow OBU : $\{M_2, R_2, t_2\}$.

- (iv) When RSU receives the message from the TA side, check the freshness of t_2 and verify whether the received message has expired. On success, the RSU computes $ID_i = BID_i \oplus h(DID_i, x)$ and verifies $M_1 = h(ID_i, DID_i, A_i^*, R_1, t_0)$; if RSU finds these parameters correct and satisfies the originality, the process continues; otherwise, it stops. After that, RSU computes $R_2 = kP, K_S = h(R_0, R_2, kR_0)$, $M_2 = h(K_S, A_i^*, R_0, R_2, t_2)$. Now RSU stores data tuple $\langle PID_i^*, A_i^*, K_S \rangle$. RSU sends the message $\{M_2, R_2, t_2\}$ to OBU.

Step 5. The OBU reacts by executing the following steps.

- (v) When the OBU receives the message from the RSU, it checks the freshness of t_3 , and on success, the SC computes $K_S = h(R_0, R_2, rR_2)$ and verifies $M_2 = h(K_S, A_i^*, R_0, R_2, t_2)$. On successful verification, the OBU considers K_S as session key and U_i as authenticated user.

4.5. Malicious User Tracking. If the malicious vehicle/node tries to authenticate itself, then the following steps will be performed to identify and track the malicious node:

- (i) When RSU gets the message from OBU and computes the $PID_i^* = DID_i \oplus h(R_1, t_0)$, $A_i^* = M_0 \oplus h(R_1, t_0)$, then RSU gets the value of A_i^* from database (stored tuple) (PID_i^*, A_i^*, K_S) . RSU computes the $MA = PID_i^* \oplus x \oplus t_3$, $M_3 = h(PID_i^*, A_i^*, MA, MP, t_3)$. After that, RSU sends message $\{MA, MP, M_3, t_3\}$ to trusted authority. When TA receives a message from the RSU, TA checks and verifies the freshness of message and stops the process if freshness is not validated. The TA computes the $PID_i^* = MA \oplus x \oplus t_3$ and $A_i^* = MP \oplus PID_i^* \oplus t_3$. The TA checks and verifies the $M_3 = h(PID_i^*, A_i^*, MA, MP, t_3)$. If it holds, the process continues.
- (ii) The TA searches the verifier table; if the table contains $(ID_i, PID_i^*, \alpha_i)$, the process continues. The TA checks and verifies $A_i^* = h(ID_i, x) \oplus \alpha_i$; if this parameter holds, the process continues. After the confirmation of the malicious vehicle, TA computes the $MN = ID_i \oplus x \oplus t_4$, $M_4 = h(ID_i, MN, t_4)$ and sends a message to RSU $\{MN, M_4, t_4\}$. RSU deletes the entry from the legal user table and declares that malicious user is not a legitimate user. After receiving the message from TA, the RSU computes the message again and checks its originality such as $ID_i = MN \oplus x \oplus t_4$, $M_4 = h(ID_i, MN, t_4)$. Now, RSU broadcasts the malicious node identity (ID_i, PID_i^*) to inform other nodes or vehicles.

4.6. Password and Biometric Change. Under this phase, the user changes his password or gives the vehicle to another user. The user changes his biometric key using the following step:

- (i) The U_i inserts SC into OBU and enters the identity ID_i^* and password PW_i^* and imprints the biometric information W_i' . The FE extracts $\alpha_i^* = \text{Rep}(W_i', \beta_i)$. The SC computes $y_i^* = C_i \oplus h(ID_i^*, PW_i^*, \alpha_i^*)$. The SC checks and verifies $D_i = h(PW_i^*, y_i^*, \alpha_i^*)$; if equation carries these parameters, U_i is granted permission to change his/her password and biometric key; otherwise, it stops the process. In case U_i wants to change his/her password, the SC computes the $C_{i_{New}} = h(ID_i^*, PW_{i_{New}}^*, \alpha_i^*) \oplus y_i$, $D_{i_{New}} = h(PW_{i_{New}}^*, y_i^*, \alpha_i^*)$. The SC replaces the values of C_i, D_i with $C_{i_{New}}, D_{i_{New}}$ and stores these into memory.
- (ii) If U_i wants to hand over the vehicle temporarily to another user, he/she must change biometric key. $U_{i_{New}}$ puts his own biometric information $W_{i_{New}}$ in the special device to get $\text{Gen}(W_{i_{New}}) = (\alpha_{i_{New}}, \beta_{i_{New}})$ via fuzzy extractor. SC computes the

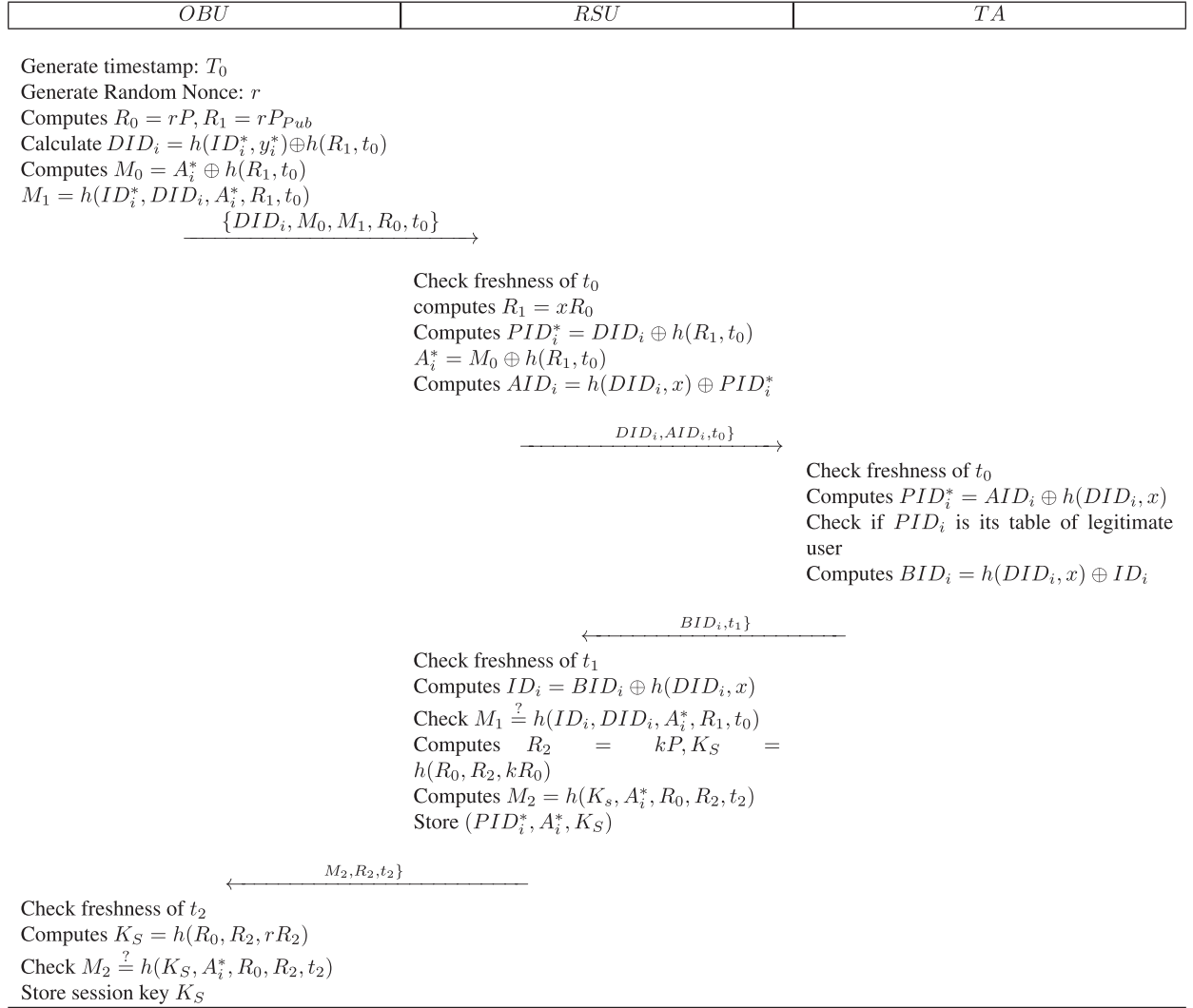


FIGURE 3: User authentication phase of Xu et al.'s scheme.

$C_{i_{New}} = h(ID_i^*, PW_i^*, \alpha_i) \oplus y_i^*$ and $D_{i_{New}} = h(PW_i^*, y_i^*, \alpha_{i_{New}})$. SC computes $A_i^* = E_i h(ID_i^*, y_i^*)$, $h(ID_i^*, x) = A_i^* \oplus \alpha_i$ and $E_{i_{New}} = h(ID_i^*, x) \oplus \alpha_{i_{New}} \oplus h(ID_i^*, y_i^*)$. SC replaces (β_i, C_i, D_i, E_i) in memory with $(\beta_{i_{New}}, C_{i_{New}}, D_{i_{New}}, E_{i_{New}})$ to complete the process of biometric key exchange.

5. Weaknesses of Xu Et Al.'s Scheme

This section describes the ability of a dishonest RSU to bypass TA and construct a session key with requesting OBU.

5.1. TA Bypassing. If an RSU is dishonest, it can easily by pass TA and create a session key directly with OBU, and for this, RSU can skip sending message (DID_i, AID_i, t_0) . In this case, the RSU will calculate $R_1 = xR_0$, $PID_i^* = DID_i \oplus h(R_1, t_0)$, and $A_i^* = M_0 \oplus h(R_1, t_0)$. Now RSU just skips some of the remaining steps and goes directly on the step which computes $R_2 = kP, K_S = h(R_0, R_2, kR_0)$ and $M_2 = h(K_S, A_i^*, R_0, R_2, t_2)$ and sends PID_i^*, A_i^*, K_S to OBU. The OBU checks validity of t_2 and then computes

$K_S = h(R_0, R_2, rR_2)$. Finally, the OBU checks $M_2 = h(K_S, A_i^*, R_0, R_2, t_2)$. As the computation of K_S involves R_0, R_2 , and $rR_2 = kR_0$ and the RSU has access to all these parameters, it does not require any information from the TA. Therefore, it can easily compute K_S without any verification by the TA. Hence, in the scheme of Xu et al. [10], a dishonest RSU can bypass the TA.

6. Proposed Scheme

The following subsections explain the main phases of the proposed scheme.

6.1. System Initialization. Under this phase, TA performs the following steps for registration:

- (i) TA selects the cyclic additive group G with order of p and a generator P .
- (ii) TA selects an ECE : $y^2 = X^3 + iX + j \pmod{P}$ where $\{i, j\} \in \{-RZ_p^*\}$.

- (iii) TA generates a primary key x as a random number and then computes the $P_{\text{pub}} = x.P$ as the public key.
- (iv) Through secure channel, TA uploads the primary key x into RSUs and TPD.

6.2. User Registration. Under this phase, user and TA interact through following steps for the completion of registration process, where U_i approaches the TA to complete the process:

- (i) The U_i puts his biometric information on the reader to get $\{\alpha_i, \beta_i\} = \text{GEN}(W)$ via FE and provides his original identity ID_i and password PW_i to the TA .
- (ii) TA generates a random number y_i for each U_i , and TA computes $\text{PID}_i = h(ID_i, y_i)$, $A_i^* = h(ID_i, x) \oplus \alpha_i$, $C_i = h(ID_i, PW_i, \alpha_i) \oplus y_i$, $D_i = h(PW_i, y_i, \alpha_i)$, $E_i = \text{PID}_i \oplus A_i^*$.
- (iii) TA forwards the SC to U_i , which is engraved with the following tuple: $\langle G, p, P, \beta_i, C_i, D_i, h \rangle$. The TA now stores the tuple $\{ID_i, \alpha_i\}$ in the verification table.

6.3. User Login. Under the user login phase, OBU checks and verifies U_i 's legitimacy via the following steps:

- (i) User U_i inserts the SC into OBU and enters the ID_i^* and PW_i^* and imprints biometric information W_i^* . The FE extracts $\alpha_i^* = \text{Rep}(W', \beta_i)$.
- (ii) The SC computes $y_i^* = C_i \oplus (ID_i^*, PW_i^*, \alpha_i^*)$.
- (iii) The SC verifies $D_i \stackrel{?}{=} h(PW_i^*, y_i^*, \alpha_i^*)$. If this information is true, the user login succeeds and SC computes $A_i^* = E_i \oplus (ID_i^*, y_i^*)$. After that, user attenuation will start. Otherwise, SC terminates the registration process and SC sets an error threshold to increase the security. If U_i tries repeatedly through entering wrong information and attempts exceed the threshold value, U_i is blocked.

6.4. User Authentication. Under the user authentication phase, OBU and RSU perform mutual authentication and produce a session key for data/information communication. Figure 4 describes the complete process of user authentication phase of the proposed scheme.

Step 1. OBU \rightarrow RSU: $\{DI, D_i, M_0, M_1, R_0, t_0\}$.

- (i) The OBU generates a random number r and computes $R_0 = rP$, $R_1 = rP_{\text{pub}}$.
- (ii) The OBU computes the dynamic identity $\text{DID}_i = h(ID_i^*, y_i^*) \oplus h(R_1, t_0)$ and t_0 (timestamp).
- (iii) The OBU computes $M_0 = A_i^* \oplus h(R_1, t_0)$ and $M_1 = h(ID_i^*, \text{DID}_i, A_i^*, R_1, t_0)$.
- (iv) The OBU sends $\text{DID}_i, M_0, M_1, R_0, t_0$ to RSU through insecure channel.

Step 2. RSU \rightarrow TA: $\{\text{DID}_i, \text{AID}_i, t_0\}$.

- (v) After receiving the message from the OBU, the RUS checks the freshness of t_0 and verifies

whether the message has expired or not. If the message is fresh, the process continues; otherwise, RSU stops the process.

- (vi) The RUS computes $R_1 = xR_0$, $\text{PID}_i^* = \text{DID}_i \oplus h(R_1, t_0)$, and $A_1^* = M_0 \oplus h(R_1, t_0)$.
- (vii) The RSU computes $\text{AID}_i = h(\text{DID}_i, x) \oplus \text{PID}_i^*$.
- (viii) Now, the RSU sends the message $\{\text{DID}_i, \text{AID}_i, t_0\}$ to the TA .

Step 3. $TA \rightarrow$ RSU: $\{\text{BID}_i, t_1\}$.

- (ix) When TA receives the message form the RSU, it checks the freshness of t_1 and verifies whether message timeliness has expired or not. On successful validation of timeliness, the process continues; otherwise, the process is stopped.
- (x) Now, TA computes $\text{PID}_i^* = \text{AID}_i \oplus h(\text{DID}_i, x)$. TA searches the verifier table for PID_i^* . If corresponding entry in the table is not found, the TA stops the process; otherwise, the process continues.
- (xi) TA computes the $\text{BID}_i = h(\text{DID}_i, x) \oplus ID_i$. After computing the BID_i , TA sends message $\{\text{BID}_i, t_1\}$ to RSU.

Step 4. RSU \rightarrow OBU: $\{M_2, R_2, t_2\}$.

- (xii) When RSU receives the message from the TA side, check the freshness of t_2 and verify whether the received message has expired.
- (xiii) On successful validation of timeliness, the RSU computes $ID_i = \text{BID}_i \oplus h(\text{DID}_i, x)$.
- (xiv) RSU verifies $M_1 \stackrel{?}{=} h(ID_i, \text{DID}_i, A_i^*, R_1, t_0)$ and on success executes the next steps.
- (xv) The RSU computes $R_2 = kP$, $K_S = h(R_0, R_2, kR_0, \overline{ID}_i)$, $M_2 = h(K_S, A_i^*, R_0, R_2, t_2)$.
- (xvi) The RSU stores the data tuple $\langle \text{PID}_i^*, A_i^*, K_S \rangle$.
- (xvii) The RSU sends the message $\{M_2, R_2, t_2\}$ to OBU

Step 5. The OBU performs following steps.

- (xviii) When the OBU receives the message from the RSU, it checks the freshness of t_3 .
- (xix) On successful validation of timeliness, the SC computes the $K_S = h(R_0, R_2, rR_2, \overline{ID}_i)$.
- (xx) Now, SC verifies the $M_2 \stackrel{?}{=} h(K_S, A_i^*, R_0, R_2, t_2)$, and if it is proved, the process of mutual authentication is assumed to be successfully completed. Furthermore, the K_S will be kept for further use.

6.5. Malicious User Tracking. Following is the malicious user tracking phase of the proposed scheme:

- (i) RSU gets the message from OBU, computes the $\text{PID}_i^* = \text{DID}_i \oplus h(R_1, t_0)$, $A_i^* = M_0 \oplus h(R_1, t_0)$, and gets stored tuple $(\text{PID}_i^*, A_i^*, K_S)$. RSU computes the $MA = \text{PID}_i^* \oplus x \oplus t_3$, $MA = \text{PID}_i^* \oplus A_i^* \oplus t_3$, $M_3 = h(\text{PID}_i^*, A_i^*, MA, MP, t_3)$. After that, RSU sends

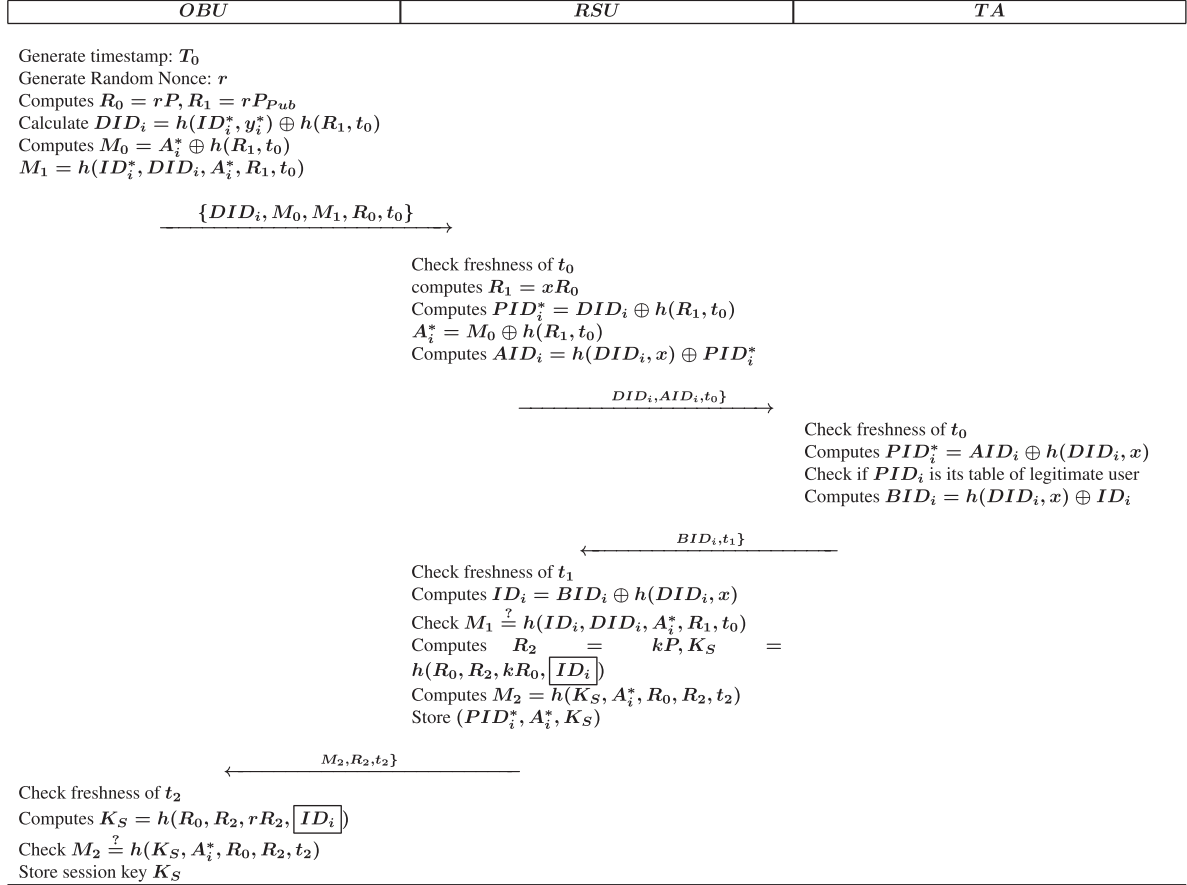


FIGURE 4: User authentication phase of the proposed scheme.

message $\{MA, MP, M_3, t_3\}$ to the trusted authority. When TA receives message from the RSU , TA checks and verifies the freshness of message. On successful validation of timeliness, the TA computes the $PID_i^* = MA \oplus x \oplus t_3, A_i^* = MP \oplus PID_i^* \oplus t_3$. The TA then checks and verifies the $M_3 = h(PID_i^*, A_i^*, MA, MP, t_3)$. On successful validation, the process continues; otherwise, the process is stopped by RSU .

- (ii) The TA searches the user verifier table for $(ID_i, PID_i^*, \alpha_i)$; if these values are found in the table, rest of the process continues; otherwise, the process is stopped. The TA checks and verifies $A_i^* = h(ID_i, x) \oplus \alpha_i$; if this equation holds, the malicious vehicle is identified. After the confirmation of the malicious vehicle, TA computes $MN = ID_i \oplus x \oplus t, M_4 = h(ID_i, MN, t_4)$ and sends a message $\{MN, M_4, t_4\}$ to RSU . The RSU selects the entry from the legal user table and declares that vehicle is malicious. After receiving the message from TA , the RSU computes the message again and checks its originality $ID_i = MN \oplus x \oplus t_4$. RSU broadcasts the malicious node identity (ID_i, PID_i^*) to inform other nodes or vehicles about the malicious node and warns RSU that malicious node is no more allowed to communicate with system entities including the $RSUs$.

6.6. *Password and Biometric Change.* Under this phase, the user changes his password or hands over his vehicle to some other user, and it needs to change his own biometric key. We consider the same process as that used by Xu et al.'s scheme. Therefore, it is not reproduced here.

7. Security Analysis

Under this section, we have performed the formal security analysis using BAN-Logic [29–31] in addition to the security discussion of the proposed scheme.

7.1. *Formal Security Analysis.* This section provides the detailed formal security analysis of the proposed security scheme using the BAN-Logic. It first describes the basic notations of BAN-Logic that are used to analyze the proposed scheme's secure authentication and correctness. Here, X is used for the formula, and N and Q are used as participants.

- (i) $(\#X)$: X is fresh.
- (ii) $N| \equiv X$: N believes that X is trustworthy.
- (iii) $N| \sim X$: N said X once.
- (iv) $N \triangleleft X$: N sees X .
- (v) $N|X$: N has jurisdiction over X .

- (vi) $N \stackrel{K_S}{\leftrightarrow} Q$: between N and Q , K_S is the shared key.
- (vii) $\{X, Y\}_K$: K is used to encrypt X and Y .
- (viii) $(X)_Y$: X and Y are combined.

Following are the rules of BAN-Logic:

Rule 1: message meaning rule.

If N sees X and believes that X is encrypted by shared key K among N and Q , then N believes Q said X once.

$$\frac{N| \equiv \stackrel{K}{\leftarrow} Q, N \triangleleft \{X\}_K}{N| \equiv Q| \sim X} \quad (5)$$

Rule 2: nonce verification rule.

If N believes that the statement X is updated and N also believes that Q once said X , then N believes Q is the statement of X .

$$\frac{N| \equiv \#X, N| \equiv Q| \sim X}{N| \equiv Q| \equiv X} \quad (6)$$

Rule 3: jurisdiction rule.

If N believes Q has jurisdiction over the statement X and N believes Q the statement X , then N believes the statement of X .

$$\frac{N| \equiv Q \Rightarrow X, N| \equiv Q| \equiv X}{N| \equiv X} \quad (7)$$

Rule 4: session key rule.

If N believes the freshness of X , N and Q believes on X , then N believes that a key is shared between N and Q .

$$\frac{N| \equiv \#(X), N| \equiv Q| \equiv X}{N| \equiv N \stackrel{K}{\leftrightarrow} Q} \quad (8)$$

Rule 5: freshness rule.

If a part of X is believed by N as updated, then $\{X, Y\}$ is also believed by N as updated.

$$\frac{N| \equiv \#(X)}{N| \equiv \#(X, Y)} \quad (9)$$

Rule 6: belief rule.

If N believes that Q believes in the statement of $\{X, Y\}$, then N believes that Q believes in the part of statement X .

$$\frac{N| \equiv Q| \equiv \{X, Y\}}{N| \equiv Q| \equiv X} \quad (10)$$

The goals of our TFPPASV protocol are proved through BAN-Logic as under:

- (i) G1: $\text{OBU} | \equiv \text{OBU} \stackrel{K_S}{\leftrightarrow} TA$
- (ii) G2: $TA | \equiv \text{OBU} \stackrel{K_S}{\leftrightarrow} TA$
- (iii) G3: $\text{RSU} | \equiv \text{OBU} \stackrel{K_S}{\leftrightarrow} TA$
- (iv) G4: $\text{OBU} | \equiv TA | \equiv \text{OBU} \stackrel{K_S}{\leftrightarrow} TA$

- (v) G5: $TA | \equiv \text{OBU} | \equiv \text{OBU} \stackrel{K_S}{\leftrightarrow} TA$
- (vi) G6: $\text{RSU} | \equiv \text{OBU} | \equiv \text{OBU} \stackrel{K_S}{\leftrightarrow} TA$
- (vii) G7: $\text{RSU} | \equiv TA | \equiv \text{OBU} \stackrel{K_S}{\leftrightarrow} TA$

In the proposed TFPPASV scheme, the messages are sent over the public channel. The details of these messages are mentioned below:

- (i) M1: $\text{OBU} \longrightarrow \text{RSU}: \text{DID}_i, M_0, M_1, R_0, t_0$
- (ii) M2: $\text{RSU} \longrightarrow TA: \text{DID}_i, \text{AID}_i, t_0$
- (iii) M3: $TA \longrightarrow \text{RSU}: \text{BID}_i, t_1$
- (iv) M4: $\text{RSU} \longrightarrow \text{OBU}: M_2, R_2, t_2$

Furthermore, the following assumptions are used for analyzing the proposed scheme using BAN-Logic.

- (i) A1: $\text{OBU} | \equiv \#(r_{\text{OBU}})$
- (ii) A2: $TA | \equiv \#(r_{\text{RSU}})$
- (iii) A3: $\text{RSU} | \equiv \#(r_{TA})$
- (iv) A4: $\text{RSU} | \equiv \text{OBU} \Rightarrow \text{DID}_i$
- (v) A5: $\text{RSU} | \equiv \text{OBU} \Rightarrow^{A1} TA$
- (vi) A6: $\text{RSU} | \equiv \#(r_{\text{OBU}})$
- (vii) A7: $\text{RSU} | \equiv \text{OBU} \Rightarrow r_{\text{OBU}}$
- (viii) A8: $\text{RSU} | \equiv \#(A_i^*)$
- (ix) A9: $\text{RSU} | \equiv \text{OBU} \Rightarrow (A_i^*)$
- (x) A10: $TA | \equiv \text{RSU} \Rightarrow \text{DID}_i$
- (xi) A11: $TA | \equiv \text{RSU} \Rightarrow \text{AID}_i$
- (xii) A12: $TA | \equiv \#(\text{AID}_i)$
- (xiii) A13: $TA | \equiv \#(\text{PID}_i^*)$
- (xiv) A14: $TA | \equiv \text{RSU} \Rightarrow \text{PID}_i^*$
- (xv) A15: $TA | \equiv \#(r_{\text{OBU}})$
- (xvi) A16: $\text{RSU} | \equiv TA \Rightarrow^{R_1, t_0} TA$
- (xvii) A17: $\text{RSU} | \equiv \#(r_{TA})$
- (xviii) A18: $\text{RSU} | \equiv TA | \Rightarrow r_{TA}$
- (xix) A19: $\text{OBU} | \equiv \text{RSU} | \equiv A_1$
- (xx) A20: $\text{OBU} | \equiv \#(A_1)$
- (xxi) A21: $\text{OBU} | \equiv \#(r_{TA})$
- (xxii) A22: $\text{OBU} | \equiv \text{RSU} | \Rightarrow r_{TA}$
- (xxiii) A23: $\text{OBU} | \equiv \text{OBU} \Rightarrow^{R_1, t_0} TA$
- (xxiv) A24: $\text{OBU} | \equiv (r_{\text{RSU}})$
- (xxv) A25: $\text{OBU} | \equiv TA | \Rightarrow r_{\text{RSU}}$
- (xxvi) A26: $\text{RSU} | \equiv \text{RSU} | \equiv r_{\text{OBU}}$

7.1.1. BAN-Logic Proof. The proof of proposed scheme through BAN-Logic analysis is as follows.

S_1 can be acquired from M_1 .

$S_1: \text{RSU} \triangleleft \{\text{DID}_i, M_0, M_1, R_0, t_0\}$.

$S_2: \text{RSU} | \equiv \text{OBU} | \equiv \text{DID}_i$. Based on A_4, S_2 , and rule 3, we can obtain $S_3: \text{RSU} | \equiv \text{DI } D_i$. According to S_1 , it implies that $S_4: \text{RSU} \triangleleft r_{\text{OBU}}, \text{DID}_{iA1}$. By A_5, S_4 , and rule 1, it implies that $S_5: \text{RSU} | \equiv \text{OBU} | \sim (r_{\text{OBU}}, \text{DID}_i)$. By A_6, S_5 , and rule 2,

we can obtain $S_6: \text{RSU} \equiv \text{OBU} \equiv r_{\text{OBU}}$. According to A_7, S_6 , and rule 3, it implies that $S_7: \text{RSU} \equiv r_{\text{OBU}}$. According to S_1 , we have acquired $S_8: \text{RSU} \triangleleft \text{AID}_i$. By A_5, S_8 , and rule 1, it implies that $S_9: \text{RSU} \equiv \text{OBU} \sim \text{AID}_i$. By A_8, S_9 , and rule 2, we can obtain $S_{10}: \text{RSU} \equiv \text{OBU} \equiv \text{AID}_i$. According to A_9, S_{10} , and rule 3, it implies that $S_{11}: \text{RSU} \equiv \text{AID}_i$.

By M_2 , we can obtain $S_{12}: \text{TA} \triangleleft \text{DID}_i, \text{AID}_i, t_0$ and further $S_{13}: \text{TA} \equiv \text{RSU} \equiv \text{DID}_i$. Based on A_{10}, S_{13} , and rule 3, we can obtain $S_{14}: \text{TA} \equiv \text{DID}_i$. By A_{11}, A_{12} , and rule 4, it implies that $S_{15}: \text{TA} \equiv \xrightarrow{A_1} \text{RSU}$. According to S_{12} , we have $S_{16}: \text{TA} \triangleleft \text{PID}_i^* \text{RSU}$. Based on S_{15}, S_{16} , and rule 1, it implies that $S_{17}: \text{TA} \equiv \text{RSU} \sim r_{\text{RSU}}$. By A_{13}, S_{17} , and rule 2, we can obtain $S_{18}: \text{TA} \equiv \text{RSU} \equiv r_{\text{RSU}}$. According to A_{14}, S_{18} , and rule 3, it implies that $S_{19}: \text{TA} \equiv r_{\text{RSU}}$. Based on A_{11}, A_{12} , and rule 4, we have $S_{20}: \text{TA} \equiv \text{TA} \xrightarrow{\text{PID}_i^*} \text{RSU}$. According to S_{12} , we have $S_{21}: \text{TA} \triangleleft r_{\text{OBU}}$. By S_{20}, S_{21} , and rule 1, it implies that $S_{22}: \text{TA} \equiv \text{RSU} \sim r_{\text{OBU}}$. By A_{15}, S_{22} , and rule 2, we can obtain $S_{23}: \text{TA} \equiv \text{RSU} \equiv r_{\text{OBU}}$. Based on A_{26}, S_{23} , and rule 3, it implies that $S_{24}: \text{TA} \equiv r_{\text{OBU}}$. $K_S = h(R_0, R_2, kR_0, ID_i)$.

$S_{25}: \text{TA} \equiv \text{OBU} \xleftrightarrow{K_S} \text{TA}$ is obtained. (G2). According to A_2, S_{25} , and rule 4, we can obtain $S_{26}: \text{RSU} \equiv \text{OBU} \xrightarrow{A_1} \text{RSU}$. (G5).

By M_3 , we have $S_{27}: \text{RSU} \triangleleft \text{BID}_i, t_1$ and further $S_{28}: \text{RSU} \triangleleft ID_i = \text{BID}_i \oplus h(\text{DID}_i, x)$. Based on A_{16}, S_{28} , and rule 1, we can obtain $S_{29}: \text{RSU} \equiv \text{TA} \equiv r_{\text{TA}}$. By A_{17}, S_{29} , and rule 2, it implies that $S_{30}: \text{RSU} \equiv \text{TA} \equiv r_{\text{TA}}$. Based on A_{18}, S_{30} , and rule 3, we can obtain $S_{31}: \text{RSU} \equiv r_{\text{TA}}$. According to S_{27}, S_{31} , and S_{31} , it implies that $S_{32}: \text{RSU} \equiv \text{OBU} \xleftrightarrow{K_S} \text{TA}$. (G3). Based on A_{16}, S_{32} , and rule 4, we can obtain $S_{33}: \text{RSU} \equiv \text{OBU} \equiv \xleftrightarrow{K_S} \text{TA}$. (G6). According to A_{12}, S_{32} , and rule 4, it implies that $S_{34}: \text{RSU} \equiv \text{TA} \equiv \text{OBU} \xleftrightarrow{K_S} \text{TA}$. (G7).

By M_4 , we have $S_{35}: \text{OBU} \triangleleft M_2, R_2, t_2$. Based on A_{19}, A_{20} , and rule 4, we can obtain $S_{36}: \text{OBU} \equiv \text{OBU} \xrightarrow{h(R_0, R_2, kR_0, ID_i)} \text{RSU}$. According to S_{35} , we have $S_{37}: \text{OBU} \triangleleft r_{(\text{RSU}), h(R_0, R_2, kR_0, ID_i)}$. Based on S_{36}, S_{37} , and rule 1, it implies that $S_{38}: \text{OBU} \equiv \text{RSU} \sim r_{\text{RSU}}$. By A_{21}, S_{38} , and rule 2, we can obtain $S_{39}: \text{OBU} \equiv \text{RSU} \equiv r_{\text{RSU}}$. According to A_{22}, S_{39} , and rule 3, it implies that $S_{40}: \text{OBU} \equiv r_{\text{RSU}}$.

We have $S_{41}: \text{OBU} \triangleleft r_{\text{TA}}, h(\text{OBU} \| r_{(\text{RSU})})$. Based on A_{23}, S_{41} , and rule 1, it implies that $S_{42}: \text{OBU} \equiv \text{TA} \sim r_{\text{TA}}$. By A_{24}, S_{42} , and rule 2, we can obtain $S_{43}: \text{OBU} \equiv \text{TA} | r_{\text{TA}}$. Based on A_{25}, S_{43} , and rule 3, it implies that $S_{44}: \text{OBU} \equiv r_{\text{TA}}$. According to S_{40} and S_{44} , we can obtain $S_{45}: \text{OBU} \equiv \text{OBU} \xleftrightarrow{K_S} \text{TA}$. (G1). According to A_{24} and S_{45} , we can obtain $S_{46}: \text{OBU} \equiv \text{TA} \equiv \text{OBU} \xleftrightarrow{K_S} \text{TA}$. (G4).

7.2. Security Discussion. The security feature provision and resistance of the proposed scheme against various attacks are explained in the following subsection.

7.2.1. Anonymity and Untraceability. In the proposed TFPPASV protocol, the identity ID_i of the user is secure, because in TFPPASV, the vehicle sends a pseudo identity $\text{DID}_i = h(ID_i^*, y_i^*) \oplus h(R_1, t_0)$ instead of its original identity

ID_i over the communication channel. The attacker can intercept DID_i , but it cannot extract ID_i because it is concealed in a oneway hash function along with a random number and other parameters. The only method to get the identity is to break the hash function and get knowledge of random numbers involved in the computation of DID_i . Thus, the protocol provides user anonymity. In addition, the proposed protocol provides untraceability for the user because when the message is transmitted on a communication channel, it uses a random number during the authentication process. Thus, the attacker is not able to track the user.

7.2.2. Perfect Forward Secrecy. The proposed TFPPASV protocol provides ultimate forward secrecy because it uses various random numbers during the message transmission. Three parameters $R_O = rP, R_2 = kP$ and kR_0 are used to construct the session key $K_S = h(R_0, R_2, kR_0, \text{DID}_i)$. If an attacker wants to launch an attack on the basis of a compromised session key, the attacker is not able to obtain the previous and subsequent session keys. Thus, the proposed protocol provides forward secrecy.

7.2.3. Replay Attack. The proposed TFPPASV protocol provides resistance against the replay attack. Three entities (OBU, RSU, and TA) are involved in the authentication phase of the proposed TFPPASV protocol. These entities send the messages to each other such as $(\text{DID}_i, M_0, M_1, R_0, t_0)$, $(\text{DID}_i, \text{AID}_i, t_0)$, (BID_i, t_1) , and (M_2, R_2, t_2) . In each of these messages, random numbers and timestamps are used and these are session specific. If an attacker wants to launch a replay attack, the replayed message cannot pass the verification process and the recipient can easily identify the replay attack.

7.2.4. Offline Password Guessing Attack. Our TFPPASV protocol provides resistance against offline password guessing attack. During registration phase, some parameters are stored into SC such as $C_i = h(ID_i, PW_i, \alpha_i) \oplus y_i$, $D_i = h(PW_i, y_i, \alpha_i)$. The PW_i is masked with y_i generated randomly and the biometric key α_i . Thus, attacker is not able to guess the password.

7.2.5. Impersonation Attack. Our TFPPASV protocol provides resistance against impersonation assaults such as OBU impersonation assault, RSU impersonation assault, and TA impersonation assault.

OBU impersonation attack: if an attacker tries to impersonate the OBU, it requires to construct the original login request message: $(\text{DID}'_i, M'_0, M'_1, R'_0, t'_0)$ $\text{DID}'_i = h(ID_i^*, y_i^*) \oplus h(R_1, t_0)$, $M'_0 = A_i^* \oplus (R_1, t_0)$, $M'_1 = h(ID_i^*, \text{DID}_i, A_i^*, R_1, t_0)$, and $R'_0 = rP$ with updated random number r and timestamp t_0 . However, it is computationally difficult to recover t_0, DID_i , and R_1 for constructing $(\text{DID}'_i, M'_0, M'_1, R'_0, t'_0)$. Thus, the proposed protocol provides security against OBU impersonation.

RSU impersonation attack: for the execution of a RSU impersonation attack, the attacker tries to instigate a forgery

to TA on behalf of the RSU. The attacker needs to construct the (DID'_i, AID'_i, t'_0) with updated timestamp. In addition, it requires more confidential parameters such as (x, R_0) and R_1 . It is computationally hard to calculate these parameters from $(DID_i, M_0, M_1, R_0, t_0)$. Thus, the proposed TFPPASV scheme provides security against RSU impersonation.

TA impersonation attack: in the case of TA impersonation, the attacker needs to construct BID'_i, t_1 with an updated timestamp, and in addition, it requires the private key x , where $BID'_i = (DID_i, x) \oplus ID_i$. However, the attacker is not able to form the message until it gets the private key x and DID_i . Thus, the attacker is not able to launch TA impersonation attack.

7.2.6. Smart Card Stolen. The proposed protocol provides security against SC stolen. If an attacker captures the SC and gets the information $(G, p, P, \beta_i, C_i, D_i, E_i, h)$ from the SC and it wants to login via SC, the attacker also needs the user ID_i, PW_i , and the biometric key α_i in polynomial time, which is not possible for the attacker. Thus, the attacker is not able to complete a successful login.

7.2.7. Man-in-the-Middle Attack. If the attackers want to launch attack as a man-in-the-middle, it needs to capture the messages $(DID_i, M_0, M_1, R_0, R_1)$, (DID_i, AID_i, t_0) , (BID_i, t_1) , and (M_2, R_2, t_2) from the public communication channel. The attacker must change or replace the message and forward it on the channel to get authenticated from both sides. However, due to the inability of construction of legal messages, the attacker may not be able to get authenticated from any side without getting ID_i and PW_i and private key x of the TA .

7.2.8. Insider Attack. The proposed TFPPASV protocol protects from insider attacks because at the time of registration, user registers itself with TA on a secure channel. In addition, stored user passwords are in the ciphertext. It is computationally difficult for any dishonest insider to get information related to passwords and keys.

8. Security and Performance Analysis

This section describes the security features and computational and communication cost of the proposed TFPPASV scheme in relation to other schemes [10, 32–34].

8.1. Security Feature. Table 3 provides the complete bird's eye view of the security feature comparison of our TFPPASV scheme with related schemes [10, 32–34]. Through BAN-Logic analysis, we prove that our proposed scheme is correct. Section 5.1 discusses Xu et al.'s scheme [10] which has TA bypassing issue, and if RSU is dishonest, it can easily bypass the TA and establish a connection directly with OBU. Ma et al.'s [32] scheme does not provide security against malicious user tracking, offline password attack, and smart card stolen attack. Cui et al.'s [33] scheme is also insecure against the man-in-the-middle attack, offline password, and

TABLE 3: Security feature performance analysis.

Schemes	Ours	[10]	[32]	[33]	[34]
Correctness	✓	✓	✓	✓	✓
Vehicle impersonation attack	✓	✓	✓	✓	✓
Trusted authority impersonation attack	✓	✓	✓	✓	✓
Vehicle server impersonation attack	✓	✓	✓	✓	✓
Stolen SC attack	✓	✓	✗	✗	✗
Anonymity attack	✓	✓	✓	✓	✓
Untraceability attack	✓	✓	✓	✓	✓
Man-in-the-middle attack	✓	✓	✗	✗	✗
Offline password guessing attack	✓	✓	✗	✗	✗
Replay attack	✓	✓	✓	✓	✓
Mutual authentication	✓	✓	✓	✓	✓
Malicious user tracing	✓	✓	✓	✗	✗
TA bypassing	✓	✗	✓	✓	✓

Note. ✓: provides or resists; ✗: does not provide or does not resist.

smart card stolen attacks. Zhong et al.'s [34] scheme failed to provide security against the man-in-the-middle attack, offline password attack, and smart card stolen attack. The proposed scheme provides better security features compared to other related schemes [10, 32–34].

8.2. Computational Cost. In this section, we calculate the computational cost (CC) of the proposed TFPPASV scheme and compare it with the related schemes. Before calculating CC, we denote some symbols as follows: xor operation is denoted by T_{\oplus} , the execution time for scale multiplication on ECC is denoted by T_{sm} , and the execution time for hash function is represented by T_h . For calculating the CC, the real-time hardware platform with the following specifications: CPU: Intel I7-6700, with 4.00 GHz RAM 16 GB OS windows 10th, is adopted from [35]. T_{sm} furnishes in 0.442 ms, and the running time of T_h is 0.0001, while T_{\oplus} takes negligible time to complete the execution. Thus, T_{\oplus} is being ignored in the comparisons. We used SHA256 with 256 bit hash digest and the size of identity and random numbers are fixed at 64 bits. The proposed scheme executes $\{6T_{sm} + 15T_h + 8T_{\oplus}\}$ operations with the running time of 2.6535 ms. Referring to Table 4, computational cost of the proposed TFPPASV scheme is low as compared to Ma et al.'s scheme [32] and a bit high as compared to Cui et al. and Zhong et al.'s schemes [33, 34], respectively. However, the proposed TFPPASV scheme offers more security features as compared with related schemes.

8.3. Communication Cost. To calculate the communication cost of the proposed TFPPASV scheme and to compare it with related schemes, we adopted SHA-256 with 256 bit size. We also adopted 256 bit ECC parameters. In addition, identities and timestamps are taken as 64 bit length. In the proposed TFPPASV scheme, total four messages are exchanged for a successful authentication process completion. In message 1, $\{DID_i, M_0, M_1, R_0, t_0\} = \{256 + 256 + 256 + 256 + 64\} = 1088$ bits are sent from OBU to RSU. In message 2, $\{DID_i, AID_i, t_0\} = \{256 + 256 + 64\} = 576$ bits are sent from

TABLE 4: Performance comparisons.

Scheme	Computation cost	RT (ms)	ME	BE	SO
Ours	$6T_{sm}^* + 15T_h^* + 8T_{\oplus}^*$	2.6535	4	2560	1024
Xu et al. [10]	$6T_{sm}^* + 15T_h^* + 8T_{\oplus}^*$	2.6535	4	2560	1024
Ma et al. [32]	$17T_{sm}^* + 19T_h^* + 3T_{\oplus}^*$	7.5159	4	4992	832
Cui et al. [33]	$6T_{sm}^* + 3T_h^*$	2.6523	3	3840	512
Zhong et al. [34]	$1T_{sm}^* + 6T_h^*$	0.4426	3	4648	320

Note. RT: running time; ME: no. of message exchanges; BE: bit exchange; SO: storage overhead.

RSU to TA . In message 3, $\{BID_i, t_1\} = \{256 + 64\} = 320$ bits are sent from TA to RSU. In message 4, $\{M_2, R_2, t_2\} = \{256 + 256 + 64\} = 576$ bits are sent from RSU to OBU. Total communication cost of the proposed TFPPASV scheme is $= \{1088 + 576 + 320 + 576\} = 2560$ bits. Referring to Table 2, the TFPPASV scheme has low communication as compared to other related schemes [32–34].

8.4. Storage Cost. The proposed TFPPASV stores four authentication related parameters $\{P, \beta_i, C_i, D_i\}$ in addition to h function and system parameters $\{G, p\}$. The system parameters and functions take marginal memory and are stored in the smart card in all competing authentication schemes. Therefore, for analysis and comparison purposes, we focus on the authentication related parameters. The storage cost of the proposed TFPPASV $\{P, \beta_i, C_i, D_i\} = \{256 + 256 + 256 + 256\} = 1024$ bits. The storage cost of Xu et al.'s scheme is also same (i.e., 1024 bits). The storage cost of Ma et al. [32], Cui et al. [33], and Zhong et al. [34] is 832, 512, and 320, respectively.

9. Conclusion

In this study, we analyzed a recent authentication scheme and proved that the scheme of Xu et al. can become a victim of TA bypassing attack by a dishonest RSU. We then introduced an improved and bypassing free authentication scheme (TFPPASV) for VANETs. We used the lightweight ECC and symmetric key based functions to design our proposed TFPPASV scheme. In addition to a comprehensive discussion on the security feature provision of TFPPASV, we utilized the BAN-Logic analysis to prove the formal security of the TFPPASV. We also compared the security and performance of the TFPPASV with related schemes and showed that the proposed TFPPASV offers a good trade-off between the security and performance criterion. Therefore, it can be concluded that the TFPPASV is best suitable in practical VANET scenarios.

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.

Authors' Contributions

J.M. and Z.D. were responsible for conceptualization. Y.Y., M.N.M.B., and M.A.B. were responsible for investigation. J.M. and Z.D. were responsible for original draft preparation. J.M., Y.Y., A.K.A.Y., and S.A.C. were responsible for review and editing. Z.D. and S.A.C. were responsible for supervision. Z.D. was responsible for funding acquisition. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

This study was supported by funds for Key Research and Development Plan Project of Shaanxi Province, China (grant nos. 2019ZDLGY17-08, 2019ZDLGY03-09-01, 2020ZDLGY09-02, and 2020GY-013), and funds for Science and Technology Innovation Leading Talent of Shaanxi Province, China (grant no. TZ0336).

References

- [1] Y. Wang, H. Zhong, Y. Xu, J. Cui, and G. Wu, "Enhanced security identity-based privacy-preserving authentication scheme supporting revocation for vanets," *IEEE Systems Journal*, vol. 14, no. 4, pp. 5373–5383, 2020.
- [2] J. Mahmood, Z. Duan, Y. Yang, Q. Wang, J. Nebhen, and M. N. Mumtaz Bhutta, "Security in Vehicular Ad Hoc Networks: Challenges and Countermeasures," *Security and Communication Networks*, vol. 2021, no. 3, pp. 1–20, 2021.
- [3] F. Ahmed, L. Wei, Y. Niu et al., "Toward fine-grained access control and privacy protection for video sharing in media convergence environment," *International Journal of Intelligent Systems*, vol. 37, no. 5, pp. 3025–3049, 2022.
- [4] F.-Y. Wang, D. Zeng, and L. Yang, "Smart cars on smart roads: an IEEE intelligent transportation systems society update," *IEEE Pervasive Computing*, vol. 5, no. 4, pp. 68–69, 2006.
- [5] Y. Ming and X. Shen, "Pcpa: a practical certificateless conditional privacy preserving authentication scheme for vehicular ad hoc networks," *Sensors*, vol. 18, no. 5, p. 1573, 2018.
- [6] J. J. Cheng, J. L. Cheng, M. C. Zhou, F. Q. Liu, S. C. Gao, and C. Liu, "Routing in internet of vehicles: a review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2339–2352, 2015.
- [7] S. Bao, W. Hathal, H. Cruickshank, Z. Sun, P. Asuquo, and A. Lei, "A lightweight authentication and privacy-preserving scheme for vanets using tesla and bloom filters," *ICT Express*, vol. 4, no. 4, pp. 221–227, 2018.
- [8] H. Oh, C. Yae, D. Ahn, and H. Cho, "5.8 ghz dsrc packet communication system for its services," *Gateway to 21st Century Communications Village. VTC 1999-Fall. IEEE VTS 50th Vehicular Technology Conference (Cat. No. 99CH36324)*, vol. 4, pp. 2223–2227, 1999.
- [9] S. S. Manvi and S. Tangade, "A survey on authentication schemes in vanets for secured communication," *Vehicular Communications*, vol. 9, pp. 19–30, 2017.
- [10] T. Xu, C. Xu, and Z. Xu, "An efficient three-factor privacy-preserving authentication and key agreement protocol for

- vehicular ad-hoc network,” *China Communications*, vol. 18, no. 12, pp. 315–331, 2021.
- [11] V. S. Miller, “Use of elliptic curves in cryptography,” in *Conference on the Theory and Application of Cryptographic Techniques*, vol. 218, pp. 417–426, Springer, Berlin, Heidelberg, 1985.
 - [12] Y. Zheng, G. Chen, and L. Guo, “An anonymous authentication scheme in vanets of smart city based on certificateless group signature,” *Complexity*, vol. 2020, pp. 1–7, 2020.
 - [13] Y. Chen, X. Cheng, S. Wang, and M. Gao, “Research on certificateless group signature scheme based on bilinear pairings,” *Netinfo Security*, vol. 3, pp. 53–58, 2017.
 - [14] N. Zhao, G. Zhang, and X. Gu, “Certificateless aggregate signature scheme for privacy protection in vanet,” *Computer Engineering*, vol. 46, no. 1, pp. 114–128, 2020.
 - [15] J. Qu and X.-L. Tan, “Two-factor user authentication with key agreement scheme based on elliptic curve cryptosystem,” *Journal of Electrical and Computer Engineering*, vol. 2014, pp. 1–6, 2014.
 - [16] T. Nandy, M. Y. I. Idris, R. M. Noor et al., “A secure, privacy-preserving, and lightweight authentication scheme for vanets,” *IEEE Sensors Journal*, vol. 21, no. 18, pp. 20998–21011, 2021.
 - [17] S. A. Chaudhry, “Comments on ”a secure, privacy-preserving, and lightweight authentication scheme for vanets,” *IEEE Sensors Journal*, vol. 22, no. 13, pp. 13763–13766, 2022.
 - [18] M.-C. Chuang and J. F. Lee, “Team: Trust-extended authentication mechanism for vehicular ad hoc networks,” *IEEE Systems Journal*, vol. 8, no. 3, pp. 749–758, 2013.
 - [19] Y. Zhou, X. Zhao, Y. Jiang, F. Shang, S. Deng, and X. Wang, “An enhanced privacy-preserving authentication scheme for vehicle sensor networks,” *Sensors*, vol. 17, no. 12, p. 2854, 2017.
 - [20] L. Wu, Q. Sun, X. Wang et al., “An efficient privacy-preserving mutual authentication scheme for secure v2v communication in vehicular ad hoc network,” *IEEE Access*, vol. 7, pp. 55050–55063, 2019.
 - [21] H. Vasudev, V. Deshpande, D. Das, and S. K. Das, “A lightweight mutual authentication protocol for v2v communication in internet of vehicles,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6709–6717, 2020.
 - [22] J. Mahmood, Z. Duan, H. Xue et al., “Secure message transmission for v2v based on mutual authentication for vanets,” *Wireless Communications and Mobile Computing*, vol. 2021, pp. 2021–2116, 2021.
 - [23] Q. Jiang, J. Ma, F. Wei, Y. Tian, J. Shen, and Y. Yang, “An untraceable temporal-credential-based two-factor authentication scheme using ecc for wireless sensor networks,” *Journal of Network and Computer Applications*, vol. 76, pp. 37–48, 2016.
 - [24] X. Li, J. Niu, S. Kumari et al., “A three-factor anonymous authentication scheme for wireless sensor networks in internet of things environments,” *Journal of Network and Computer Applications*, vol. 103, pp. 194–204, 2018.
 - [25] F. Wang, Y. Xu, H. Zhang, Y. Zhang, and L. Zhu, “2flip: a two-factor lightweight privacy-preserving authentication scheme for vanet,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 2, pp. 896–911, 2016.
 - [26] V. Paruchuri and A. Durrezi, “Paave: protocol for anonymous authentication in vehicular networks using smart cards,” in *Proceedings of the 2010 IEEE global telecommunications conference GLOBECOM 2010*, pp. 1–5, IEEE, Manhattan, New York, USA, 2010.
 - [27] B. Ying and A. Nayak, “Anonymous and lightweight authentication for secure vehicular networks,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10626–10636, 2017.
 - [28] C.-M. Chen, B. Xiang, Y. Liu, and K.-H. Wang, “A secure authentication protocol for internet of vehicles,” *IEEE Access*, vol. 7, pp. 12047–12057, 2019.
 - [29] A. C. Shehzad, “Designing an Efficient and Secure Message Exchange Protocol for Internet of Vehicles,” *Security and Communication Networks*, vol. 2021, pp. 1–9, 2021.
 - [30] T. Maitra, M. S. Obaidat, R. Amin, S. K. H. Islam, S. A. Chaudhry, and D. Giri, “A robust elgamal-based password-authentication protocol using smart card for client-server communication,” *International Journal of Communication Systems*, vol. 30, no. 11, Article ID e3242, 2017.
 - [31] T.-Y. Wu, L. Yang, Z. Lee, S.-C. Chu, S. Kumari, and S. Kumar, “A provably secure three-factor authentication protocol for wireless sensor networks,” *Wireless Communications and Mobile Computing*, vol. 202115 pages, 2021.
 - [32] M. Ma, D. He, H. Wang, N. Kumar, K. K. R. Choo, and R. C. Kwang, “An efficient and provably secure authenticated key agreement protocol for fog-based vehicular ad-hoc networks,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8065–8075, 2019.
 - [33] J. Cui, J. Zhang, H. Zhong, and Y. Xu, “Spacf: a secure privacy-preserving authentication scheme for vanet with cuckoo filter,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10283–10295, 2017.
 - [34] H. Zhong, B. Huang, J. Cui, Y. Xu, and L. Liu, “Conditional privacy-preserving authentication using registration list in vehicular ad hoc networks,” *IEEE Access*, vol. 6, pp. 2241–2250, 2018.
 - [35] D. He, S. Zeadally, B. Xu, and X. Huang, “An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.

Research Article

Encoding Test Pattern of System-on-Chip (SOC) Using Annular Scan Chain

Guilin Huang , Zhengjin Zhang , Honghai Wang , Jiabao Jiang , and Qilin Wu 

School of Information Engineering, Chaohu University, Hefei, Anhui, China

Correspondence should be addressed to Guilin Huang; 747818979@qq.com

Received 14 March 2022; Revised 10 May 2022; Accepted 24 August 2022; Published 2 September 2022

Academic Editor: Jinguang Han

Copyright © 2022 Guilin Huang 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 improvement of System-on-Chip integration, the chip requires an increasingly large amount of test data. To solve the contradiction between the storage capacity and bandwidth of automatic test equipment (ATE), a new method of test data compression/decompression is proposed based on an annular scan chain. Corresponding fault bits of different test patterns are incompatible, moving test patterns in an annular scan chain, makes all of the new corresponding bits of different test patterns be compatible or backward-compatible, so different adjacent test patterns form a new relation that are indirectly compatible or indirectly backward-compatible, achieves the purpose of test data compression by encoding these indirectly compatible test patterns or indirectly backward-compatible test patterns. According to experimental results, the average compression ratio increases by %6.94 to % 15.1 compared with the other schemes, relative decompression architecture is simple. In the annular scan chain, the test pattern moves clockwise with the minimal bits, generating subsequent test patterns quickly, it is advantageous to reduce the test application time of a single IP core.

1. Introduction

The rapid development of integrated circuit technology has improved the design and manufacturing capabilities of integrated circuits. Microelectronic products that perform composite functions are integrated into a die due to the advancement in IC and semiconductor technology. The improved fabrication technology leads to billions of transistors on an IC with all functionality required for a system which is called system on chip, which continuously brings a series of new challenges to the test [1, 2, 3]. The significance of testing integrated circuits lies in finding out the problems existing in the design or manufacturing process of circuits as early as possible and finding out the defective or faulty chips. Chips are widely used in various electronic devices, such as aviation, medical care, transportation, and household appliances. Whether the chips can work normally relates to the reliable operation of the devices and the safety of people's lives and property, so it is of great significance to ensure the reliable and trouble-free operation of the chips.

Techniques to reduce test cost includes test scheduling based on the system and test data compression based on a single IP core [4, 5]. Increasing test data is a challenge that chip testing must face. Limited ATE storage space and I/O bandwidth store and transmit huge test data, which will improve test cost and lengthen test time, test data compression technology has been trying to solve this test problem [6, 7].

At present, many mature test data compression schemes are proposed, which can be divided into three categories [8, 9]: compression method based on LFSR structure [10], compression method based on broadcast scan [11, 12], encoding compression method [13, 14, 15]. And encoding compression is a very popular method, encodes the data block according to the corresponding relation of the codewords, such as GOLOMB [16], FDR [17], ALT-FDR [18], EFDR [19], 9C [20], BM [21], RL-HC [22], SHC [23], and VIHC [13]. These are all very classic and excellent schemes, the compression effect it can achieve is still very ideal, but it is difficult to accept in the time of test pattern generation. Test data is generated in the unit of the data

block, the whole test set cannot be generated quickly. This paper presents a test pattern encoding method based on the annular scan chain. The proposed scheme moves all test pattern in a clockwise direction, then finds all indirectly compatible or backward-compatible test patterns. Encodes test pattern according to the minimal bits of moving test patterns in the annular scan chain, achieves the purpose of lossless compression.

Contribution of this paper: (1) Compression ratio increases compared with the other schemes. (2) Besides the conventional control Unit, only MUX and inclusive-OR gate are needed by decompression architecture. Decompression architecture is simple. (3) According to the sequence of linear generation graph, the test pattern moves clockwise with the minimal bits generating subsequent test patterns quickly, reduces the time of test pattern generation, helps reduce the test application time of a single IP core.

2. Proposed Test Data Compression Scheme

2.1. Annular Scan Chain. When the test patterns are loaded into the circuit under test, the circuit will give a test response. After the test response is compressed by the test response compressor, it is sent back to the comparator of ATE and compared with the expected test response of the fault-free circuit. If the test responses of both are the same, it indicates that the tested circuit is considered fault-free; otherwise, it is faulty. With the improvement of integrated circuit integration, more and more IP cores are integrated on the circuit, which increases the complexity of the circuit and makes the test of the circuit more and more complicated. Only relying on the special test equipment ATE to test, the test cost is increasing constantly, so the test depends on testability design.

In order to match the design of the test wrapper, a register is added to the function input of the circuit under test, which is called the boundary scanning register. A boundary scan register is also added to the function output of the circuit under test. As shown in Figure 1, there are four boundary scan registers 1, 2, 3, and 4. During the test, registers 1 and 2 are used to store the function input, that is, to store the test bit, and registers 3 and 4 are used to capture the test response, that is, to store the test response. At this time, the scan chain length of the circuit under test is $m(x + y + z)$. Boundary scanning is an application of scanning path in the I/O boundary. Scanning design can provide controllability and observability for testing.

The integrated circuit includes application logic, related input and output, and a scanning path composed of BSCs, in which case each pin is connected to a BSC. As shown in Figure 2, BSC structures are interconnected, forming a scanning path between the input end (TDI) and the output end (TDO) of the integrated circuit. During the normal operation of IC, input and output signals pass through the BSC module from NDI to NDO, respectively. When entering the boundary test mode, the test data moves in serial mode from the TDI, the test response is moved out from the TDO serially and observed.

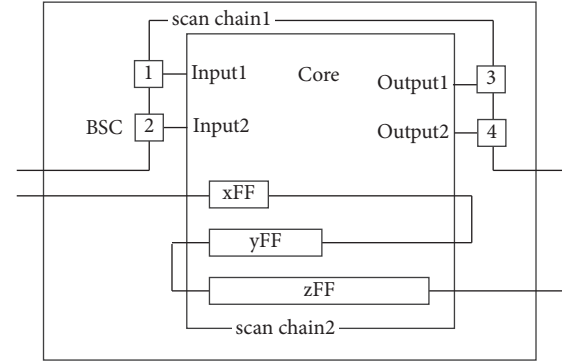


FIGURE 1: Boundary scanning design.

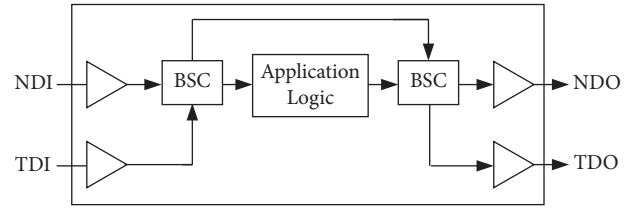


FIGURE 2: Boundary scan cell structure.

Definition 1. All flip-flops in the chip test wrapped are connected together in a string, forming a single unidirectional scan chain. Input is the functional input terminal (external input) of the sequential circuit, be used to transmit partial data of test patterns. Output is the functional output terminal (external output) of the sequential circuit, be used to transmit partial data of test response. The Q of each flip-flop is connected to the input of the combinational circuit in the sequential circuit, and also to the Q' of the flip-flop unit of the next stage. The first flip-flop in the scan chain is connected with the last flip-flop, that is, the Q of the last flip-flop is also used as the input terminal TD of the first flip-flop, as shown in Figure 3. The dotted circuit structure in the figure connects with the original flip-flops to form an annular scan chain. test patterns can be moved clockwise in the scan chain circularly, therefore, it is called an annular scan chain.

Definition 2. The test pattern moves clockwise in the scan chain, forming a logical annular test pattern. test pattern 1: $A_0, A_1, A_2, \dots, A_n$; test pattern 2: $B_0, B_1, B_2, \dots, B_n$; If A_0 and B_0, A_1 and B_1, A_2 and B_2, \dots, A_n and B_n are all compatible or backward-compatible, test pattern 1 and test pattern 2 is directly compatible or directly backward-compatible.

Test pattern1 (x0011) moves clockwise one bit in the annular scan chain forming test pattern 1x001. Test pattern 1x001 is compatible with test pattern 2 (1xx01), is backward-compatible with test pattern3 (01x10). Test pattern1 is said to be indirectly compatible with test pattern2, and indirectly backward-compatible with test pattern3.

2.2. Linear Generation Graph. Each test pattern can be regarded as an annular test pattern in the scan chain logically. Firstly, constructs an annular generation graph, nodes

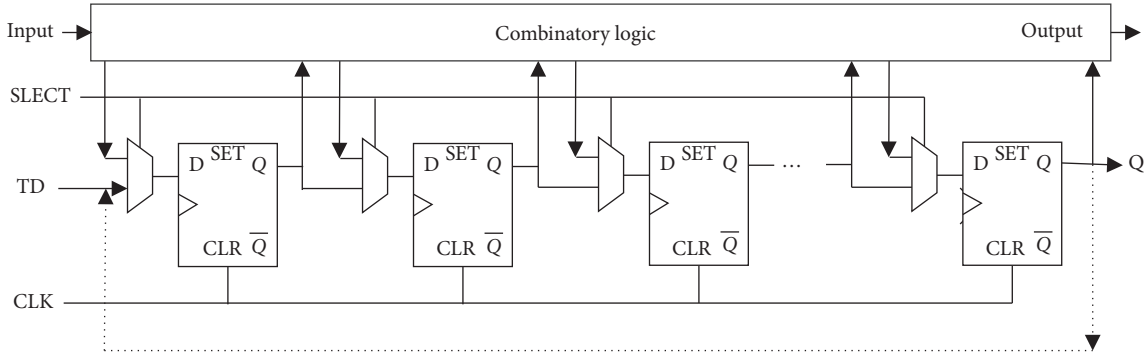


FIGURE 3: Annular scan chain.

are test patterns, out-degrees, and in-degrees of all nodes are 1, there are N edges in N nodes. Secondly, constructs a linear generation graph, the out-degree of the head node is 1, and the in-degree of the tail node is 1, the out-degree and in-degree of each other node are 1, and there are $N-1$ edges. They are indirectly compatible or backward-compatible between nodes, the bits of moving clockwise is the weight of the graph. Test pattern 0 is expressed as $T(0)$. Bit n of test pattern is expressed as $B(n)$.

Assumes that the test pattern length is L , the number of test patterns is N .

Step 1: the initial state of list List1 is an empty set, and the initial state of list List2 contains the whole test set, that is, all test patterns of the circuit under test.

Step 2: test pattern $T(0)$ enters List1 from List2, then there is no $T(0)$ in List2. In step 3 to step 7, the union of List1 and List2 is the whole test set of the circuit under test, that is, all test patterns of the circuit under test.

Step 3: the length of the test pattern is L , and the test pattern $T(1)$ moves n bits clockwise, then the $B(n)$, $B(n+1)$, ..., $B(L-1)$, $B(0)$, $B(1)$, ...and $B(n-1)$ of the annular test pattern $T(1)$, is corresponding to the $B(0)$, $B(1)$, $B(2)$, ..., $B(L-2)$, $B(L-1)$ of the annular test pattern $T(0)$, respectively, counts the number of incompatible bits between $T(0)$ and $T(1)$ and counts the number of compatible bits between $T(0)$ and $T(1)$. The range of n is between 1 and $L-1$. According to different n , counts the number of incompatible bits and compatible bits between $T(0)$ and $T(1)$ for $L-1$ times. In this statistical process, if the minimal number of incompatible bits is 0, it indicates that $T(0)$ and $T(1)$ are indirectly compatible, then $T(1)$ enters List1 from List2. If the minimal number of compatible bits is 0, it indicates that $T(0)$ and $T(1)$ are indirectly backward-compatible, then $T(1)$ enters List1 from List2.

Step 4: repeats this process, that is, compatibility analysis between any one test pattern $T(i)$ in List1 and certain test pattern $T(j)$ in List2. If $T(i)$ is indirectly compatible or backward-compatible with $T(j)$, then $T(j)$ enters List1 from List2.

Step 5: if any test pattern in List1 is neither indirectly compatible nor backward-compatible with any test

pattern in List2, then selects any test pattern $T(k)$ in List1 to move clockwise. When $T(k)$ moves by one bit every time clockwise, it is judged whether it is indirectly compatible or backward-compatible with other test patterns. If it is found that $T(m)$ is indirectly compatible or backward-compatible with $T(k)$ firstly, then $T(m)$ is set as the adjacent node of $T(k)$. $T(m)$ will be removed from List1.

Step 6: repeats step 5 until List1 is an empty set. The last test pattern $T(n)$ becomes the adjacent node of $T(k)$, forming an annular generation graph as shown in Figure 4.

Step 7: removes the edge with the largest weight and construct the directed linear generation graph as shown in Figure 5. In the test process, test patterns will be generated according to the sequence of directed linear generation graph, the seed test pattern $T4$ will generate test pattern $T1$, $T1$ will generate $T2$, and so on until $T6$ is generated, and all test patterns will be generated.

Step 8: selects the first node $T4$ on the linear generation graph as the seed, and other nodes are encoded by the number of bits moved clockwise.

Step 9: returns step 3 to step 8, and so on and constructs a linear generation graph for the rest of the test patterns in List2 until all the test patterns enter the linear generation graph and completes the encoding.

In order to describe the generation process of linear generation graph, provide pseudocode as follow.

Assumes that List2 includes all original test patterns in test set, List1 includes test patterns, which is indirectly compatible or backward-compatible with $TV1$. $TV1$ is selected from List2, which is the first element added to the null set List1. TV_i stores the result that $TV1$ moves i bits clockwise in the annular scan chain, TV_j represents the test pattern j in test set. TV stores the calculation result that TV_i and TV_j perform exclusive-or operation by bit. TV_k represents the bit k of TV . TVB stores the calculation result that TV_i and TV_j perform inclusive-or operation by bit. TVB_k represents the bit k of TVB .

Assumes that $0 \oplus X = 0$, $1 \oplus X = 0$, $X \oplus X = 0$. Assumes that $0 \circ X = 0$, $1 \circ X = 0$, $X \circ X = 0$.

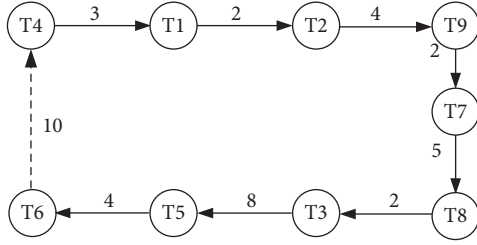


FIGURE 4: Annular generation graph.

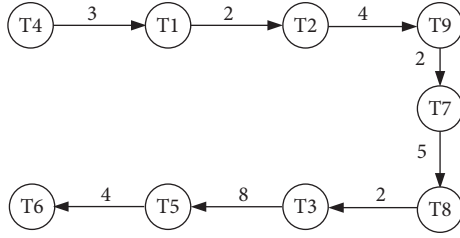


FIGURE 5: Linear generation graph.

```

do.
{
i = 1;
do.
{t = 0;
TVi = ((TV1 << L - i) | (TV1 >> i));
do.
{t++;
j = 2;
TV = TVi ⊕ TVj;
TVB = TVi ⊙ TVj;
if (∑k=0L-1TVk = 0 || ∑k=0L-1TVBk = 0).
{
List1.append (TVj);
List3.append (t);
List4.append (j);
t = 0;
break;
}
j++;
}while (j ≤ Num);
i++;
}while (i ≤ L);
m = len (list3);
list3.append (L - ∑x=1m list3[x]);
y = max {list3 [x] | 0 ≤ x ≤ m};
z = list3.index (y);
selects list3 [z - 1] as seed test pattern;
repeat the above calculation.
}while (len (list2) ≠ 0);

```

2.3. *Encoding Modes And Applications.* There are 9 test patterns as shown in Table 1, each of which is 40 bits, and all the 9 test patterns are indirectly compatible or backward-compatible. During the test, subsequent test patterns will be generated in the sequence of linear generation graph. $T1$ and

its subsequent test patterns move clockwise by one bit every time, find out whether there is a test pattern that is indirectly compatible or backward-compatible with it.

When $T1$ moves clockwise by 2 bits, it is found that the newly formed test pattern is backward-compatible with $T2$, and $T2$ is indirectly backward-compatible with the first test pattern $T1$ at first. Therefore, $T2$ is encoded as 100010.

$T2$ continues to move clockwise, when the $T2$ cycle moves clockwise by 4 bits, it is found that the newly formed test pattern is directly compatible with $T9$ at first, and $T9$ is indirectly compatible with $T2$, $T9$ is encoded as 110100.

And so on, $T7$, $T8$, $T3$, $T5$, $T6$, and $T4$ are encoded as 100010, 110101, 110010, 111000, 110100, and 11+ seed, respectively. Selects the first test pattern $T4$ as the seed test pattern, after removing the edge with the largest weight 10.

During the test, according to the sequence of the linear generation graph, $T4$ moves clockwise by 3 bits to get $T1$, and $T1$ generates $T2$, until $T6$ is generated. The sum of moving bits is 31, that is, the sum of weights is 31, which is less than the test pattern length. Generally, the number of clock cycles for test pattern generation in the linear generation graph is far less than the length of a single test pattern.

3. Proposed Decompression Architecture

The decompression architecture of this scheme is mainly composed of a control Unit, MUX, inclusive-OR gate. As shown in Figure 6, en is input enable signal, bit_in is used to transmit compressed test data including the mode, the seed and the encode to the control Unit. $Mod1$ is the mux channel selection signal. Seed and Q are connected to the two input terminals of MUX, the input terminal Q is connected to the end of scan chain, and the output terminal TD is connected to the head of scan chain. If $mod2$ is 1, $mod2$ is used to reverse the Q signal.

- (1) The test pattern enters the scan chain. The first input of MUX is gated, TD outputs seed, and the seed test pattern enters the scan chain, that is, bit_in inputs the seed test pattern into the scan chain through the seed channel. After each clock cycle, one bit of the test pattern is moved into the scan chain. After L clock cycles, the L bits of test pattern are moved in, that is, the whole seed test pattern is moved into the scan chain.
- (2) According to the test pattern encode, moving generates the next new test pattern. The second input of MUX is gated, TD outputs Q , and the original test pattern in the scan chain moves clockwise to generate a new next test pattern. In the process of moving clockwise, in the first clock cycle, the data in the flip-flop L enters the first flip-flop of the scan chain. Similarly, the data in flip-flop 1 enters flip-flop 2, the data in flip-flop 2 enters flip-flop 3, ..., and the data in flip-flop $L-1$ enters flip-flop L . After one clock cycle, all the test data moves one bit clockwise in the scan chain. Test data moves clockwise as many bits as the clock cycles.

TABLE 1: Compatibility analysis among test patterns.

Number	Test pattern	Bits	Mode1	Mode2	Encode
T4	01x10011011010001101010101001111001010	3	0	—	Seed
T1	0100x110011011010001101010101001111001	2	1	1	0011
T2	10101100011x010010111001010101010100001	4	1	0	0010
T9	000110101100011x010010111001010101010110	2	1	1	0100
T7	01111001010011100x1011010001101010101010	5	1	0	0010
T8	01010011x1001010011100110110100011010101	2	1	1	0101
T3	010101001x110010100111001101101000110101	8	1	1	0010
T5	001101010101x100111100101001110011011010	4	1	1	1000
T6	10100011010101x1010011110010100111001101	(10)	1	1	0100

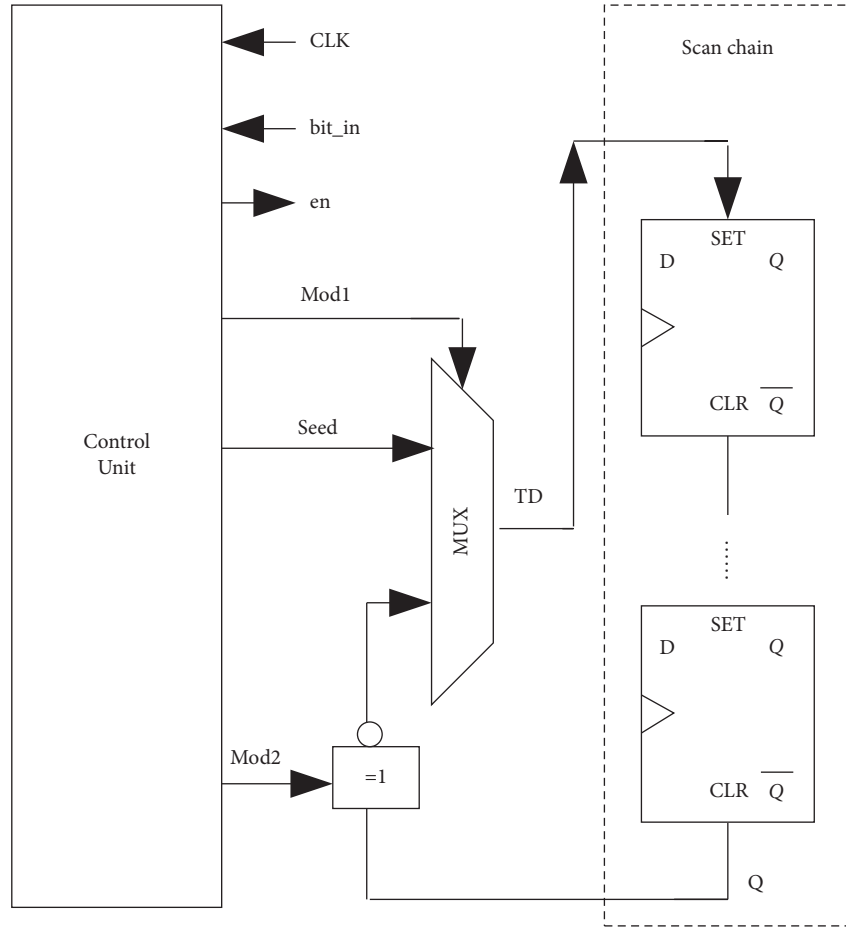


FIGURE 6: Decompression architecture.

- (3) Repeat (2), continuously generates the next test pattern until all the test patterns in the linear generation graph are generated according to to encode.

4. Experiments

4.1. Compression Analysis. Time complexity of constructing an annular generation graph is $O(N^*L)$, and the time complexity of constructing a linear generation graph is $O(L)$, then the time complexity of the whole algorithm is $O(L)$, and the algorithm is simple.

Obviously, the sum of weights $D[i][i+1]$ in annular generation graph is L , and the sum of weights in the linear

generation graph is less than L . Except for the seed test pattern, the number of clock cycles for test pattern generation in the linear generation graph is Nu .

$$Nu < \sum_{i=1}^N D[i][i+1] + (\log_2 L + 2) * N$$

$$Nu < L + (\log_2 L + 2) * N \quad (1)$$

$$\frac{nu < 1}{N} + (\log_2 L + 2).$$

For the experimental circuit, the values range of $\log_2 L$ is [5, 11], In formula (1), N is greater than 2, with the increase

TABLE 2: Experimental results.

Experimental circuit	FDR	ALT-FDR	EFDR	9C	BM	RL-HC	SHC	VIHC	GOLOMB	Proposed scheme
s5378	47.98	50.77	53.67	51.64	54.98	53.75	55.10	51.52	37.11	74.3
s9234	43.61	44.96	48.66	50.91	51.19	47.59	54.20	54.84	45.25	59.6
s13207	81.30	80.23	82.49	82.31	84.89	82.51	77.00	83.21	79.74	88.47
s15850	66.21	65.83	68.66	66.38	69.49	67.34	66.00	60.68	62.82	73.94
s38417	43.37	60.55	62.02	60.63	59.39	64.17	59.00	54.51	28.37	63.1
s38584	60.93	61.13	64.28	65.53	66.86	62.40	64.10	56.97	57.17	69.72
Average	60.01	60.58	63.55	63.35	65.48	62.72	63.28	61.44	56.42	71.52

of N , the range of average time μ decreases linearly, which is far less than 1, and the average number of test clock cycles of each test pattern is far less than the length of the test pattern. The compression ratio is calculated by $(T_D - T_E)/T_D \times 100$. $\mu = T_D - T_E$.

$$\mu = N * (L - 2 - \log_2 L) + \log_2 L + 1 - L. \quad (2)$$

The variation range of $\log_2 L$ is very small. Therefore, it can be seen from formula (2) that the compression ratio is determined by N , that is, the number of test patterns in the linear generation graph. Therefore, this paper is committed to solving the problem that makes more test patterns enter the linear generation graph.

$$\frac{d\mu}{dN} = L - 2 - \log_2 L. \quad (3)$$

Here, $d\mu/dN > 0$, it means that when L is constant and N increases successively, the compression ratio is improving constantly.

4.2. Experiment Comparison. Selected 6 large scale circuits from ISCAS'89 benchmark circuits as the experimental circuit as shown in Table 2, used atalanta to generate original test patterns which is will be encoded.

In order to prove the simplicity of decompression architecture, used the Design Compiler of Synopsys Company to synthesize and simulate, and calculated the area of the decompression architecture. The hardware area overhead of decompression architecture is $4684.32 \mu m^2$.

In order to evaluate the performance of this scheme in compression, compressed test patterns of ISCAS'89 benchmark circuits using Python. During the experiment, test patterns are continuously injected into the compression program module, during the test pattern injection process, the compression ratio keeps upward trends as shown in Figure 7. We can see that the test data compression scheme proposed in this paper is robust and has practical application value, meets the expectation of formula (1). The abscissa indicates the number of test patterns injected into the algorithm. The ordinate indicates the compression ratio.

The last column of Table 2 shows the compression ratio of this scheme. The compression ratio is better than other schemes, which proves that the scheme in this paper has good adaptability, and the maximum compression ratio reaches 88.47%.

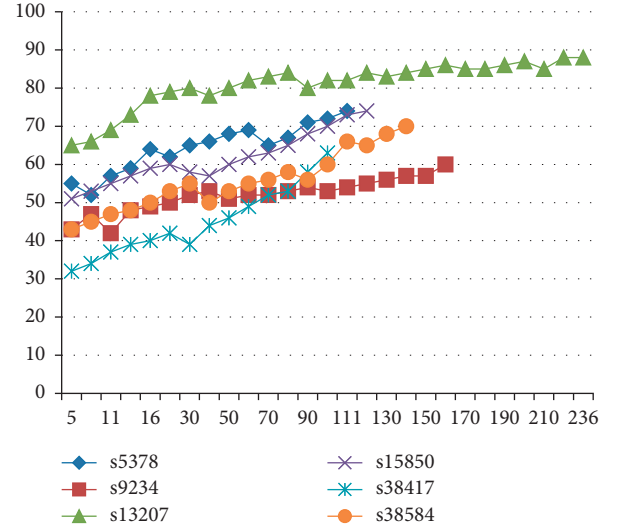


FIGURE 7: Compression ratio change curve.

5. Conclusions

In order to cover all faults of the test circuit, the corresponding bits of different test patterns are incompatible. This paper mainly explores the compatibility between different test patterns. Moves the test patterns in the scan chain clockwise, staggers the corresponding bits, and corresponding bits are no longer corresponding.

There are a large number of irrelevant bits in the test patterns, newly formed corresponding bits in test patterns are all compatible or backward-compatible after moving clockwise, eliminates the incompatibility between the original corresponding bits, solves the incompatibility between different test patterns, makes different test patterns have a new relationship which is indirectly compatible or backward-compatible, and provides a basis for test data compression.

Every time the seed test pattern moves 1 bit clockwise, find out whether there is a test pattern that is indirectly compatible or backward-compatible with it, test patterns that are indirectly compatible or backward-compatible with it will be constructed into the linear generation graph. According to the sequence of directed linear generation graph, the test pattern moves clockwise with the smallest amplitude, and the subsequent test patterns are generated quickly, reduces the time of test patterns generation and reduces the test application time effectively.

Data Availability

We evaluate the performance of the proposed scheme based on the ISCAS'89 benchmark circuits, and our model and related hyperparameters are provided in our paper.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Key Projects of Natural Science Research in Universities of Anhui Province: Research and Implementation of Decoding Unit for Ternary Optical Processor (Grant nos. KJ2020A0681 and KJ2019A0682), The Provincial Natural Science Research Program of Higher Education Institutions of Anhui Province (Grant no. KJ2021A1030), The Quality Improvement Project of Chaohu university on Discipline Construction (Grant no. kj21gczx03), Special Support Plan for Innovation and Entrepreneurship Leaders in Anhui Province, and Key Discipline of Computer Science and Technology of Chaohu University (Grant no. kj22zdxxk01).

References

- [1] G. Chandrasekaran, S. Periyasamy, and K. P. Rajamanickam, "Minimization of test time in system on chip using artificial intelligence-based test scheduling techniques," *Neural Computing & Applications*, vol. 32, pp. 5303–5312, 2020.
- [2] G. Chandrasekaran, "Test scheduling of system-on-chip using dragonfly and ant lion optimization algorithms," *Journal of Intelligent and Fuzzy Systems*, vol. 40, no. 3, pp. 4905–4917, 2021.
- [3] G. Chandrasekaran, S. Periyasamy, and P. R. Karthikeyan, "Test scheduling for system on chip using modified firefly and modified ABC algorithms," *SN Applied Sciences*, vol. 1, no. 9, p. 1079, 2019.
- [4] S. Huhn, D. Tille, and R. Drechsler, "Hybrid architecture for embedded test compression to process rejected test patterns," in *Proceedings of the European Test Symposium (ETS) 2019 IEEE European*, pp. 1–2, Baden Baden, Germany, May, 2019.
- [5] S. Rooban and R. Manimegalai, "Prediction of theoretical limit for test data compression," in *Proceedings of the International Conference on Recent Trends in Advance Computing (ICRTA-C)*, pp. 41–46, Chennai, India, November, 2018.
- [6] G. Chandrasekaran, V. Kumarasamy, and G. Chinraj, "Test scheduling of core based system-on-chip using modified ant colony optimization," *Journal Européen des Systèmes Automatisés*, vol. 52, no. 6, pp. 599–605, 2019.
- [7] G. Chandrasekaran, G. Singaram, R. Duraisamy, A. S. Ghodake, and P. K. Ganesan, "Test scheduling and test time reduction for SoC by using enhanced firefly algorithm," *Revue d'Intelligence Artificielle*, vol. 35, no. 3, pp. 265–271, 2021.
- [8] H. Yuan, K. Guo, X. Sun, J. Mei, and H. Song, "A power efficient BIST TPG method on don't care bit based 2-D adjusting and hamming distance based 2-D reordering," *Journal of Electronic Testing*, vol. 31, no. 1, pp. 43–52, 2015.
- [9] H. Vohra and A. Singh, "Test data compression using hierarchical block merging technique," *IET Computers & Digital Techniques*, vol. 12, no. 4, pp. 176–185, 2018.
- [10] M. Yi, H. Liang, and K. Zhan, "Optimal LFSR-coding test data compression based on test cube dividing," in *Proceedings of the International Conference on Computational Science and Engineering (CSE 2009)*, pp. 698–702, Vancouver, Canada, August, 2009.
- [11] S. Mitra and K. Kim, "XPAND: an efficient test stimulus compression technique," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 163–173, 2006.
- [12] Y. Yu, X. Peng, and Y. Zhang, "Test data compression method for multiple scan chains based on repeated subvectors," *Chinese Journal of Scientific Instrument*, vol. 30, no. 2, pp. 356–361, 2009.
- [13] M. Zhang, Z. Xia, and J. Kuang, "Improving compression ratios for code-based compressions by test set significant components based transform-decomposing method," *Journal of Electronic Measurement and Instrument*, vol. 34, no. 09, pp. 94–100, 2020.
- [14] H. Wu, W. Zhan, and Y. Cheng, "Dictionary encoding method based on independent of test data," *Journal of Electronic Measurement and Instrument*, vol. 30, no. 4, pp. 638–644, 2016.
- [15] W. Zhan and S. Zhao, "Compatible compression method for a maximum of approximately compatible grouping test patterns," *Journal of Electronic Measurement and Instrument*, vol. 30, no. 12, pp. 1933–1940, 2016.
- [16] A. Chandra and K. Chakrabarty, "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 3, pp. 355–368, 2001.
- [17] A. Chandra and K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes," *IEEE Transactions on Computers*, vol. 52, no. 8, pp. 1076–1088, 2003.
- [18] J. Kuang, Y. Zhou, and S. Cai, "Adaptive EFDR coding method for test data compression," *Journal of Electronics and Information Technology*, vol. 37, no. 10, pp. 2529–2535, 2015.
- [19] W. Zhan, Y. Cheng, H. Wu, and J. Jiang, "Automatic generation method of generalized folding set by guiding test patterns," *Journal of Southeast University (Natural Science Edition)*, vol. 48, no. 02, pp. 265–269, 2018.
- [20] M. Tehranipoor, M. Nourani, and K. Chakrabarty, "Nine-coded compression technique for testing embedded cores in SoCs," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 13, no. 6, pp. 719–731, 2005.
- [21] A. H. El-maleh, "Efficient test compression technique based on block merging," *IET Computers & Digital Techniques*, vol. 2, no. 5, pp. 327–335, 2008.
- [22] M. Nourani and M. H. Tehranipoor, "RL-huffman encoding for test compression and power reduction in scan applications," *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, no. 1, pp. 91–115, 2005.
- [23] A. Jas, J. Ghosh-dastidar, and N. G. Mom-eng, "An efficient test vector compression scheme using selective Huffman coding," *IEEE Transaction on. Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 797–806, 2003.

Research Article

Blockchain-Assisted Distributed Fog Computing Control Flow Attestation

Hongchao Li ¹, Tao Shen ¹, Fenhua Bai ¹ and Bei Gong²

¹Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Jingming South Street, Kunming 650093, Yunnan, China

²Faculty of Information Technology, Beijing University of Technology, Nanmofang Street, 100124 Beijing, China

Correspondence should be addressed to Tao Shen; shentao@kust.edu.cn

Received 3 May 2022; Revised 8 July 2022; Accepted 1 August 2022; Published 28 August 2022

Academic Editor: AnMin Fu

Copyright © 2022 Hongchao Li 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 control flow hijacking attack poses a serious threat to the integrity of the software. The attacker exploits the loophole to hijack the control flow of the running program to achieve the purpose of the attack. Remote control flow attestation is a method for embedded devices to ensure the integrity of the software. With the continuous development of Internet of Things (IoT) technology, embedded devices have exploded. None of the existing control flow attestation schemes can adapt to the real-time attestation requests of such massive embedded devices. This paper proposes a blockchain-assisted distributed fog computing control flow attestation scheme BDFCFA to deal with this scenario. The scheme uses a simplified control flow representation model, which can effectively represent the control flow of the program and reduce the runtime overhead of the prover in the attestation process. We use SGX technology to protect the integrity and confidentiality of verifier and prover data during the attestation process. Our proposed bidirectional control flow attestation protocol based on the elliptic curve can greatly protect the communication security between verifiers and provers without incurring excessive performance overhead and communication cost. We evaluate the performance of BDFCFA through the SNU real-time benchmark and demonstrate that BDFCFA has better performance. Finally, compared to the existing remote control flow attestation scheme, the results show that BDFCFA has the highest security.

1. Introduction

With the rapid development of IoT technology, a large number of embedded devices appear in our lives. A large part of it is deployed in critical information infrastructure and plays an important role. Once these embedded devices are maliciously attacked, it will pose a great threat to our lives. In recent years, control flow hijacking attacks have caused great security threats to embedded devices by tampering with the runtime behavior of programs. Control flow integrity (CFI) [1] was proposed to defend against this threat. The CFI obtains the control flow graph (CFG) of the program by analyzing the normal control flow of the program so that the control flow is transferred within the range limited by the control flow graph, and the execution process of the program is guaranteed to be safe and credible.

Remote attestation is a method of verifying the integrity of software on a remote device. It usually consists of two entities, a verifier who wants to know the state and one or more provers who provide reports of their state. Typically, there is agreement between a verifier and a prover. The verifier accepts a report of the hash value of the running state of the software to be executed signed by the security chip inside the device (such as the Trusted Platform Module, TPM) sent by the prover to verify whether the software state meets expectations. Remote attestation transfers the most expensive part of the entire attestation process to the verifier, thereby reducing the performance overhead of the prover.

In the early years, people used static measurements to verify the state of programs. In this way, the prover obtains a static measurement that is usually a signature or MAC calculated from the hash value of the program code and

sends them to the verifier. However, static attestation cannot detect return-oriented programming (ROP) [2] and jump-oriented programming (JOP) [3] in control flow hijacking attacks. The ROP attack is the most common code reuse attack today. It does not need to call the complete function block but calls the program code segment and the gadget that ends with the `ret` instruction in the dynamic link library code. By operating the program running stack, the program control flow is controlled so that the program jumps to the corresponding instruction segment when the execution function returns to achieve the purpose of attack. The JOP attack is achieved by using a chain of gadgets that ends with a `Jmp` instruction. In recent years, it has been proposed to measure and attest the integrity of the program runtime control flow through the prover [4–6] in order to more accurately verify the runtime program control flow. This adds overhead as it gets more contextual information at the basic block level. To balance runtime overhead and control flow security, some people have proposed a mutable control flow attestation scheme based on probability prediction [7] and a granularity adaptive control flow attestation scheme based on Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [8]. A log-based control flow attestation scheme [9] has also been proposed to deal with ROP [2] attacks and uses Physical Unclonable Functions (PUF) as a lightweight root of trust for the prover.

None of the above control flow attestation schemes can adapt to today's explosive growth of embedded devices. They are all single-server and single-database architectures that cannot handle real-time attestation requests from massive embedded devices, and because they are stand-alone databases, they cannot resist a centrally managed database being tampered with. Once the database is tampered with, it will affect the attestation results of the entire system. Moreover, the resources of the embedded devices are limited. Although MGC-FA [7] and GACFA [8] reduce runtime overhead to a certain extent, it comes at the cost of reducing control flow security. We use a simplified control flow representation model to effectively reduce the runtime overhead without reducing the security of the control flow, making the whole scheme more suitable for resource-constrained embedded devices. Moreover, the control flow attestation schemes mentioned above all use the challenge-response method for control flow attestation, and only perform one-way identity authentication, that is, the verifier authenticates the prover, which will cause great security risks.

In recent years, blockchain technology has been widely used in the Internet of Things [10, 11]. A blockchain is a distributed ledger distributed throughout a distributed system where multiple nodes maintain the same information without requiring a central authority. Therefore, this technology can not only mitigate the tampering attack of the centrally managed database but also reduce the communication overhead between the data center and the regional manager.

In this paper, we propose a blockchain-assisted distributed fog computing control flow attestation scheme to alleviate the above problems. The main contributions of this paper are as follows:

- (1) We propose a blockchain-assisted distributed fog computing control flow attestation scheme, which can adapt to today's explosively growing embedded devices and mitigate centralized database tampering attacks.
- (2) We use fog computing to deploy verifiers and blockchain nodes to the edge of the network, thereby reducing the communication overhead between verifiers and provers and improving the real-time nature of control flow attestation.
- (3) We propose a simplified control flow representation model by simply using the \langle source address, destination address, number of jumps \rangle of the jump instruction to represent the program control flow, thereby effectively representing the control flow of the program and reducing the runtime overhead of the prover during the attestation process.
- (4) We propose a lightweight bidirectional control flow attestation protocol based on elliptic curves, which can greatly ensure the communication security between the verifier and the prover and does not generate excessive performance overhead and communication cost.
- (5) We use SGX technology to protect the control flow remote attestation, and protect the integrity and confidentiality of the data of the verifier and the prover during the attestation process.

2. Related Work

2.1. Remote Attestation. Remote attestation obtains the running status of software on a resource-constrained prover through a resource-rich verifier. This reduces the attestation overhead for resource-constrained devices. The early remote attestation is mostly based on static, which can only protect the binary code when the program is started, but cannot prevent the control flow hijacking attack during the program execution. Later, the proposed C-FLAT [4] realized a more comprehensive runtime remote attestation of the program, and completed the work that could not be done by static remote attestation. The program control flow is protected from alteration by computing the cumulative hash of the basic blocks of program code. During the running of the program, all control flow instructions are intercepted by the runtime tracker trampoline and transmitted to the safe area for hash operation, and finally a hash value representing the execution state of the current program control flow is obtained. The resulting LO-FAT [5] uses a microcontroller to intercept instructions, instead of using a runtime tracker tool for software to intercept instructions in C-FLAT, allowing C-FLAT to be implemented with a lower performance overhead. ATRIUM [6] is a hardware-based runtime attestation protocol that not only checks the control flow of the program but also checks the specific instructions. It provides resilience against software- and hardware-based Time of Check Time of Use (TOCTOU) attacks while incurring minimal area and performance overhead. Although the hardware-based control flow attestation scheme reduces the

attestation overhead of the prover, the hardware implementation increases the cost and reduces the scalability of the scheme. MGC-FA [7] uses the probability model in machine learning to predict the fragile probability of each function in the program, thus distinguishing the normal part from the fragile part of the program. It is then subjected to lightweight coarse-grained checks and expensive fine-grained checks, respectively, balancing runtime overhead and control flow security. GACFA [8] optimizes the security and performance overhead of the attestation of the control flow of the program using the NSGA-II algorithm and then adaptively performs coarse- and fine-grained checks on the functions in the program according to the optimization results, thus reducing the performance overhead. Liu et al. [9] proposed to record the control flow through a program status log, and the verifier can effectively verify whether the target program has been damaged using the information in the log. At the same time, for more secure storage, they used a lightweight root of trust based on on-chip SRAM Physical Unclonable Functions in the attestation. However, this scheme log records function pointers and function return addresses, so it may be powerless against JOP attacks and attacks against decision data (branch variables or loop variables) in the program.

However, in today's environment of explosive growth of embedded devices, none of these solutions can adapt to new challenges—real-time attestation requests of massive embedded devices and attacks on centralized databases are tampered with. Hence, we use blockchain technology and fog computing to deal with these problems. Moreover, there are certain problems in their attestation protocols, so we propose a lightweight bidirectional control flow attestation protocol based on elliptic curve encryption.

2.2. Blockchain Technology. The blockchain is a chain data structure in which blocks are linked in chronological order and are protected by cryptographic algorithms, and the security of the ledger in the entire blockchain network is jointly maintained by means of distributed accounting. At present, according to the open authority of the network [12], the blockchain system can be divided into the public blockchain, the alliance blockchain, and the private blockchain. In this paper, we mainly study the consortium blockchain that is jointly managed by several different organizations or institutions. Unlike the public blockchain, the entities of the consortium blockchain are no longer a single individual but multiple organizations, so as long as most of the organizations in the consortium obtain a consensus, the data can be operated and managed. In a consortium blockchain, it is not the nodes participating in the chain that have permission to access data, but need to be preapproved by the institution to gain access, and because the consortium blockchain is semi-centralized, it is more efficient than the public blockchain.

In the past few years, due to the rapid development of blockchain technology, blockchain technology has been widely used in a large number of fields, such as finance, medical care, Internet of Things, edge computing, etc., for

example, the application of blockchain technology in the field of cross-data center authentication of vehicular fog services (VFSs) [13]. By effectively combining modern cryptography and blockchain technology, the communication between service managers during user authentication is eliminated because the records of all service managers are updated synchronously and can effectively resist the attack of tampered database managed by a central because the public ledger is maintained by all service managers. There are also applications of blockchain technology to distributed data systems IoT [14]. To solve the conflict between the operation performance and security of the blockchain system, the conflict between transparency and privacy, and the compatibility problem of a large number of IoT devices that run together, a distributed data system for IoT based on blockchain technology is proposed. It provides a new system architecture for different industrial IoT devices to deploy high-performance blockchain systems in many scenarios. There is also the application of blockchain technology to the radio frequency identification (RFID) supply chain authentication protocol in the 5G mobile edge computing environment [15]. By applying 5G and blockchain technology to the supply chain, the supply chain process can be simplified and allow automatic payment upon receipt of goods. It will help save the company millions of dollars in operating costs by eliminating the need for distributors who have to handle accounts receivable and pay bill with department personnel to track unpaid invoices. Additionally, this will also help avoid the legal fees that it may incur in disputes. However, our extensive literature survey shows that no one has applied blockchain technology to the realm of program control flow attestation. This article adopts blockchain as the database to improve the security and performance of BDFCFA.

Consensus algorithms are used mainly in distributed systems to ensure data consistency. The consensus algorithm used in blockchain is to solve the “block conflict” problem that may arise when a new transaction block is added to the blockchain. Nowadays, common consensus algorithms in blockchain include proof of work (PoW) [16], proof of stake (PoS) [17], DPoS (delegated proof of stake), practical byzantine fault tolerance algorithm (PBFT) [18] and RAFT algorithm [19], and so on. These five consensus algorithms have their own advantages, and their performance comparisons are shown in Table 1. First of all, since the management departments in BDFCFA are all credible, the consensus algorithm we choose does not need to consider Byzantine fault tolerance, only crash fault tolerance. Second, since the blockchain in BDFCFA is mainly managed by multiple management departments, the selected consensus algorithm does not need to have a high degree of decentralization. Finally, due to the high real-time requirement of control flow attestation, the selected consensus algorithm should have a lower communication complexity, faster verification speed, and higher throughput. After considering the Byzantine fault tolerance, crash fault tolerance, degree of decentralization, communication complexity, verification speed, and throughput of each algorithm, we chose the RAFT algorithm. Although the scalability of the RAFT

TABLE 1: Comparison of the consensus algorithm.

Consensus algorithm	PoW	PoS	DPoS	PBFT	RAFT
Byzantine fault tolerance	50%	50%	50%	33%	N/A
Crash fault tolerance	50%	50%	50%	33%	50%
Decentralization	High	High	Low	Low	Low
Resource consumption	High	Medium	Low	Low	Low
Scalability	Strong	Strong	Strong	Weak	Weak
Communication complexity	$O(n)$	$O(n)$	$O(n)$	$O(n^2)$	$O(n)$
Verification speed	>100 s	<100 s	<100 s	<10 s	<10 s
Throughput (TPS)	<100	<1000	<1000	<2000	>10 k

algorithm is not strong, its resource consumption is low, and its other properties are also very suitable for the high real-time and high security scenarios of control flow attestation.

2.3. Fog Computing. The concept of fog computing was first proposed by Cisco. By allowing devices deployed at the edge of the network to provide computing, storage, and network transmission services for terminal devices in a small area, the efficiency of data analysis and processing can be improved, and latency and network transmission pressure can be reduced. Today, the rapidly developing IoT faces many new challenges, such as strict latency requirements, network bandwidth constraints, resource-constrained devices, etc. [20], which cannot be adequately addressed by today's cloud computing and mainframe computing models alone. Therefore, to adapt to these new challenges, we adopt fog computing in BDFCFA to deploy verifier and blockchain nodes to the edge of the network, thereby improving the real-time performance of control flow attestation and alleviating the performance bottleneck of the central authority.

2.4. Intel SGX Technology. SGX (Software Guard Extensions), the Intel Software Guard Extensions, is a set of instruction sets supported by Intel since 2013 [21]. SGX provides an isolated Trusted Execution Environment TEE (Trusted Execution Environment) called enclave, which protects the safe operation and data of legitimate software from malicious attacks. No one, but the CPU, can access the code and data in it. SGX can now provide a trusted execution environment for many application scenarios, such as by introducing a trusted execution environment in edge computing, that is, software protection extension technology, to ensure the confidentiality of the medical IoT data analysis process [22]. There are also applications of SGX to verifiable confidential cloud computing, which guarantees code and data confidentiality, as well as the correctness and integrity of its results [23]. Confidentiality and integrity are preserved even when large components such as Hadoop, the operating system, and the hypervisor are compromised, and it outperforms Hadoop without SGX protection. There is also the use of SGX technology in the authentication of the IoT end device, enabling shielded execution of measurements and attestation procedures. The sensitive data in the authentication process are hidden through the specific key of the SGX enclave, which ensures the security of the sensitive data [24]. We enable SGX on the verifier and the prover, and

the verifier and the prover process the remote control flow attestation related data in the enclave area, such as attestation request, program control flow data, and attestation report. Because the data processing process is carried out in the enclave area provided by SGX, malicious programs outside the enclave area cannot view the data, let alone tamper with them.

2.5. Elliptic Curve. The application of elliptic curves in cryptography first appeared in 1986, proposed by Miller [25] and Koblitz [26] based on the elliptic curve logarithm problem. After that, elliptic curve cryptography (ECC) became a popular research direction in cryptography and gradually became the mainstream of public-key cryptography. The security of ECC relies on the discrete logarithm problem of the elliptic curve (ECDLP), that is, for two points P, Q on an elliptic curve over a finite field, solve k so that $k \cdot P = Q$ holds. On a classical computer, the time complexity of solving the discrete logarithm problem of the elliptic curve is exponential [27], which guarantees the security of the encryption of the elliptic curve. Compared to encryption techniques such as DSA, RSA, DH, etc., ECC is a more efficient security encryption technique that uses a shorter key to provide the same level of security. Therefore, ECC is often used in the authentication of the identity of devices constrained by resources, such as the Internet of Things. For example, Mahmood et al. [28] applied elliptic curve encryption to the communication authentication of smart grid. Their scheme not only provides mutual authentication with low computational and communication costs but is also resistant to most attacks, while also providing anonymity and privacy. However, Sadhukhan et al. [29] proposed an ECC-based lightweight remote user authentication scheme for IoT devices and remote user authentication, which uses three-factor authentication to protect user privacy and data confidentiality from multiple attacks. Rostampour et al. [30] proposed an efficient scheme to secure communication between IoT edge devices and cloud servers, where the authors used an ECC-based authentication protocol. We design a lightweight two-way control flow attestation protocol based on an elliptic curve. By using elliptic curve encryption, we can effectively ensure the security of communication between the verifier and the prover and at the same time reduce the performance overhead and communication cost as much as possible.

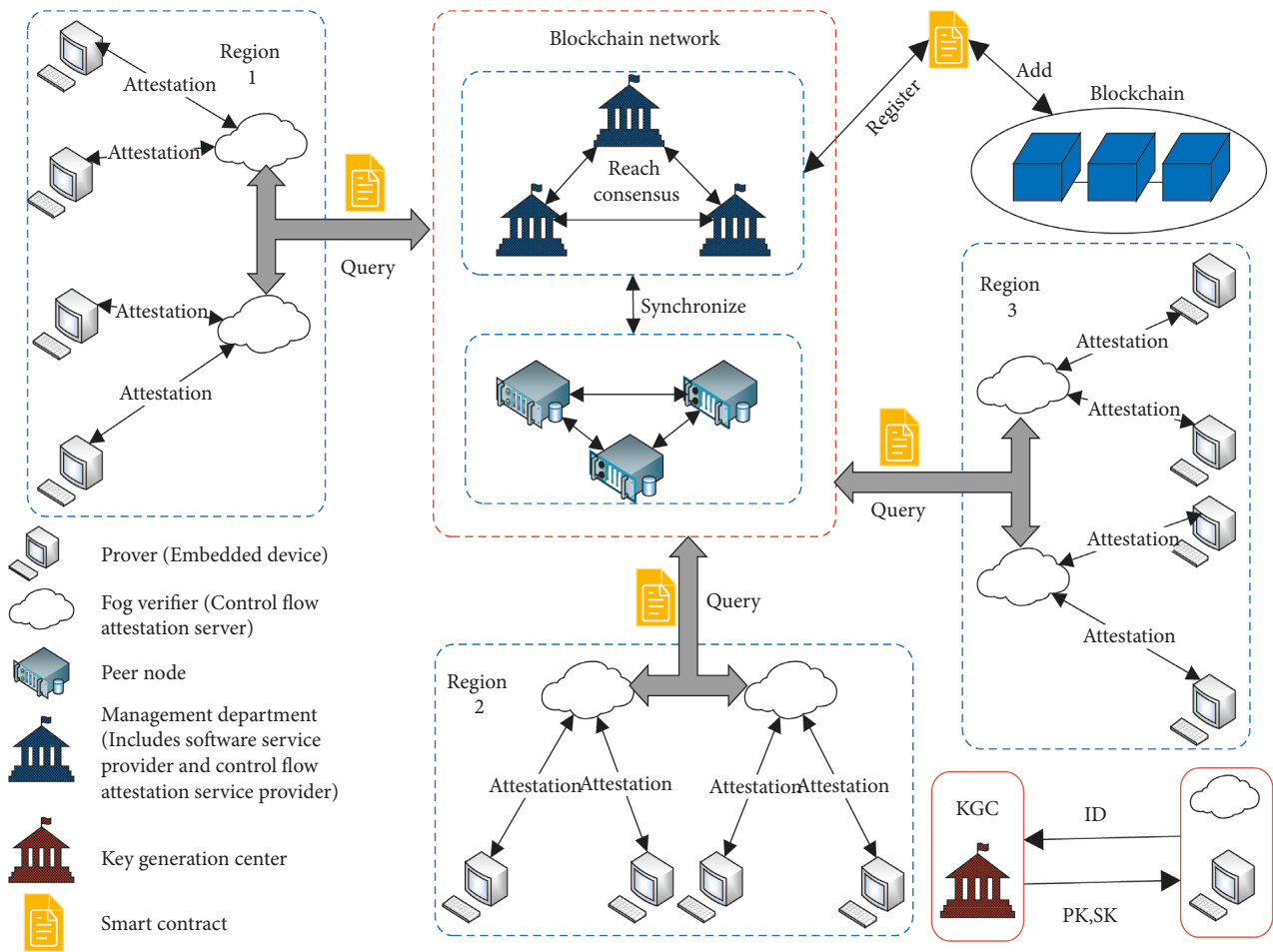


FIGURE 1: System framework.

3. Problem Statement and Assumptions

At present, the implementation of control flow hijacking attacks mainly includes controlled data attacks and non-controlled data attacks. Among them, control data attacks can be divided into code injection attacks and code reuse attacks according to the source of the attack code. In code injection attacks, the attacker uses the input operation of the software to inject malicious code into the memory space of the target software and then tampers with the control flow data to make the program jump to the malicious code. In code reuse attacks, attackers use existing instructions in programs and shared libraries to achieve attack goals, such as ROP [2] and JOP [3]. Non-controlled data attack is to modify the user identity data, configuration data, user input data, decision data, and other data in the program and then hijack the program control flow to complete the attack, such as (Data-oriented Programming, DOP) [31].

There are two assumptions about our scheme. First, we assume that the embedded device deploys the DEP (Data Execution Prevention) scheme that makes the executable area read-only, which is a built-in protection scheme commonly used on embedded platforms. Therefore, we do not consider code injection attacks. Second, we assume that attackers can carry out code reuse attacks and attacks on

decision data (branch variables or loop variables) in the program as well as replay attacks, man-in-the-middle attacks, and impersonation attacks in network attacks. However, the attacker cannot perform any other attack such as pure data attack and physical attack.

4. BDFCFA System Model

Figure 1 shows the system framework designed in this paper, which consists of multiple areas. Each zone includes a peer node, multiple fog verifiers, and multiple embedded devices. There are five types of entities in the system.

4.1. Management Department. The management department is a completely credible institution and is a member of the blockchain network. There are multiple management departments in the entire system. The management sector includes software service providers and control flow attestation service providers. The software service provider is responsible for registering programs that require attestation of the control flow. When registering a program, first instrument the target program, obtain the control flow data of each possible control flow path, and measure the control flow data of each possible control flow path, respectively, to

obtain the expected measurement value. Finally, all the expected measurement values of the program and the program input range corresponding to the expected measurement values are uploaded to the local blockchain through smart contracts, consensus is reached among various management departments, and finally the data are synchronized to each peer node,

$$\begin{cases} H(E_1) = H(E_1, H(0)), & n = 1, \\ H(E_n) = H(H(E_{n-1}), E_n), & n > 1, \\ H = H(H(E_n), F), & F = \{f_i | i \in n\}. \end{cases} \quad (1)$$

The measurement calculation formula is shown in equation (1), where H is the final measurement value, and $H(0)$ is the initial hash value, which is set by the management department, and is generally 0. E represents the jump edge in the control flow graph, the data in the edge includes the source address of the jump instruction and the destination address of the jump instruction, and $F = \{f_i | i \in n\}$ is the set of execution times of all the jump instructions in the current path (see Section 5.1 for details).

4.2. Key Generation Center (KGC). The key generation center is another trusted entity in the system, responsible for generating a public-private key pair for the fog verifier and prover in BDFCFA, which is used for identity authentication between a fog verifier and prover in control flow attestation. For the specific key generation process, see Section 5.3.

4.3. Peer Node. A peer node is a member of the blockchain network. In BDFCFA, not every fog verifier needs to maintain the blockchain ledger but establishes a peer node in an area to join the blockchain network, reducing the number of nodes in the blockchain network, thereby reducing the cost of deploying the infrastructure. Through smart contract settings, peer nodes cannot push data to the blockchain in the blockchain network but can only query the data on the blockchain, thereby reducing the possibility of peer nodes becoming malicious nodes. Peer nodes provide query services to fog verifiers by providing services.

4.4. Fog Verifier. The fog verifier provides program control flow attestation services for embedded devices within a certain physical range. Before performing control flow remote attestation, fog verifiers need to register with the key generation center. During attestation, the fog verifier needs to query the blockchain for all expected measurement values of the program to be verified and the program input range corresponding to the expected measurement values. The query process is shown in Figure 2.

First, the fog verifier sends a query request to the peer nodes in the area to which it belongs, and then randomly selects to send a query request to $2n$ nodes in the blockchain network. Finally, after receiving all the results of the request, the final result is determined according to most principles, and according to this result, we attest to the target program. There is a special case, where the randomly selected nodes

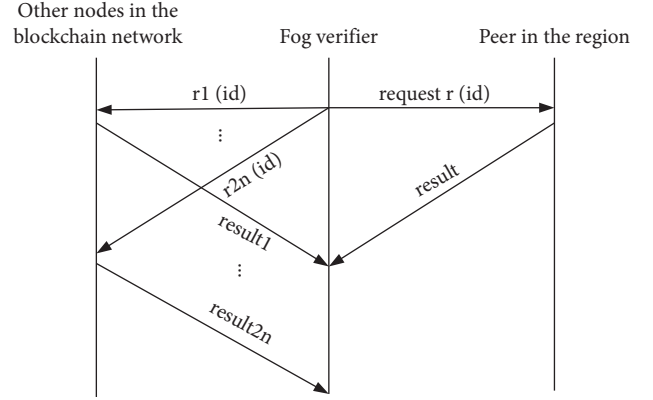


FIGURE 2: Query mechanism.

include the nodes of the management department, and the final result is the result of the node of the management department because the management department is a completely trusted authority. Through such a query mechanism, the harm of attackers tampering with a single database can be greatly alleviated.

4.5. Prover. The prover is an embedded device that needs to attest the control flow of the program running on it. It can be industrial control equipment and other important embedded devices deployed in critical information infrastructure. The program on the prover is provided by the software service provider. Before remote attestation of the control flow, it needs to be registered with the key generation center.

5. BDFCFA Design

5.1. Simplified Control Flow Representation Model. To authenticate the program control flow, the verifier needs to measure the program running path on the prover. However, it is obviously not feasible for the prover to directly transmit every executed instruction to the verifier. This requires the prover to store lengthy control flow data, resulting in a large performance overhead and attestation delay. To reduce the complexity of program control flow representation, thereby reducing the performance overhead and attestation delay of the prover, this paper proposes a simplified control flow representation model. The model is specifically defined as follows:

Definition 1. The simplified control flow representation model is a directed graph G that represents the control flow of a program, represented by a quadruple $\langle V, S, D, F \rangle$.

Definition 2. $S = \{s_i | i \in N\}$, S is the set of source addresses of all jump instructions in the directed graph G ; $D = \{d_i | i \in N\}$, D is the set of destination addresses of all jump instructions in the directed graph G ; and $F = \{f_i | i \in N\}$, F is the set of execution times of all jump instructions.

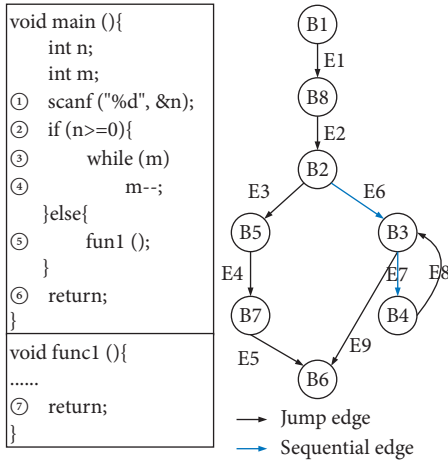


FIGURE 3: Control flow diagram for a simple program.

Definition 3. $V = \{v_i | i \in N\}$, V is the set of all vertices in the directed graph G , and a node represents a basic block. The basic block is a sequence of code instructions with only one entry and one exit without branching, so the basic block ends with a jump instruction. Hence, s_i is the exit of v_i and d_i is the entrance of v_{i+1} .

Definition 4. $E = \{v_i \rightarrow v_{i+1} | i \in N\}$, E is the set of all edges in the directed graph G ; there are two types of edges, jump edges and sequential edges. The jump edge indicates that the jump instruction at the exit of the basic block executes the jump, while the sequence edge indicates that the jump instruction at the exit of the basic block does not jump but executes the next instruction. A jump edge is represented by a two-tuple $\langle S, D \rangle$.

Theoretically, the two-tuple $\langle S, D \rangle$ can fully represent the control flow of a program. However, a large number of basic block jump edges will be generated by loops and recursive calls during program operation, resulting in a large number of repetitions of control flow data information and increasing performance overhead. Therefore, when we represent the control flow, we choose to use the triple $\langle S, D, F \rangle$ to represent it.

Taking Figure 3 as an example, we explain the control flow model and its measurement results.

We can see $V = \{B1, B2, B3, B4, B5, B6, B7, B8\}$ from Figure 3, where $B8$ is the basic block of the scanf function that contains the return instruction, and there is only one basic block of $B7$ in the func1 function. $E = \{E1, E2, E3, E4, E5, E6, E7, E8, E9\}$, where $E6$ and $E7$ are sequential edges.

As shown in Figure 3, the control flow graph has two paths, and their expected measurement results are as follows:

$$\begin{aligned}
 B1 \rightarrow B2 \rightarrow B5 \rightarrow B6: H_1 &= H(H(H(H(H(E1, H(0)), E2), E3), E4), E5), F_1), \\
 B1 \rightarrow B2 \rightarrow B3 \rightarrow B6: H_2 &= H(H(H(H(E1, H(0)), E2), E8), E9), F_2),
 \end{aligned}
 \tag{2}$$

where F_1 and F_2 are the set of execution times of all jump instructions in the two paths, respectively. Since in the second path, $E6$ and $E7$ are sequential edges and no jumps are performed, they are not added to the measurement calculation. The specific measurement calculation formula is mentioned in Section 4.

Then, we use the simple program shown in Figure 3 as an example to demonstrate the detection of five types of attacks, including the four types of attacks mentioned in Section 3:

- (1) ROP attack: the attacker tampered with the return address of the function func1 so that the jump edge $E5$, which should have returned $B7$ from the exit of the node to the entry of the $B6$ node, pointed to an illegal malicious code address. Therefore, the destination address of the edge $E5$ will be different from expected, thus detecting that the program has been attacked by control flow hijacking.
- (2) JOP attack: the attacker tampered with the jump address of the Jump class instruction so that the jump edge $E3$, which should have jumped from the exit of node $B2$ to the entrance of node $B5$, pointed to an illegal malicious code address. Therefore, the

destination address of the edge $E3$ will be different from expected, thus detecting that the program has been attacked by control flow hijacking.

- (3) Branch variable attack: the attacker tampers with the prover's input so that the sequential edge that should be executed sequentially from node $B2$ to node $B3$ becomes a jump edge to jump to the entry of node $B5$, causing the control flow to enter an unexpected but legal path. However, the input of the prover does not conform to the program input range corresponding to this legal path, so it can also be detected that the program has been attacked by control flow hijacking.
- (4) Loop variable attack: the attacker altered the value of the loop control variable m , causing the number of loops to change, resulting in a different number of $E8$ than expected. Finally, it was detected that the program was attacked by control flow hijacking.
- (5) Function pointer attack: the attacker tampers with the code pointer so that the jump edge $E1$, which should be called from the exit of node $B1$ to the entry of node $B8$, points to an illegal malicious code

address. The destination address of the edge $E1$ will be different than expected, thus detecting that the program has been attacked by control flow hijacking.

In summary, our method can detect these five attacks.

5.2. SGX-Based Control Flow Attestation. As shown in Figure 4, SGX-based control flow attestation includes two roles of the fog verifier and prover. Each role is divided into a secure area and an insecure area. The SGX enclave is the security area inside the role. We deploy the request generation part and the report verification part of the fog verifier in the SGX enclave. Guarantee the security of the fog verifier request and report the verification process and key information during the remote attestation process of the control flow. At the same time, we deploy the runtime tracking part of the program in the unsafe area in the prover, and deploy the authentication, counting, measurement, and report generation part in the SGX enclave to protect the security of the attestation report, key information, and control flow measurement process.

First, the fog verifier generates the attestation request and session key in the request generator and sends the session key to the report verifier and then the attestation request to the prover through the network communicator. The request contains the timestamp, program id, program input in, authentication information, and auxiliary information of encrypted session key. After the prover's network communicator receives the attestation request, it forwards the request to the authenticator. The authenticator first verifies whether the timestamp meets a certain threshold, otherwise the session is terminated. Then verify the identity information of the fog verifier and ensure that the verification does not pass the session termination. Finally, the auxiliary information of the session key will be decrypted through the attestation request to calculate the session key, which will be forwarded to the report generator, and the runtime tracker will execute the corresponding program and instrument program according to the program id and program input in the attestation request. When the application runs to the jump instruction, the interceptor in the runtime tracker will intercept it, and then judge whether the instruction is executed for the first time; if so, send the instruction address and destination address to the measurer. The measurer performs hash operation according to formula (1); if not, it sends a signal that the instruction count is incremented by one to the counter, and the counter increments the execution times of the target instruction by one according to the signal. When the program ends, the counter sends the set of execution times of the jump instruction to the measurer for the final hash operation to obtain the final measure, which is then sent to the report generator. The report generator generates an attestation report and sends it to the fog verifier through the network communicator. The fog verifier's network communicator receives the attestation report and sends it to the report verification. The report verification first verifies whether the timestamp meets a certain threshold, otherwise the session is terminated. Then verify the identity information of the prover, and the

verification does not pass the session termination. Then use the session key calculated by the request generator to verify the information in the attestation report, and if the verification fails, the session is terminated. Finally, query the blockchain for the expected measurement value corresponding to the input of the program to be verified and check whether the measurement value calculated according to the attestation report is the same as the expected measurement value. The specific calculation process in the above process is shown in Section 5.3.

The key modules are described in the following:

- (1) Request generator: this part uses the fog verifier's private key, the prover's public key, timestamp, program id, program input, and a random number to generate the attestation request and session key and send to the network communicator and report verification device, respectively.
- (2) Report verification: authenticates the attestation report received by the network communicator from the prover and verifies whether the control flow measurement value meets the expectations, thereby determining whether the program on the prover is subject to a control flow hijacking attack.
- (3) Network communicator: the part of the fog verifier and prover that performs network communication, providing network communication and data forwarding functions for modules in the SGX enclave.
- (4) Authenticator: the part of the prover that verifies the identity of the fog verifier. The identity of the fog verifier is verified through the attestation request, and the session key is further calculated to generate the attestation report.
- (5) Runtime tracker: this part is used to track the target application. We rewrite the Pin tool in Intel Pin-3.15 as a tool for intercepting jump instructions when the program is running.
- (6) Interceptor: intercepts the jump instruction when the target application is running, and judges whether it is a jump instruction that has been intercepted. If it is, it will send a signal to the counter that the number of executions of the instruction is incremented by one. If not, it will intercept the received address information of the instruction being sent to the measurer for measurement.
- (7) Counter: the part that obtains the execution times of the jump instruction and puts it in the enclave container to protect the execution times of the jump instruction from being tampered with. Because the address information will be hashed after interception and the set of execution times needs to be completed after the program runs, the counter is deployed in the safe area.
- (8) Measurer: when a hash operation is performed in the enclave container, the security of the measurement process is guaranteed. After the final measurement h is calculated, it is sent to the report generator.

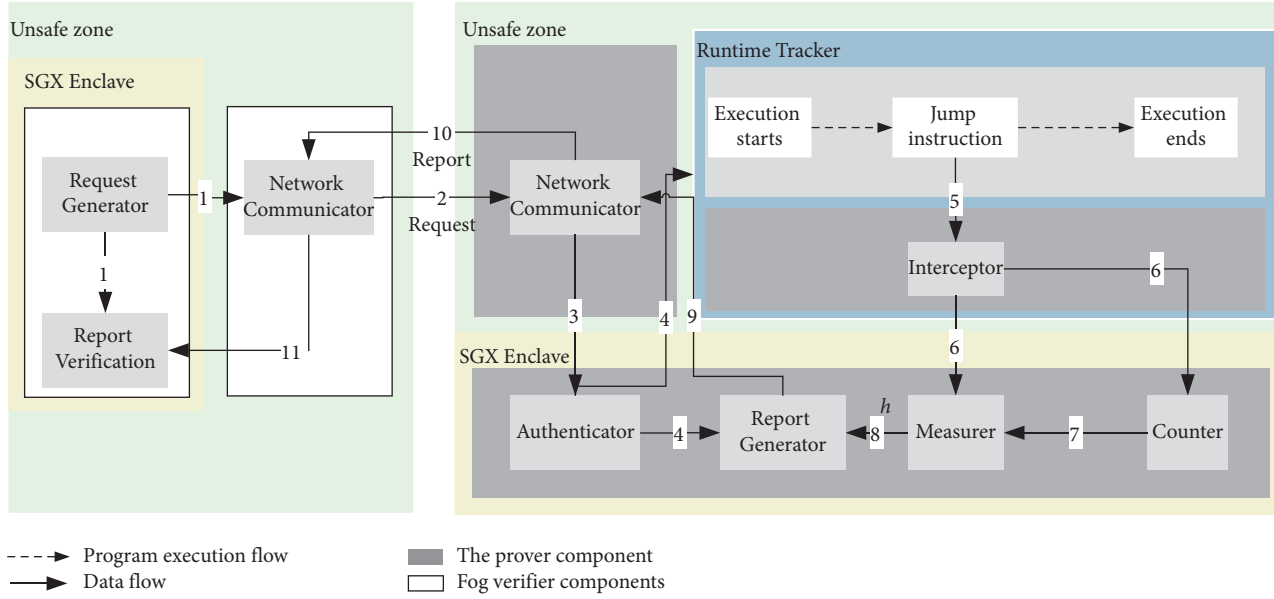


FIGURE 4: SGX-based control flow attestation architecture.

TABLE 2: Description of the notation.

Notations	Description
p, q	Large prime numbers
$\text{GF}(p)$	A finite field over prime p
$E_p(a, b)$	An elliptic curve over $\text{GF}(p)$
G	An additive cyclic group of points on the elliptic curves with prime order q
P	A generator of G
$k \cdot P$	Multiplication of elliptic curves, $k \cdot P = P + P + \dots + P$ (k times), where $k \in \mathbb{Z}_q^*$ and $P \in E_p(a, b)$
\mathbb{Z}_q^*	A finite additive group of nonzero integers modulo q
\mathbb{Z}_p	A finite additive group of integers modulo p
\parallel, \oplus	Data concatenation and bitwise XOR operations
$H1, H2, H3$	Secure hash function
ID_v, ID_p	Unique identities of the fog verifier and prover
PK_i	The public key of i
SK_i	The private key of i
$\text{Comp}(M, N)$	Compare if M and N are the same

- (9) Report generator: using the prover's private key, the fog verifier's public key, and the session key calculated by the authenticator, an attestation report is generated and sent to the verifier through the network communicator.

5.3. A Lightweight Bidirectional Control Flow Attestation Protocol Based on Elliptic Curve

5.3.1. Protocol Scheme Design. This section proposes a lightweight bidirectional control flow attestation protocol scheme based on elliptic curves. The scheme is divided into three stages: initialization stage, registration stage, and attestation stage. The initialization stage is executed only once by the KGC during the system establishment process. The symbols used in this paper are shown in Table 2.

- (1) Initialization stage: in this stage, the KGC performs the following operations to initialize a set of system parameters.
- (1) Set a system security parameter k and generate two large security prime numbers p and q .
 - (2) Choose an elliptic curve $y^2 \equiv x^3 + ax + b \pmod{p}$, denoted by $E_p(a, b)$, where $a, b \in \mathbb{Z}_p$ and $4a^3 + 27b^2 \pmod{p} \neq 0$.
 - (3) Choose an additive cyclic group G of order q , and choose a generator P for G , which contains the elliptic curve points defined by $E_p(a, b)$.
 - (4) Define the following secure hash functions: $H1: \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, $H2: \{0, 1\}^* \times G \rightarrow \mathbb{Z}_q^*$, and $H3: \{0, 1\}^* \rightarrow \{0, 1\}^*$.
 - (5) Public system parameter $(q, E_p(a, b), H1, H2, H3)$.

- (2) Registration stage: this stage completes the registration of fog verifiers and provers, as shown in Figure 5. It is executed only once in the lifetime of each fog verifier and prover. The fog verifier registration process will do the following. The prover registration process is similar to that of the fog verifier, which will not be described here.
- (1) The fog verifier uses the hardware information on the device (such as CPU serial number, motherboard serial number, hard disk serial number, physical MAC, etc.) to calculate its unique ID_V , and transmits it to the key generation center through a secure and trusted channel.
 - (2) After receiving the unique identity ID_V of the fog verifier, the key generation center randomly selects an integer R , connects ID_V and R , and uses the formula $SK_V = H1(ID_V || R)$ to calculate the private key of the fog verifier $SK_V, SK_V \in Z_q^*$. Then, the public key PK_V of the fog verifier is calculated by $PK_V = SK_V \bullet P$. Finally, the key generation center transmits PK_V and SK_V to the fog verifier using a secure and trusted channel.
 - (3) After the fog verifier receives the keys PK_V and SK_V generated by the key generation center, the registration process ends.
- (3) Attestation stage: this stage completes the attestation of the program control flow on the prover by the fog verifier, as shown in Figure 6. Among them, the fog verifier attests the operation of the program on the prover through a series of operations and judgments, that is, judges whether the running path of the program on the prover is safe and credible. The attestation protocol stipulates that both the fog verifier and the prover can access the binary of the program, and through the traditional static attestation protocol, the program being executed on the prover is guaranteed to be unmodified and complete. The specific process of the attestation stage is as follows:
- (1) The fog verifier first calculates the shared key DHK used for identity authentication according to $DHK = PK_p \bullet SK_v$, and then randomly generates a number n , $n \in Z_q^*$. N is calculated by elliptic curve multiplication $N = n \bullet P$, and session key S is calculated using $S = N \bullet SK_v$. Use $h1 = H3(in || T || id)$ to hash the program id, program input in , and current timestamp $T1$ to get $h1$. Then calculate the XOR of $h1$, n and DHK , $C1 = h1 \oplus n \oplus DHK$, and then perform hash operation $C2 = H2(C1, S)$ on $C1$ to get $C2$. Finally, the input in of the program, timestamp $T1$, program id, and identity of the authentication information $C1$ and $C2$ are spliced together to form an attestation request Req , which is sent to the prover.
 - (2) After the prover receives the attestation request Req from the fog verifier, it first checks the freshness of the timestamp. If the difference between timestamps exceeds a certain threshold, the session abruptly ends. Then use the secure hash function $H3$ to calculate $h1 = H3(in || T || id)$, and then calculate the shared key DHK' used for identity authentication through $DHK' = PK_v \bullet SK_p$. Through $n' = C1 \oplus h1' \oplus DHK'$, n' is calculated, which is used to calculate the session key. Use $S' = n' \bullet PK_v$ to calculate the session key S' , and then perform a hash operation $C2' = H2(C1, S')$ on $C1$ to obtain $C2'$. Then $Comp(C2, C2')$ compares whether $C2$ and $C2'$ are the same; if not, the session is terminated. So far, the prover has completed the identity authentication of the fog verifier. Then the prover executes the target program aid according to the program id and the program input in and obtains the address information $[Src_0, Dest_0], \dots, [Src_n, Dest_n]$ and the set F of the execution times of the jump instruction. Then, according to formula (1), the secure hash function $H3$ is used to calculate the cumulative hash value h through the address information $[Src_0, Dest_0], \dots, [Src_n, Dest_n]$ of the jump instruction and the set F of execution times of the jump instruction. Then use $hs = h \oplus S'$ to encrypt the final measurement value h to calculate hs , and then perform hash operation $Cr = H2(T2 || h, DHK')$ on the timestamp $T2$ and the final measurement value h to obtain Cr . Finally, the timestamp $T2$, the encrypted measurement value hs' and the identity authentication information Cr are spliced together to form an attestation report Rep , which is sent to the fog verifier.
 - (3) After the fog verifier receives the attestation report, it first checks the freshness of the timestamp. If the difference between timestamps exceeds a certain threshold, the session abruptly ends. Then use the session key S to decrypt the final measurement value h' according to $h' = hs \oplus S$, then use the secure hash function $H2$ to calculate $Cr' = H2(T2 || h', DHK)$ to get Cr' , then $Comp(Cr, Cr')$ compares Cr and Cr' to see if they are the same, and if they are different, the session is terminated. So far, the fog verifier has completed the identity authentication of the prover. Then the fog verifier queries the blockchain for the expected measurement value $h_{expected}$ corresponding to the input in of the program to be verified, and finally $Comp(h_{expected}, h')$ compares whether $h_{expected}$ and h' are consistent; if they are consistent, it means the target program aid is not subject to control flow hijacking attacks.

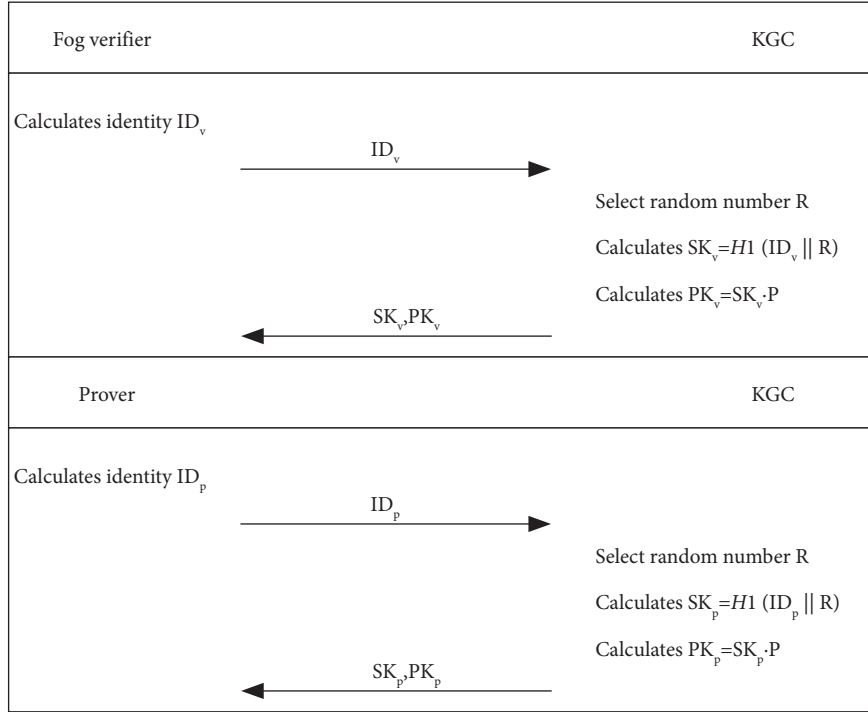


FIGURE 5: Registration process.

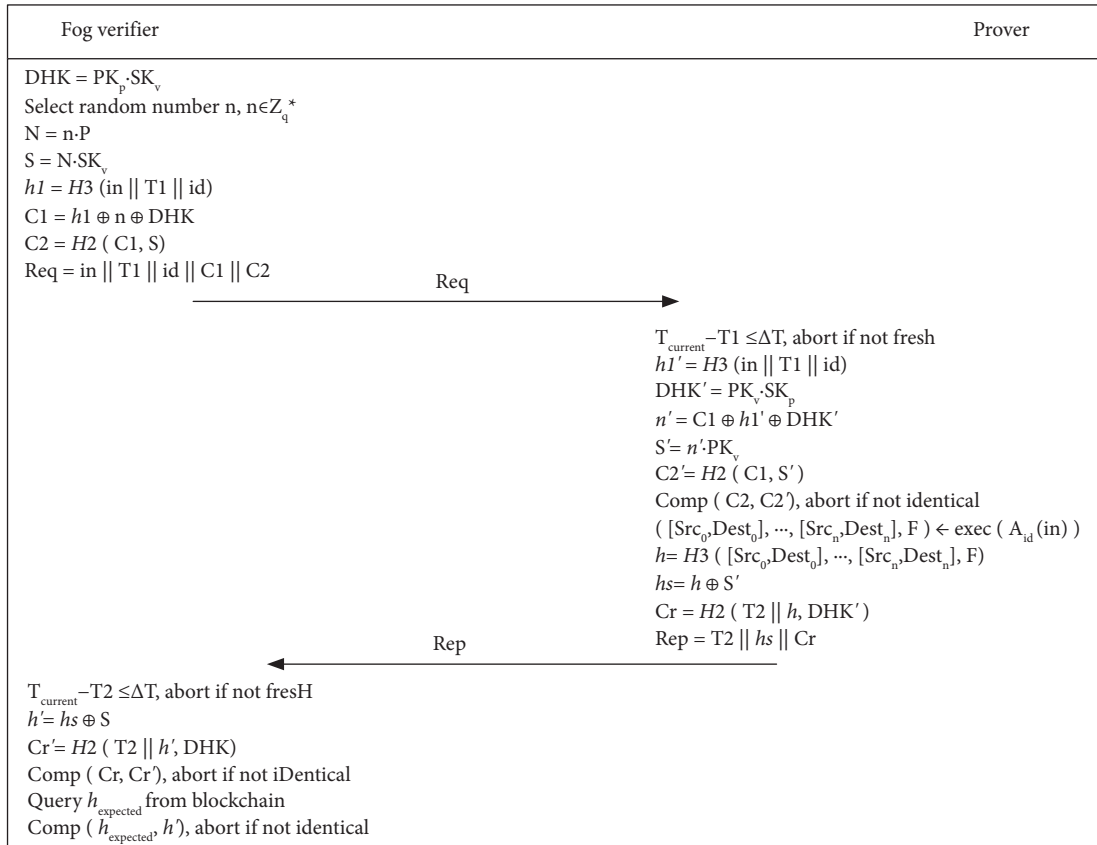


FIGURE 6: Attestation process.

TABLE 3: Parameters of the Hyperledger Fabric.

Consensus algorithm	Batch timeout (s)	Maximum message count	Block size (kB)	Number of order nodes	Number of peer nodes
RAFT	2	10	512	3	7

TABLE 4: Query time of the query mechanism under different peer nodes.

Query number of peer nodes	1	3	5	7
Average query time (ms)	11.894	23.976	37.895	51.682

TABLE 5: Runtime performance comparison of the two schemes.

Program	BDFCFA		Control flow events for MGC-FA			Control flow events for GACFA
	Control flow events	Runtime overhead (ms)	$p=0$	$p=0.3$	$p=1$	
adpcm-test.c	509	317.630	2×106	1.1×105	1×105	151
fft1k.c	329	18.392	3.6×105	2.7×105	3.3×104	49
fir.c	338	69.569	7800	2400	734	61
lms.c	353	24.363	1.2×105	4.4×104	8912	73
ludcmp.c	367	42.759	571	410	14	17
qurt.c	248	17.900	248	202	14	15
minver.c	310	16.204	310	300	6	25
fft1.c	381	31.720	884	776	136	58
sqrt.c	296	13.189	296	4	4	10

5.3.2. *Protocol Security Analysis.* The lightweight bidirectional control flow attestation protocol based on the elliptic curve in this paper has the following security features:

- (1) Two-way authentication: the protocol proposed in this paper realizes two-way authentication between the fog verifier and the prover, and the identities of both parties are authenticated before the control flow attestation. Both the fog verifier and the prover use their own private key and the other party's public key when calculating the shared key DHK . It is almost impossible for an attacker to calculate the shared key DHK only knowing their public key. The prover verifies the fog verifier by verifying whether $C2$ and $C2'$ are the same. The fog verifier verifies the prover by verifying whether Cr and Cr' are the same. The calculation of $C2$ and Cr will use DHK . Therefore, as long as the public key infrastructure is secure, only legitimate fog verifiers and provers can perform control flow attestation.
- (2) Anti-impersonation attack: the private key SK_V of the fog verifier is strictly kept secret. Although the public key PK_V and the generator P are open to the public, it is extremely difficult to calculate the private key SK_V from the public key PK_V and the generator P , which belongs to solving the elliptic-curve discrete logarithm problem. It is also extremely difficult for an attacker to directly calculate the private key SK_V through the ID_V of the fog verifier. Because a random number R is added when calculating the private key SK_V , it is almost impossible to calculate the private key SK_V without knowing the random number R . The prover is similar to the fog verifier, which will not be described here. Therefore, in the protocol proposed in this paper, attackers cannot

pretend to be fog verifiers and provers and can prevent impersonation attacks.

- (3) Anti-man-in-the-middle attack: when A and B communicate, the attacking host C becomes a forwarder in the middle, and the information between them is forwarded by C. C can not only eavesdrop on the communication of A and B but also tamper with the information. Then pass it on to the other party. We assume that the middleman C intercepts the attestation request $Req = in || T1 || id || C1 || C2$ and forwards it after maliciously tampering with the attestation request. If the middleman tampers with the data in in , $T1$ or id , then the prover will not be able to get the expected value by calculating $h1' = H3(in || T || id)$, resulting in failure to compare $C2$ and $C2'$. If the middleman tampered with the data of $C1$, the prover will not be able to calculate the correct n' through $n' = C1 \oplus h1' \oplus DHK'$, resulting in the termination of the session. If the middleman tampered with the data of $C2$, it will directly lead to the failure of comparing $C2$ and $C2'$. Similarly, if the middleman C intercepts the attestation report $Rep = T2 || hs || Cr$, and forwards it to the fog verifier after maliciously tampering with the attestation report. Assuming that the middleman has tampered with the data of $T2$, the fog verifier will make an error in calculating $Cr' = H2(T2 || h', DHK)$, resulting in a failure to compare Cr and Cr' . If the middleman tampered with the data of hs , the fog verifier will not be able to decrypt the final measurement value h' with $h' = hs \oplus S$, resulting in the termination of the session. If the middleman tampered with the Cr data, it will directly lead to the failure of comparing Cr and

Cr'. Therefore, for the protocol proposed in this paper, the attacker cannot achieve the purpose of the attack through the man-in-the-middle attack.

- (4) Anti-replay attack: the replay attack is that the attacker sends an authentication information that the destination host has received to deceive the other party. In the protocol proposed in this paper, both the attestation request and the attestation report contain the timestamp T , which is not only sent in clear text but is also hidden in C1 and C2 in the attestation request and Cr in the attestation report. Therefore, if an attacker replays the attestation request from the fog verifier or the prover's attestation report, the fog verifier and the prover can identify it by checking the freshness of T . If the attacker replaces a new timestamp T' in the attestation request or attestation report, the identity authentication of the prover or fog verifier will also fail. Because the prover makes an error when calculating C2, which causes the authentication to fail. The same applies to fog verifiers. The protocol proposed in this paper ensures that the attestation request or report of each transmission is different and there is no leakage of any valuable information, so the attacker cannot deceive the other party by replaying the intercepted message.
- (5) Known session key security: the fog verifier uses the random number n , $n \in Z_q^*$ to calculate N , and then uses the private key SK_v and N to calculate the session key S . Since n for each calculation of the session key is randomly selected, the attacker cannot obtain other session keys through the known session key. It is not even possible to obtain any valid information from the known session key because the session key needs to use the private key SK_v of the fog verifier in addition to N , and it is almost impossible to obtain any valid information only knowing S .

6. Evaluation

6.1. Query Mechanism Performance Evaluation. We use Hyperledger Fabric version 1.4.12 to build a blockchain network for query mechanism performance evaluation. The configuration of the blockchain network built by Hyperledger Fabric is shown in Table 3.

In this network, we use fog verifiers to randomly query 1, 3, 5, and 7 peer nodes 45 times, respectively, and get the average query time, as shown in Table 4. It can be seen that although the increase in the number of query peer nodes will increase the query time, in fact, compared with the entire attestation process, this overhead is not large. During the attestation process, we ask fog verifiers to randomly query 3 peer nodes in the blockchain network. The average attestation overhead of obtaining fog verifiers after 45 remote attestations is 29.50 ms. The overhead of attestation includes the cost of the fog verifier to generate the attestation request and the cost of the fog verifier to verify the attestation report, which includes the query time. This overhead is acceptable

for fog verifiers. In summary, our query mechanism does not incur too much overhead, while mitigating the harm of database tampering by attackers.

6.2. Runtime Overhead Evaluation. Since our BDFCFA scheme is aimed at embedded devices, to evaluate the performance of BDFCFA, we use the SNU real-time benchmark [32] to test BDFCFA. There are many C files in this benchmark, such as adpcm-test.c and fft1.c for embedded platforms. In addition, since the runtime overhead is mainly determined by the number of control flow events, we mainly compare the number of control flow events for each scheme when evaluating the runtime overhead.

The experimental results of the runtime overhead are shown in Table 5. We compare the MGC-FA [7] and GACFA [8] with our BDFCFA scheme. The experimental data on the runtime overhead of MGC-FA come from the literature [7]. The scheme with $p=0.3$ in MGC-FA is the scheme proposed in the literature [7]. The scheme when $p=0$ is the same as the control flow checking scheme in the literature [33], and the scheme when $p=1$ is the same as the control flow checking scheme in the literature [8]. Runtime overhead is the average value obtained after the experiment is repeated many times. As can be seen from the table, the number of BDFCFA control flow events is less than or equal to the number of control flow events when the probability threshold $p=0$ of the MGC-FA scheme. Among them, the number of control flow events of the three programs adpcm-test.c, fft1k.c, and lms.c is very different and is not of the same order of magnitude. Although the probability threshold for the MGC-FA scheme is $p=0.3$ or 1, there are some programs with fewer control flow events than our BDFCFA. However, when the probability threshold is $p=0.3$ or 1 of the MGC-FA schemes, they reduce the number of control flow events by sacrificing the control flow security, thereby reducing the runtime overhead. Because the number of control flow events directly affects the timeto check and measure control flow events, and the time overhead of checking and measuring control flow events accounts for the vast majority of the runtime overhead. Similarly, GACFA makes its control flow events smaller than BDFCFA on the premise of sacrificing control flow security. In summary, our BDFCFA scheme effectively reduces runtime overhead without sacrificing control flow security, making the whole scheme more suitable for resource-constrained embedded devices.

6.3. Attestation Protocol Performance Evaluation. Since our attestation protocol is used for control flow attestation, we compare the performance of the challenge-response based unidirectional control flow attestation protocol used in [4–8] with our elliptic curve-based bidirectional control flow attestation protocol. We use two attestation protocols separately in the BDFCFA scheme for comparison. Among them, the elliptic curve used by the bidirectional control flow attestation protocol based on the elliptic curve is 256 bits, and the secure hash function is also 256 bits. The challenge-response based unidirectional control flow attestation

TABLE 6: Comparison of communication cost theory.

Protocol	Communication costs	Communication bandwidth consumption (byte)
The elliptic curve-based bidirectional control flow attestation protocol	Req + Rep	144
The challenge-response-based unidirectional control flow attestation protocol	$C + R$	120

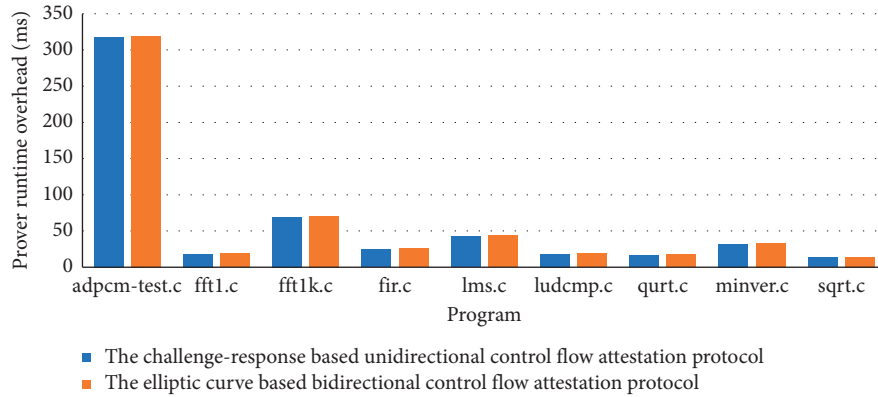


FIGURE 7: Comparison graph of runtime costs for both protocol provers.

protocol uses 256-bit elliptic curve based ECSDA and 256-bit hash function.

6.3.1. Communication Cost Performance Analysis. First, we theoretically analyze the communication cost of the two protocols. In the challenge-response-based unidirectional control flow attestation protocol, the verifier first needs to send a challenge C containing the program id, the program input in, and the random number N to the prover. The prover needs to return an attestation report R containing the expected measurement value h , the challenge C , and the digital signature S obtained by signing the expected measurement value and the challenge. In the bidirectional control flow attestation protocol based on the elliptic curve proposed in this paper, the fog verifier first needs to send an attestation request Req containing the input in of the program, the timestamp $T1$, the id of the program, and the identity authentication information $C1$ and $C2$. The prover needs to return an attestation report containing the timestamp $T2$, the encryption measurement value hs , and the identity authentication information Cr . The theoretical communication costs of the two protocols are shown in Table 6.

We assume that the size of program id, program input in, timestamp, and random number N are all 4 bytes, so the size of challenge C is 12 bytes. Since the bidirectional control flow attestation protocol based on an elliptic curve uses a 256-bit elliptic curve, the sizes of $C1$, $C2$, and Cr are all 32 bytes, so the size of the attestation request Req is 76 bytes and the size of the attestation report Rep is 68 bytes. Therefore, the communication bandwidth consumption of the bidirectional control flow attestation protocol based on the elliptic curve is 144 bytes. The challenge-response-based unidirectional control flow attestation protocol uses 256-bit elliptic

curve-based ECSDA and a 256 bit hash function, so the size of the attestation report R is 108 bytes. The communication bandwidth consumption of the challenge-response-based unidirectional control flow attestation protocol is 120 bytes. It can be seen from the data that the communication bandwidth consumption of the two protocols is almost the same.

6.3.2. Protocol Operational Efficiency Analysis. We use the SNU real-time benchmark [32] to compare the runtime overhead of the two protocols on the prover. By performing multiple experiments on nine programs selected in the SNU real-time benchmark and averaging them, we obtain a comparison chart of the runtime overhead of the two protocol provers, as shown in Figure 7. It is obvious from the figure that the prover's runtime overhead difference between the bidirectional control flow attestation protocol based on the elliptic curve and the unidirectional control flow attestation protocol based on the challenge response is very small, only one to two milliseconds. Therefore, the runtime overhead of the two protocols for the prover is almost the same.

We also compared the attestation time of the verifiers of the two protocols. The attestation time here refers to the time when the verifier generates the attestation request plus the time when the verifier verifies the attestation report, and does not include the time caused by the communication between the verifier and the prover. Any program in the SNU real-time benchmark will only generate an attestation request of the same size for the verifier and will only receive an attestation report of the same size. Therefore, we used the two protocols to perform 45 repeated experiments on the adpcm-test.c program in the SNU real-time benchmark test

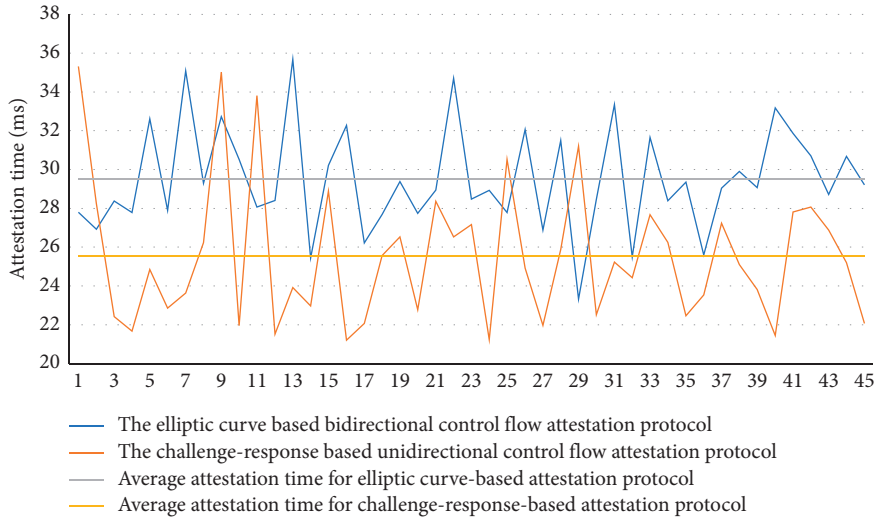


FIGURE 8: Comparison of the attestation time between the two protocol verifiers.

TABLE 7: Security comparison.

Security features	SF ₁	SF ₂	SF ₃	SF ₄	SF ₅	SF ₆	SF ₇	SF ₈	SF ₉	SF ₁₀
C-FLAT [4]	×	√	√	×	×	√	√	√	√	√
LO-FAT [5]	×	×	√	×	×	√	√	√	√	√
ATRIUM [6]	×	×	√	×	×	√	√	√	√	√
MGC-FA [7] $p=0$	×	√	√	×	×	√	√	√	√	√
MGC-FA [7] $p=1$	×	√	√	×	×	√	×	×	×	√
Liu et al. [9]	√	×	√	×	√	√	×	×	×	√
BDFCFA	√	√	√	√	√	√	√	√	√	√

√: the scheme supports features, ×: the scheme does not support features. SF1: Resist impersonation attacks, SF2: resist man-in-the-middle attacks, SF3: resist replay attacks, SF4: two-way authentication, SF5: known session key security, SF6: detect ROP attacks, SF7: detect JOP attacks, SF8: detect branch variable attacks, SF9: detect loop variable attacks, and SF10: detect function pointer attacks.

and obtained the comparison chart of the attestation time of the verifiers of the two protocols, as shown in Figure 8. It can be seen from the figure that the verifier attestation time of the unidirectional control flow attestation protocol based on the challenge response is shorter than that of the bidirectional control flow attestation protocol based on the elliptic curve. Among them, the average attestation time of the verifier of the bidirectional control flow attestation protocol based on the elliptic curve is 29.50 ms, and the average attestation time of the verifier of the unidirectional control flow attestation protocol based on challenge response is 25.53 ms. The difference between the two is close to 4 ms, and the difference is not big.

By analyzing the communication cost and operation efficiency of the two protocols, the communication cost and operation efficiency of our proposed bidirectional control flow attestation protocol based on the elliptic curve are very small compared to the unidirectional control flow attestation protocol based on the challenge response. However, our proposed protocol is more secure and can greatly ensure the communication security between the verifier and the prover. Therefore, our protocol is more suitable for program remote control flow attestation in the field of software security.

6.4. Security Performance Evaluation. We compare the security of our scheme BDFCFA with some control flow remote attestation schemes in recent years, as shown in Table 7. Since the security of the GACFA [8] scheme is related to specific procedures, there is no security comparison here. Both C-FLAT [4] and MGC-FA [7] can detect all control flow hijacking attacks proposed in the table when the probability threshold $p=0$. However, it is not resistant to impersonation attacks, there is no two-way authentication, and there is no way to maintain the security of the known session key. However, LO-FAT [5] and ATRIUM [6] cannot resist man-in-the-middle attacks except for the security features that C-FLAT does not have. Because the attestation protocols of these two schemes only digitally sign the random number in the attestation request when generating the attestation report, the attacker can completely intercept the attestation request and tamper with the id or input of the program to achieve the purpose of the attack. When MGC-FA has probability threshold $p=1$, in addition to not having the security characteristics of probability threshold $p=0$, it cannot detect the JOP attack, branch variable attack, and loop variable attack in control flow hijacking attacks. The reason for this is because MGC-FA only measures function pointers and function return addresses in the program when

the probability threshold $p = 1$. Liu et al. [9] used PUF as a lightweight root of trust for the prover, which can resist impersonation attacks and achieve known session key security. However, since it only digitally signs the log, it is also not resistant to man-in-the-middle attacks. Moreover, it does not perform two-way authentication and only records the function pointer and function return address in the program in the log, so only the ROP attack and function pointer attack in the control flow hijacking attack can be detected. Our scheme BDFCFA not only can detect all control flow hijacking attacks proposed in the table but also resist network attacks such as impersonation attacks, replay attacks, and man-in-the-middle attacks, and at the same time realizes two-way authentication, ensuring the security of known session keys, and each session key is different.

7. Conclusion

This paper proposes a blockchain-assisted distributed fog computing control flow attestation (BDFCFA), which can not only adapt to today's explosive growth of embedded devices, reduce the communication overhead between the verifier and the prover, and improve the real-time performance of control flow authentication but also mitigate the attack of the centralized database being tampered with. At the same time, we use SGX to protect the integrity and confidentiality of the verifier and prover data during the certification process. In addition, the query mechanism adopted by the fog verifier when querying the measurement data of the program from the blockchain network can spend less time overhead, thereby mitigating the harm of the attacker tampering with the database. Compared to MGC-FA [7], the simplified control flow representation model used in our scheme can effectively represent the control flow of the program under the premise of ensuring the security of the control flow, thus reducing the runtime overhead of the prover in the attestation process. Through comparative experiments with the challenge-response-based unidirectional control flow attestation protocol, it can be inferred that our proposed bidirectional control flow attestation protocol based on the elliptic curve can greatly protect the communication security between the verifier and the prover and does not generate excessive performance overhead and communication costs. This protocol is more suitable for program remote control flow attestation than the challenge-response-based unidirectional control flow attestation protocol used in the program control flow scheme by previous researchers. Finally, by comparing the security of BDFCFA with some remote control flow attestation schemes in recent years, it can be seen that the BDFCFA scheme has the highest security and can better protect the security of program control flow attestation. In summary, BDFCFA can adapt to today's explosive growth of embedded devices, improve the real-time performance of control flow attestation, alleviate the harm of attackers tampering with the database, reduce the runtime overhead of the prover during the attestation process, and greatly protect the security of the communication between the verifier and prover, and does not produce excessive performance overhead and communication cost.

In the future, we will study remote control flow attestation based on the combination of dynamic and static measurements. Because of the current runtime control flow measurement, only the control flow data when the program is dynamically running are measured. However, if the attacker tampers with the binary code of the program on the premise of ensuring the original control flow, the existing control flow remote attestation will not be able to detect this behavior.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare no conflicts of interest in this work.

Acknowledgments

This work was supported in part by the Major Scientific and Technological Projects in Yunnan Province under Grant 202002AB080001-8, the Yunnan Key Laboratory of Blockchain Application Technology under Grant 202105AG070005 and Projects YNB202109 and YNB202115, the Scientific Research Fund Project of Yunnan Provincial Department of Education under Grant 2022Y160, the National Natural Science Foundation of China under Grant 61971208, the Yunnan Reserve Talents of Young and Middle-Aged Academic and Technical Leaders (Shen Tao) under Grant 2019HB005, and the Yunnan Young Top Talents of Ten Thousands Plan (Shen Tao, Zhu Yan, Yunren Social Development) under Grant 2018 73.

References

- [1] M. Budiu, U. Erlingsson, and J. Ligatti, "Control-flow integrity," in *Proceedings of the ACM Conference on Computer & Communications Security*, p. 340, Alexandria VA USA, November 2005.
- [2] H. Shacham, "The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86)," in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 552–561, Alexandria, VA, USA, November 2007.
- [3] T. Bletsch, X. Jiang, V. W. Freeh, and Z. Liang, "Jump-oriented programming: a new class of code-reuse attack," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011*, pp. 30–40, ACM, Hong Kong, China, March 2011.
- [4] T. Abera, N. Asokan, L. Davi et al., "C-FLAT: control-flow attestation for embedded systems software," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 743–754, Vienna, Austria, October 2016.
- [5] G. Dessouky, S. Zeitouni, T. Nyman et al., "LO-FAT: low-overhead control flow attestation in hardware," vol. 24, pp. 1–24, in *Proceedings of the 54th Annual Design Automation Conference 2017, DAC 2017*, vol. 24, pp. 1–24, ACM, Austin, TX, USA, June 2017.
- [6] S. Zeitouni, G. Dessouky, O. Arias et al., "ATRIUM: runtime attestation resilient under memory attacks," in *Proceedings of*

- the 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 384–391, Irvine, CA, USA, November 2017.
- [7] J. Hu, D. Huo, M. Wang, Y. Wang, Y. Zhang, and Y. Li, “A probability prediction based mutable control-flow attestation scheme on embedded platforms,” in *Proceedings of the 2019 18th IEEE International Conference on Trust, Security and Privacy Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 530–537, Rotorua, New Zealand, August 2019.
 - [8] J. Zhan, Y. Li, Y. Liu, H. Li, S. Zhang, and L. Lin, “NSGA-II-Based granularity-adaptive control-flow attestation,” *Security and Communication Networks*, vol. 2021, Article ID 2914192, 16 pages, 2021.
 - [9] J. Liu, Q. Yu, W. Liu, S. Zhao, D. Feng, and W. Luo, “Log-based control flow attestation for embedded devices,” in *Cyberspace Safety and Security. CSS 2019*, J. Vaidya, X. Zhang, and J. Li, Eds., vol. 11982, Cham, Springer, 2019 Lecture Notes in Computer Science.
 - [10] H. -N. Dai, Z. Zheng, and Y. Zhang, “Blockchain for Internet of Things: a survey,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019.
 - [11] Q. Wang, X. Zhu, Y. Ni, L. Gu, and H. Zhu, “Blockchain for the IoT and industrial IoT: A review,” *Internet of Things*, vol. 10, pp. 1–9, 2020.
 - [12] V. Buterin, “On public and private blockchains,” 2015, <https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/>.
 - [13] Y. Yao, X. Chang, J. Mišić, V. B. Mišić, and L. Li, “BLA: blockchain-assisted lightweight Anonymous authentication for distributed vehicular fog services,” *IEEE Internet of Things Journal*, vol. 6, pp. 3775–3784, Article ID 2892009, 2019.
 - [14] N. Gao, R. Huo, S. Wang, T. Huang, and Y. Liu, “Sharding-hashgraph: a high performance blockchain-based framework for industrial Internet of Things with hashgraph mechanism,” *IEEE Internet of Things Journal*, 2021.
 - [15] S. Jangirala, A. K. Das, and A. V. Vasilakos, “Designing secure lightweight blockchain-enabled RFID-based authentication protocol for supply chains in 5G mobile edge computing environment,” *IEEE transactions on industrial informatics*, vol. 16, no. 11, pp. 7081–7093, 2020.
 - [16] M. Jakobsson and A. Juels, “Proofs of work and bread pudding protocols (extended abstract),” in *Secure Information Networks*, B. Preneel, Ed., vol. 23, Boston, MA, Springer, 1999 IFIP — the International Federation for Information Processing.
 - [17] A. Kiayias, A. Russell, B. David, and R. Oliynykov, “Ouroboros: A provably secure proof-of-stake blockchain protocol,” in *Advances in Cryptology – CRYPTO 2017. CRYPTO 2017*, J. Katz and H. Shacham, Eds., vol. 10401, Cham, Springer, 2017 Lecture Notes in Computer Science.
 - [18] M. Castro and B. Liskov, “Practical byzantine fault tolerance,” in *Proceedings of the third symposium on Operating systems design and implementation (OSDI ’99)*, pp. 173–186, USENIX Association, USA, 1999.
 - [19] D. Ongaro and J. Ousterhout, “In search of an understandable consensus algorithm,” in *Proceedings of the 2014 USENIX Annual Technical Conference*, Philadelphia, PA, USA, June 2014.
 - [20] M. Chiang and T. Zhang, “Fog and IoT: an overview of research opportunities,” *IEEE Internet of Things Journal*, vol. 3, pp. 854–864, 2016.
 - [21] 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.
 - [22] Y. Gao, H. Lin, Y. Chen, and Y. Liu, “Blockchain and SGX-enabled edge-computing-empowered secure IoMT data analysis,” *IEEE Internet of Things Journal*, vol. 8, pp. 15785–15795, 2021.
 - [23] F. Schuster, M. Costa, C. Fournet et al., “VC3: trustworthy data analytics in the cloud using SGX,” in *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, pp. 38–54, San Jose, CA, USA, May 2015.
 - [24] J. Wang, Z. Hong, Y. Zhang, and Y. Jin, “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, no. 1, pp. 88–96, 2018.
 - [25] V. S. Miller, “Use of elliptic curves in cryptography,” in *Advances in Cryptology – CRYPTO ’85 Proceedings. CRYPTO 1985*, H. C. Williams, Ed., vol. 218, Berlin, Heidelberg, Springer, 1986 Lecture Notes in Computer Science.
 - [26] N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.
 - [27] S. D. Galbraith and P. Gaudry, “Recent progress on the elliptic curve discrete logarithm problem,” *Designs, Codes and Cryptography*, vol. 78, no. 1, pp. 51–72, 2016.
 - [28] K. Mahmood, S. A. Chaudhry, H. Naqvi, S. Kumari, X. Li, and A. K. Sangaiah, “An elliptic curve cryptography based lightweight authentication scheme for smart grid communication,” *Future Generation Computer Systems*, vol. 81, pp. 557–565, 2018, Pages 557–565, ISSN 0167-739X.
 - [29] D. Sadhukhan, S. Ray, G. P. Biswas, M. K. Khan, and M. Dasgupta, “A lightweight remote user authentication scheme for IoT communication using elliptic curve cryptography,” *The Journal of Supercomputing*, vol. 77, no. 2, pp. 1114–1151, 2021.
 - [30] S. Rostampour, M. Saffkhani, Y. Bendavid, and N. Bagheri, “ECCbAP: a secure ECC-based authentication protocol for IoT edge devices,” *Pervasive and Mobile Computing*, vol. 67, Article ID 101194, 2020.
 - [31] H. Hu, Z. Leong Chua, S. Adrian, P. Saxena, and Z. Liang, “Automatic generation of data-oriented exploits,” in *Proceedings of the 24th USENIX Conference on Security Symposium*, pp. 177–192, Washington, D.C. USA, August 2015.
 - [32] SATABS, “SNU real-time benchmarks,” 2022, <http://www.cprover.org/goto-cc/examples/snu.html>.
 - [33] S. Das, W. Zhang, and Y. Liu, “A fine-grained control flow integrity approach against runtime memory attacks for embedded systems,” *IEEE Transactions on Very Large Scale Integration Systems*, vol. 24, pp. 3193–3207, Article ID 2548561, 2016.

Research Article

Detecting Privacy Leakage of Smart Home Devices through Traffic Analysis

Ting Yang ^{1,2}, Guanghua Zhang,³ Yin Li,² Yiyu Yang,² He Wang ¹
and Yuqing Zhang ^{1,2,4,5}

¹School of Cyber Engineering, Xidian University, Xian 710000, China

²National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing 101408, China

³School of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang 050091, China

⁴School of Computer Science and Cyberspace Security, Hainan University, Haikou 570228, China

⁵School of Information Science and Engineering, Yanshan University, Qinghuangdao 066000, China

Correspondence should be addressed to He Wang; hewang@xidian.edu.cn and Yuqing Zhang; zhangyq@ucas.ac.cn

Received 10 May 2022; Accepted 14 June 2022; Published 15 July 2022

Academic Editor: Shui Yu

Copyright © 2022 Ting 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.

Under the management of the Internet of Things platform, smart home devices can be operated remotely by users and greatly facilitate people's life. Currently, smart home devices have been widely accepted by consumers, and the number of smart home devices is rising rapidly. The increase of smart home devices introduces various security hazards to users. Smart home devices are vulnerable to side-channel attacks based on network traffic. The event of smart home devices can be identified by network surveillants. Given this situation, we designed a set of standardized workflows for traffic capturing, fingerprint feature extraction, and fingerprint event detection. Based on such workflow, we present IoTEvent, a semiautomatic tool to detect vulnerable smart home devices, which is not limited to specific types of communication protocols. IoTEvent first collects device traffic by simulating touch events for App. Then, it pairs the packet sequences with events and generates a signature file. We also test the usability and performance of IoTEvent on five cloud platforms of smart home devices. Finally, we discuss the reasons for privacy leakage of smart home devices and security countermeasures.

1. Introduction

With the progress of communication technology and network technology, the smart home market has developed rapidly. According to ABI research [1], almost 79 million homes will have a smart home device by 2024. The manufacturers of smart home devices are trying to make them “smart” by connecting them to the cloud platform, and then users can control them using mobile Apps or voice assistants. For example, if we want to listen music and say “Hey Google, play music,” a piece of wonderful music will be played by Google Home. Nowadays, smart home platforms are widely used in the process of device development, including Xiaomi [2] and Huawei [3].

However, the rapid growth of the market economy promotes the development of the manufacturing industry

toward product practicality and then ignores the safety of products, leading to a number of security vulnerabilities for smart home devices [4]. According to reports, Amazon's Alexa and Google's smart speakers can eavesdrop on users' information and even cheat them by voice. According to researchers, few consumers are aware that smart home devices collect and share private data as part of their normal operations [5]. Smart home cloud platform not only brings convenience to people but also has the risk of privacy leakage [6]. Large amounts of data from devices are collected by cloud platforms, which are transmitted through network traffic. For example, users can control the Mijia App to turn on a smart light bulb. In this process, the smart home cloud platform judges and recognizes the commands sent by the App and then forwards them to the smart light bulb. Although the control commands are encrypted, the commands

can be identified by encrypting traffic analysis. Furthermore, detecting the detailed activities of smart home devices is the premise of implementing attacks against smart home scenarios.

Many researchers [7–10] have noted that there is a link between encryption traffic and device state transitions, but their proposed method cannot accurately detect triggered events. A part of studies [11, 12] analyzed network traffic for specific types of devices. The author of [11] analyzed network traffic of the camera. The author of [12] detected users’ presence through traffic analysis of smart speakers. The author of [9] inferred device state transitions through Zigbee [13]/Z-wave [14] network traffic. The author of [15] proposed the network traffic packet signature of TCP-based devices. However, their tool cannot apply to UDP-based devices that follow a connectionless pattern.

In this article, we present IoTEvent to identify trigger events of smart home devices by fingerprinting encrypted traffic. Note that the user’s behavior privacy can be inferred from the detection result. The main contributions of this study are summarized as follows:

- (i) Design a trigger event detection tool IoTEvent. First, collect encrypted traffic of smart home devices. Then, train the signature event and generate classification model. Finally, based on the model detect the privacy events.
- (ii) Test our tool with nine popular devices from five popular cloud platforms. We observe a high accuracy of 99%.
- (iii) Analyze the reasons for privacy leakage of encrypted traffic, and then present some suggestions on how to deal with this problem.

This article is organized as follows. The threat model is described in Section 2. Then, we detail the design and implementation in Section 3 and present the evaluation results of devices in Section 4. In addition, we discuss the reason of the privacy leakage of smart home devices and how to avoid it in Section 5. In Section 6, we introduce related works. Finally, we conclude in Section 7.

2. Threat Model

There are four key entities in the process of communication: the smart home devices, the mobile App, the cloud, and the gateway, as shown in Figure 1. The privacy information can be leaked in the communication process by analyzing the timestamp, length, and direction of the data packet. We assume the adversary has two attack methods: WAN sniffing and WiFi sniffing, and knows the brand and model of the device that he wishes to passively monitor (the adversary may be the neighbor or the repairman that has been to the victim’s house). Through WiFi sniffing, the adversary can know the MAC addresses to identify which device has sent the traffic. Normally, home routers use NAT [16]: remapping all traffic to the router’s IP address. Through WAN sniffing, the adversary can know IP headers of all packets and

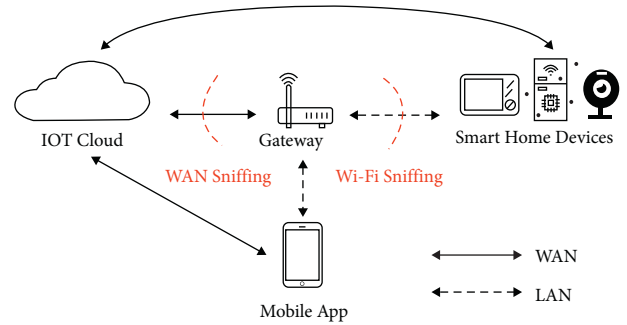


FIGURE 1: The communication modes of the smart home system. The LAN is the main communication mode between smart home devices and the cloud, and the WAN is rarely used. Mobile App control devices have two paths: when the user is at home, control commands are uploaded to the cloud via the LAN; when the user is not at home, control commands are uploaded to the cloud over the WAN.

then find the domain name that communicates with the device by IP address.

To sum up, the adversary should meet the following conditions:

- (i) The WiFi sniffer should be within the transmission distance of the wireless router. The WAN sniffer can capture all packets between the router and the cloud.
- (ii) The adversary can obtain the same smart home device and extract event signatures of the device

Smart home devices transfer packets to the router through the WiFi, and then the router transfers them to the cloud through the WAN. Before sniffing, the adversary needs to get the event’s signature of the same device. During sniffing, the adversary cannot obtain the plaintext data because the data are encrypted and then transmitted. After capturing the packets, the adversary first preprocesses the packets and then performs event detection based on the event’s signature of the same device.

3. Detailed Design

In this section, we present each step of IoTEvent and explain how to detect the triggered events of the device in detail.

3.1. Overview. For identifying the privacy events, we present IoTEvent to handle the challenges. Our tool can identify event containing out-of-order packets, which improve the accuracy of event detection. Figure 2 shows the workflow of IoTEvent. In the first step of network traffic collection, IoTEvent can simulate trigger events through scripts, which also record the name and timestamp of the triggered event. After receiving the control command, the device generates packet sequences in a short time. We use tcpdump to capture device traffic on the router. After this step, we collect traffic for different events to prepare for event signatures and data training. Next, in the step of data training, due to device events generating data packets in a short time, we propose a method to divide device traffic into packet sequences. Then,

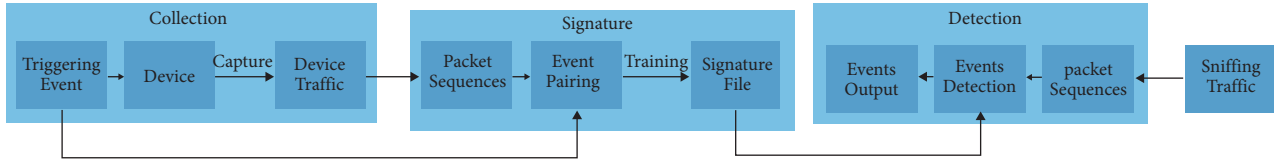


FIGURE 2: System architecture of IoTEvent.

IoTEvent pairs events and packet sequences through timestamp. Next, IoTEvent extracts the length and direction of the data packet to form an event signature. Through training, a signature file is generated to store event’s signature. In the last step of event detection, the sniffed traffic of the device is divided into packet sequences. The main task of event detection is to classify the packet sequences according to the signature file. Finally, the event’s name is output, which is also the result of classification.

3.2. Traffic Capturing. The main task of data collection is to capture the network traffic of the device and obtain the timestamp of the triggered events.

According to the survey, we find that the traffic between the device and the cloud includes the following three types: device heartbeat package, device report information, and device operation command. The device heartbeat packet is used to detect whether the device is disconnected. The cloud platform uses a simple communication packet to judge whether the device is running normally. If no response is received from the device within a specified period of time, the device will be judged to have dropped the line. The device report information includes device firmware information, device log information, and device physical state change information. The device operation command refers to the control command sent by the user to the device through the cloud.

The triggered events refer to the state change of the device by the user clicking or sliding the corresponding mobile App. For example, turn on the light by clicking button of the mobile App, as shown in Figure 3. The mobile App sends the control command of light turn on to the cloud. After the cloud receives the control command, it queries the status of the device. The cloud compares it with the status of the device in the command. If they are different, the cloud sends command of light turn on. The device executes the command and returns success message. If they are the same, the cloud does not send the command. Different device events may have different response processes, so the number of data packets generated during the event will also be different.

IoTEvent automatically generates and captures device traffic for the training set. We use the script of Auto.js [17] to trigger all events on the smartphone’s screen in turn, which runs on smartphone. All events refer to operations related to the device on the mobile App [18]. For example, in addition to basic control events (light on/off), Mi Control Gateway also includes setting events (volume settings). What we want is to trigger all device-related events on the App. The script is customized according to the device due to the different

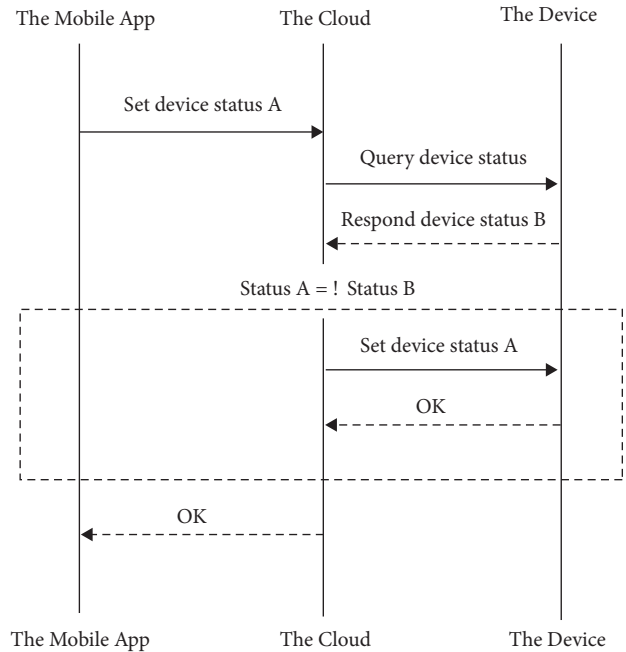


FIGURE 3: The device event and respond. The query (solid line) and the reply (solid line) of the event are shown in the figure.

functions of the device. Each event is triggered 100 times, and the interval between two clicks is 60 seconds. IoTEvent starts tcpdump to capture the device traffic before triggering the events. The network traffic includes all data packets between the device and the cloud platform, which are the result of event triggering, device logs, and so on. IoTEvent records the network traffic in a PCAP file and writes the name and timestamp of the trigger event in a text file.

3.3. Fingerprint Feature Extraction. The purpose of fingerprint feature is to generate a signature file of the device. IoTEvent divides the captured traffic into packet sequence and then matches the packet sequence and event according to the algorithm. The length and direction information of the data packet are extracted to generate feature vectors, which are trained to generate a classification model.

Due to the difference of SDK between cloud platforms or the difference of the communication protocol of the same cloud platform, the encrypted traffic generated by smart home devices will be different. At present, many scholars have studied the identification of devices on different cloud platforms and made great progress, which we will introduce in related work. Smart home devices will send heartbeat packets to the cloud platform to keep connected at a certain interval, packet number, and length. When we use the

mobile App to control the device, the encrypted traffic of different control commands of some smart home devices is different in terms of traffic rate, packet size, and so on. Based on these differences, we design event signatures to distinguish the control commands.

3.3.1. Packet Sequence. IoTEvent extracts the timestamp from the packet p of the device to obtain the time interval Δt between two packets. When the network is under the good condition, the data sending and receiving rate will be accelerated in a triggered event, so the time interval Δt is smaller. Therefore, IoTEvent judges the number of event according to Δt . If $\Delta t > a$, the packet is marked. Otherwise continue. Based on the recorded data, IoTEvent divides device traffic into packet sequences and then filters out the packet sequences generated by the trigger event.

3.3.2. Event Pairing. Event pairing is to match the event name with the packet sequence. Each packet sequence represents a triggered event, $E_i^n = p_1 p_2 p_3 \cdots p_n$, n is the number of packets, and i is the event name, which is an unknown message. IoTEvent compares the timestamp of the first packet in the packet sequences and the triggered event. If the time interval Δt is less than b , the event name and packet sequence are matched. In smart home devices, the events between device and the cloud are limited. Simple devices (such as smart sockets and smart light bulbs) have relatively few events, while complex devices (such as smart gateways) have more events.

IoTEvent extracts information from packet p , and each packet corresponds to a five-tuple, $p_n = (t_n, \Delta t_n, IP_d, IP_s, l)$, t_n is timestamp of the packet, Δt_n is the time interval between this packet and the previous packet, IP_d is the destination IP address, IP_s is the source IP address, and l is the length of packet. We define $d = 1$ for the data package from the cloud to the device, and $d = -1$ for the data package from the device to the cloud. The event can be represented as a $1 \times n$ vector, $E_i^n = [d_1 \times l_1, d_2 \times l_2 \cdots d_n \times l_n]$ i is the name of event, which is also the label of the data set, n is the packet number of event, and the n of the same event may be different due to the existence of disordered data packets (e.g., device log and device network information).

3.3.3. Signature File. The signature file is a classification model for event detection. IoTEvent groups events E_i^n according to the number of packets n , as shown in formula (1). In network traffic, query and reply packets exist in pairs, so the number of packets n of events is even. Generally, the query and reply packet pair of the trigger event is less than 15, so $\max(n) \approx 30$ in

$$\sum_{n=2}^{\max(n)} E_i^n = \{E_i^2, \dots, E_i^{\max(n)}\}, n = 2a (a \in N^+). \quad (1)$$

After the events are grouped, each group is a data set. IoTEvent uses the kNN (K-nearest neighbor) algorithm [19] to train a classification model, which is a supervised learning

method for modeling or predicting discrete random variables. The goal is to learn a classification function or model from the training sample data set with known labels, which is also known as a classifier. When getting new data, the new data item can be predicted based on the classifier, and the new data item can be mapped to a class in a given category. In terms of classification, the input training data contain the following information: feature, attribute, label, or class, which can be used to represent $(F1, F2, \dots, FN; \text{label})$. The essence of research is to find out the relationship between features and markers (i.e., mapping). The hierarchical prediction model is to map input variables (attributes) and discrete output variables (categories). In this way, if the unknown data have no label, the unknown data can be predicted by the mapping function. kNN is a case-based classification algorithm. By calculating the distance between the different eigenvalues of the test object and the sample, the label of K adjacent samples is selected as the result.

IoTEvent uses 80% of the generated sample set as the training set to train the model, and the remaining 20% data set is used as the test set. The effect of the detection model is directly judged through the test data. In addition, use the test set to improve the model before it is used to detect. According to the result of event pairing, the product of the length and direction of each packet is taken as a value of the feature vector, the packet number of event is the length of the vector, and the event name is the label. IoTEvent generates the training set S and test set T , E_i is an event in the training set, and E_j is an event in the test set. The distance between two events is calculated in

$$d_{ij} = |E_i^n - E_j^n| = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}. \quad (2)$$

IoTEvent selects K events in S with the smaller distance d_{ij} as the nearer neighbors of E_j . IoTEvent returns the event name of the most frequent occurrence of the K events as the result. IoTEvent compares test set labels and results to calculate the detection accuracy. Through training, the classification model is optimized to improve the accuracy of event detection. The value of K is related to the number of samples in the training set. IoTEvent chooses the best classification accuracy by comparing different K values. After training, all samples are recorded in the signature file as the classification model of detection.

3.4. Fingerprint Event Detection. The process of event detection is to identify device events in the sniffed traffic. We analyze the process of traffic capture from the perspective of the adversary. IoTEvent divides traffic into packet sequences, then recognizes events based on the signature file, and finally generates event output.

3.4.1. Sniffing Traffic. The adversary has two ways to sniff traffic: WiFi sniffing and WAN sniffing. Through WiFi sniffing, the adversary can capture the data link layer traffic and identify the device through the MAC address. The data

transmitted by WiFi are encrypted. The adversary should filter irrelevant data packets (e.g., 802.11 CTS, 802.11 Frag) and only retain 802.11 encrypted data. Through the WAN sniffing, the adversary can capture the network layer traffic and identify the source IP address and destination IP address, one of which is the router IP and the other is the cloud IP. The adversary can find the domain name of the IP address through IP reverse DNS. Based on the domain name, the adversary can identify whether the packet is between the device and the cloud.

3.4.2. Event Detection. Event detection is to identify the event of the packet sequence and determine whether the event is a privacy event. The private event refers to the event that we can obtain the privacy of the user's behavior, so privacy events include device physical state change information and device operation command. For example, the motion sensor reports movement.

In order to identify privacy events, we define the name of all events set on a certain device as ϕ , the name of private events set as α , and the name of nonprivate events set as β , so $\phi = \alpha + \beta$. We can define all the events E_ϕ as shown in formula (3), among which E_α^k represents the privacy event with k packets.

$$E_\phi = \sum_{k=1}^{\max(k)} E_\alpha^k + E_\beta^k. \quad (3)$$

Event detection is a concrete realization of privacy event identification for encrypted traffic based on event signature. IoTEvent uses the trained kNN model to achieve event detection. The names of all privacy events are put into a set α , and when the privacy event name is detected, an alarm will be raised. The implementation of event detection is shown as follows.

- (1) IoTEvent extracts the information from the packet, $p_n = (t_n, \Delta t_n, IP_d, IP_s, l)$. Each packet sequence is an event to be detected. The packet sequence can be represented by a $1 \times n$ vector, $E_j^n = [d_1 \times l_1, d_2 \times l_2 \dots d_n \times l_n]$, and the j is event name, which need to be identified.
- (2) Measure the distance. According to the number n of event E_j^n , IoTEvent screens out the events $\sum E_i^n$ with the number n in the signature file and then calculates the distance d_i between the events E_j^n and the events E_i^n in signature file, $d_{ij} = |E_i^n - E_j^n|$.
- (3) Select K events with the smaller distance d_{ij} as the nearer neighbors of E_j^n . IoTEvent returns the event name of the most points in the K neighborhood as the result. If it belongs to privacy events set α , an alert is issued; otherwise no alert is issued.

4. Evaluation

In this section, we evaluated the effectiveness of IoTEvent. Specifically, we conducted experiments on nine smart home devices which use five different cloud platforms. We

TABLE 1: List of smart home devices.

The cloud	Vendor	Device name	Device model
Xiaomi	Yeelight	Yeelight light strip	YLDD04YL
Xiaomi	Chuangmi	Mi plug mini	ZNCZ02CM
Xiaomi	Lumi	Mi control hub	DGNWG02LM
Tuya	Tuya	WiFi lamp	2AJ3WABEQPZ05
Tuya	Hongshi	WiFi plug	F2s501-GB
TP-Link	TP-Link	Smart WiFi plug	HS100
Hicloud	ORVIBO	Smart socket	S30c
Hicloud	Jellyfish Tur	LED light bulb	BRO-16565
JD	BroadLink	Smart plug	SP mini 3

described how the experiment was set up in and then presented the experiment result.

4.1. Experiment Setup. Before the experiment, we need to complete the construction of the experimental environment. We used mobile phones to control devices, which model is SEN-AL00 and the system is Android 10.0. The encrypted traffic capturing and sniffing were conducted by the sniffer we built. In particular, the traffic capture used the wireless network card, of which the interface is USB 3.0, the speed is 300–866 Mbps, and the wireless standard is IEEE 802.11ac/a/b/g/n. The traffic detection was conducted on a Windows 10 equipped with a Intel Core i7-10710U CPU.

Table 1 provides the smart home devices of the experiments. The first column shows the cloud platforms used by the devices. The second column presents the device vendor. The third column is the device name. The fourth is the device model. IoTEvent runs with a local Python 3.6 programming environment in Windows 10, which achieves traffic capture, training, and event detection of the devices.

4.2. Experiment Results

4.2.1. Data Collection. During the experiment, the protocol of devices is given in Table 2, which includes TLS, TCP, UDP, and MQTT. In addition, we recorded the domain name in Table 2 used by the cloud to send control commands. We found that the state of smart home devices is limited. When the smart home device connects to the cloud platform for the first time, it should report the device hardware and network information, which we call device connection events. In normal operation, the limited state of smart home devices is converted to each other. In general, the events of smart gateway mainly include heartbeat package, device log, and subdevice information, while the gateway with additional functions has relatively more events. Moreover, the number of smart gateway events also depends on the number of connected subdevices. The subdevice reports the heartbeat package and the status change information through the smart gateway. Therefore, the more subdevices the smart gateway connects to, the more events there are.

4.2.2. Training. The manual analysis indicates that the packet sequences of different events are different. In addition

TABLE 2: Summary of experimental results.

The cloud	Device name	Protocol	The domain name of command	Trigger event	WiFi sniffing Accuracy (%)	WAN sniffing Accuracy (%)
	Yeelight light strip plus	TLS	https://ots.io.mi.com	Turn on	100	100
				Turn off		
	Mi plug mini	TLS	https://ots.io.mi.com	Color	100	100
				Color setting		
Xiaomi	Mi control hub	UDP	https://ots.io.mi.com	Flow	98.54	98.88
				Flow setting		
	WiFi lamp	MQTT	https://mq.gw.tuyancn.com	Schedules	100	100
				Timer		
Tuya	WiFi plug	TLS	https://m2.tuyancn.com	Favorites	100	100
				Power on/off		
Tp-Link	Smart WiFi plug	TLS	https://use1-api.tplinkra.com	Set time on/off	98.75	98.82
				Schedules		
Hicloud	Smart socket	TCP	https://iomplatform.hicloud.com	Away mode on/off	99.60	99.54
				Alert enabled/disabled		
	LED light bulb	TCP	https://iomplatform.hicloud.com	Light turn on/turn off	100	100
				Colored lamp color		
JD	Smart plug	TCP	https://live.smart.jd.com	Colored lamp brightness	100	100
				Scene color of colored lamp		
	WiFi plug	TLS	https://m2.tuyancn.com	Timed alert	100	100
				Alert trigger device		
	WiFi plug	TLS	https://m2.tuyancn.com	Timer colored lamp	100	100
				Add child device		
	WiFi plug	TLS	https://m2.tuyancn.com	Snooze alarm clock	100	100
				Doorbell trigger device		
	WiFi plug	TLS	https://m2.tuyancn.com	Delay effective time	100	100
				Volume settings		
	WiFi plug	TLS	https://m2.tuyancn.com	Language of voice prompt	100	100
				Network radio		
	WiFi plug	TLS	https://m2.tuyancn.com	Hub alert ringtone	100	100
				Alert volume		
	WiFi plug	TLS	https://m2.tuyancn.com	Alert red light blink time	100	100
				Alert time		
	WiFi plug	TLS	https://m2.tuyancn.com	Linkage alert	100	100
				Light turn on/turn off		
	WiFi plug	TLS	https://m2.tuyancn.com	Turn on/off	100	100
				Turn on/off		
	WiFi plug	TLS	https://m2.tuyancn.com	Schedules	100	100
				Set time on/off		
	WiFi plug	TLS	https://m2.tuyancn.com	Electricity consumption	100	100
				Light turn on/turn off		
	WiFi plug	TLS	https://m2.tuyancn.com	Turn on/off	100	100
				Scheduling on/off		
	WiFi plug	TLS	https://m2.tuyancn.com	Set time on/off	100	100
				Away mode on/off		
	WiFi plug	TLS	https://m2.tuyancn.com	Turn on/off	100	100
				Timer on/off		
	WiFi plug	TLS	https://m2.tuyancn.com	Delay on/off	100	100
				Turn on/off		
	WiFi plug	TLS	https://m2.tuyancn.com	Fast on	100	100
				Slow on		
	WiFi plug	TLS	https://m2.tuyancn.com	Timer	100	100
				Delay		
	WiFi plug	TLS	https://m2.tuyancn.com	Turn on/off	100	100
				Set time on/off		

to network traffic of triggerable events, the device also generates other network traffic (e.g., device heartbeat, device logs).

For example, Mi control hub is a smart home device that connects to Mi Home. We use it as an illustrated example

and manually analyze its network traffic. We get all events that can be triggered from Mi Home App, as given in Table 3, which relate to user privacy marked in gray.

We connected the mobile phone and the device to the LAN. Then, we simulated the user clicking the button and

TABLE 3: All events of Mi Hub.

The smart home devices	The events
Mi control hub	Alert enabled/disabled
	Light turn on/turn off
	Colored lamp color
	Colored lamp brightness
	Scene color of colored lamp
	Timed alert
	Alert trigger device
	Timer colored lamp
	Add child device
	Snooze alarm clock
	Doorbell trigger device
	Delay effective time
	Volume settings
	Language of voice prompt
	Network radio
	Hub alert ringtone
	Alert volume
Alert red light blink time	
Alert time	
Linkage alert	

recorded the timestamp of the click. At the same time, we used tcpdump to capture network traffic of the device. Through manual analysis of network traffic, we found the packet sequences are different for each event, and the packet sequences of three different events are shown in Figure 4. For the events light turn on/turn off, the packet sequences are the same. We observed an exchange of UDP application data packets between the Mi Home Hub and Internet host where the packet lengths were (138, 154), (106, 106). However, for the event alert enabled, the device sends UDP packets of lengths (154, 186) to an Internet host and receives reply packets of lengths (106, 106). Similarly for the event of alert disabled, these packets were of lengths (170, 138, 170), (106, 106, 106).

All trigger events of different devices are given in Table 2. Except for trigger events, other device events (e.g., heartbeat packet and device log) are signed as other events. Not all trigger events have traffic between the device and the cloud platform, and part events are recorded in the cloud and are triggered when the set conditions are reached. IoTEvent only recorded trigger events that generate traffic. In our experiment, the kNN algorithm is used. The average precision for the different k is shown in Figure 5.

4.2.3. Event Detection. Among the devices we tested were smart gateways and cloud-connected devices. Different types of devices contain different types of privacy information, as given in Table 4. The privacy information contained in the smart gateway that we can detect by encrypting the traffic includes the connected subdevices, the subdevice sensor status, the operation commands, and the state change information. The privacy information contained in the cloud-connected devices that we can detect by encrypting the traffic includes the statue change information and the operational commands.

We simulated using the device, and each event was triggered ten times. We captured the traffic using WiFi

sniffing and WAN sniffing, and then IoTEvent was used to detect. The definition of the accuracy is shown in formula (4). TP indicates the number of trigger events that are correctly detected, FN indicates the number of trigger events that are erroneously detected, TN indicates the number of nontrigger events that are correctly detected, and FP indicates the number of nontrigger events that are erroneously detected.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

The result of event detection is given in Table 2. Most event signatures of the devices are unique, so the event would be detected. Through the detection results of events, we can infer the privacy of users' behaviors. For example, if the light is turned off at night, we can judge that users will go to sleep. When the mobile sensor detects someone moving, it can judge someone's activity in the house. Such privacy can be obtained only by monitoring network traffic, which greatly increases the risk of people's privacy leakage.

4.3. Comparison with the Existing Work. We selected two representative works for comparison: the [7] and the [15]. The comparison is mainly carried out from the following three aspects: the range of detection, the accuracy, and the computational overhead (Table 5).

IoTEvent is not designed for specific types of protocols. In our experiments, we proved that IoTEvent is suitable for TCP, UDP, TLS, and MQTT protocol. In the [15], their tool is only suitable for TCP protocol. In the [7], the author identified the event based on the traffic rate, thus which applied any protocol. IoTEvent can accurately detect the name of the event, and the average accuracy can reach 99%. In contrast, the [7] cannot accurately detect the occurrence of the event. The [15] cannot determine the event is unique, because it only clusters for a particular event. The event detection accuracy of [15] is 97%. Compared to [15], IoTEvent required more computation time and memory, but which is within a reasonable range. The [7] required more storage overhead and do not have computational overhead.

5. Discussion

According to the results of event detection, we analyzed the reason for privacy leakage, proposed several suggestions, and discussed the limitations.

5.1. The Reasons. Different cloud platforms have their own communication protocols, Huawei, Tuya, Xiaomi, and JD use custom protocols, so the protocol name is also named by themselves. Before the device development, the device developer should first select a cloud platform and develop the device according to the SDK provided by the cloud platform. Currently, the smart home cloud platform provides a variety of communication protocols in the SDK, including platform custom protocols, MQTT, Coap, Soap, Http, and Https. The experiment found that the device using the platform's custom protocol had a high probability of privacy leakage of

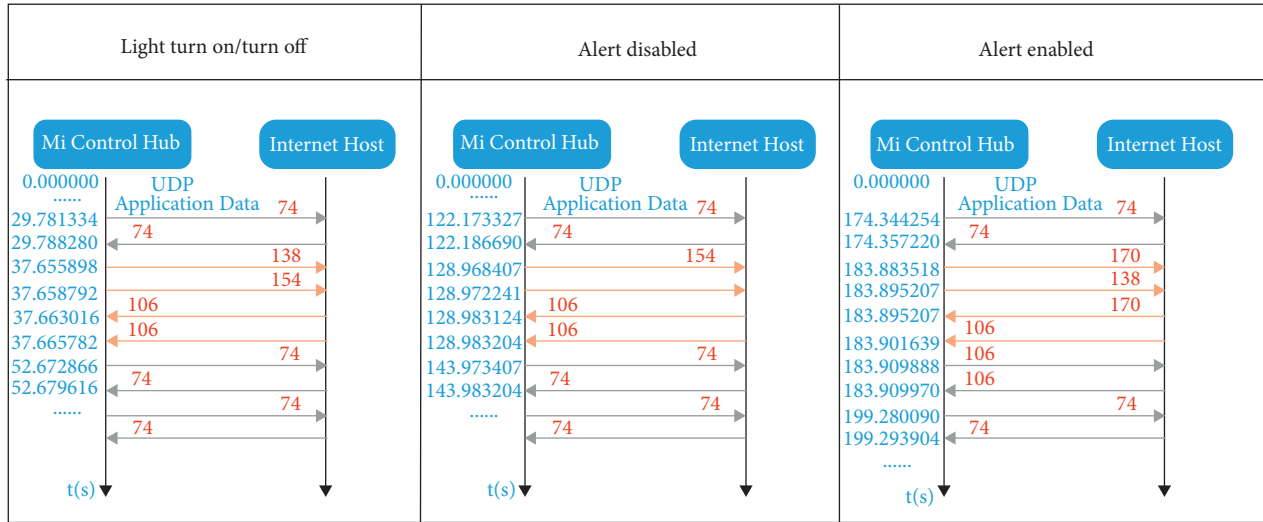


FIGURE 4: The packet sequences of three different events for Mi control hub: light turn on/turn off, alert enabled, alert disabled, the line of which is marked yellow. The UDP protocol is used to transmit application data in device and cloud communication. The arrow represents packet direction, and the number represents packet length.

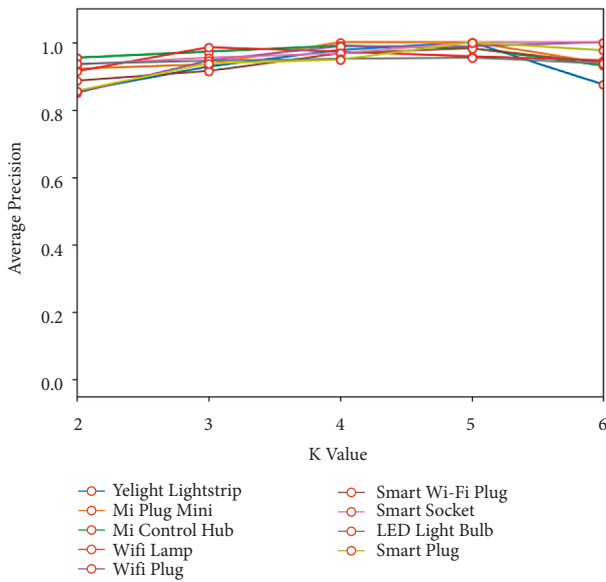


FIGURE 5: Average precision for the different k values.

TABLE 4: Type of device privacy.

Device type	Privacy information
The smart gateway	The connected subdevices
	The subdevice sensor status
	The operation commands
	The state change information
The cloud-connected devices	The operation commands
	The state change information

encrypted traffic. Taking the custom protocol of Xiaomi platform as an example, the Xiaomi cloud platform adopts AES-128-CBC [20] encryption, which requires (key, iv), which are both 16 bytes, and adopts UDP for data

transmission. The plaintext must be a multiple of 16, and the ciphertext must be the same length as the completed plaintext. Any change of more than 16 bytes in the transmitted plaintext data will change the length of the transmitted encrypted packet. Moreover, the number of packets is different for different events of the platform custom protocol.

5.2. The Suggestion

5.2.1. *Fill in a Random Information.* For the custom protocol of smart home cloud platform, random information is added in the design process of the protocol so that the size of the packet is independent of the content. Adding a random information independent of the command to the command data sent can make the information size of the communication packet to change randomly, but it does not change the packet rate.

5.2.2. *Change the Encryption Method.* For the smart home cloud platform with privacy disclosure of encrypted traffic, the purpose of traffic shaping can be achieved by using encryption. If the number and size of all event packets are unified through encryption, privacy events cannot be detected, so as to alleviate the privacy disclosure of encrypted traffic.

5.2.3. *Use VPN.* VPN (virtual private network) connects the two LANs together and encrypts the transmission function to make the network more secure and confidential. VPN can also change the user’s IP address, making it difficult for the device to be tracked. At the same time, it can also change the rate of communication packets, making it difficult to detect the privacy leakage of encrypted traffic.

TABLE 5: Comparison with existing works.

Comparison	Existing works		
	Aphorpe's work	Trimananda's work	Our work
The range	No restrictions	Only TCP	No restrictions
The accuracy	—	97%	99%
The computational overhead	High	Reasonable range	Reasonable range

6. Related Work

We review the related work on smart home security from two perspectives: the identification of smart home devices and the privacy leakage of smart home devices.

6.1. Events Identified for IoT Traffic. In recent years, there has been increasing attention paid to the privacy leakage problem of encrypted traffic. In [15], the author presents a tool that can automatically extract packet-level signatures for events from network traffic of TCP-based devices. In [11], an attack tool that infers the house status by inspecting the bit rate variation of the wireless camera traffic was proposed. In [9], the authors design an accurate and efficient smart spying strategy, which can infer a user's activity (such as web browsing, e-mail, and chat) from encrypted wireless traffic. In [10], the authors infer the state transition of smart home devices through Zigbee and Z-Wave encrypted traffic to detect eavesdrop or spoof events of smart home Apps. In [7], the authors examine four smart home devices and find that their network traffic rates can reveal potentially sensitive user interactions even when the traffic is encrypted. In [8], the authors check the action and state of smart home devices through network traffic time characteristics. In [12], the authors showcase risks of machine learning techniques to develop black-box models to automatically classify traffic and implement privacy leaking attacks. In [21], the authors investigate the privacy leakage of encrypted traffic from smart speakers.

6.2. Devices Identified for IoT Traffic. Before our work, we need to identify smart home devices through encrypted traffic. At present, many researchers have studied IoT device identification. The authors of [22] propose an acquisitional rule-based engine (ARE) that can automatically generate rules for discovery and annotation of IoT devices without any training data. The authors of [23] used the network traffic characteristics of IoT devices to train the machine learning model to detect the types of IoT devices. The authors of [24] used the small deviation in the hardware device to realize the device fingerprint so as to achieve the purpose of device identification. The authors of [25] proposed to detect chip-sets, firmware, or drivers by observing the response (or lacking of response) of 802.11 wireless devices to a series of nonstandard 802.11 frames. The authors of [26] proposed two device-type fingerprint identification methods to enhance the existing intrusion detection methods in the integrated circuit environment. The first method measures data response processing time

and exploits the static and low-latency nature of the private industrial control system network to develop accurate fingerprints, while the second method uses physical operation time to develop unique signatures for each device type. These methods can accurately identify Internet of Things devices.

7. Conclusion

Nowadays, smart home devices have become an indispensable part of our life, yet deficiency in privacy protection will be an obstacle to its development [27]. In this article, we have proposed a tool to detect the privacy leakage of smart home devices from encrypted traffic. First, we present the threat model. Then, the tool for trigger event detection is designed. Finally, IoTEvent is evaluated with nine smart home devices on five cloud platforms. We are able to detect specific behaviors and actions from encrypted traffic. More specifically, we analyze the reasons for the privacy leakage of these devices on the cloud platform and put forward suggestions to alleviate this problem. Finally, we briefly introduce the related work. In our future work, we would like to extend our tool to more devices on different cloud platforms to make it applicable for a variety of environment.

Data Availability

The data used to support this study are available at <https://github.com/yangting111/IoTEvents/tree/main>.

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 (U1836210) and the Key Research and Development Science and Technology of Hainan Province (ZDYF202012).

References

- [1] Abiresearch, "Digital Transformation Doesn't Have To Be Disruptive," 2020, <https://www.abiresearch.com/>.
- [2] XiaoMi, "XiaoMi," 2020, <https://iot.mi.com/new/index.html>.
- [3] Huawei, "Huawei," 2020, <http://iot.hilink.huawei.com/>.
- [4] P. Morgner, C. Mai, N. Koschate-Fischer, F. Freiling, and Z. Benenson, "Security update labels: establishing economic incentives for security patching of iot consumer products," 2019, <https://arxiv.org/abs/1906.11094>.

- [5] S. Zheng, N. Apthorpe, M. Chetty, and N. Feamster, "User perceptions of smart home iot privacy," *Proceedings of the ACM on Human-Computer Interaction*, CSCW, vol. 2, pp. 1–20, 2018.
- [6] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.
- [7] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: privacy vulnerabilities of encrypted iot traffic," 2017, <https://arxiv.org/abs/1705.06805>.
- [8] A. Acar, H. Fereidooni, T. Abera et al., "Peek-a-boo: I see your smart home activities, even encrypted!," 2018, <https://arxiv.org/abs/1808.02741>.
- [9] T. Hou, T. Wang, Z. Lu, and Y. Liu, "Smart spying via deep learning: inferring your activities from encrypted wireless traffic," in *Proceedings of the IEEE GlobalSIP*, Ottawa, ON, Canada, November 2019.
- [10] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "Homonit: monitoring smart home apps from encrypted traffic," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1074–1088, Toronto Canada, October 2018.
- [11] Y. Cheng, X. Ji, X. Zhou, and W. Xu, "Homespy: Inferring User Presence via Encrypted Traffic of home Surveillance Camera," in *Proceedings of the ICPADS*, pp. 779–782, Shenzhen, China, December 2017.
- [12] D. Caputo, L. Verderame, A. Merlo, A. Ranieri, and L. Cavaglione, "Are You (Google) home? Detecting Users' Presence through Traffic Analysis of Smart Speakers," *Computer Science*.
- [13] S. Farahani, "ZigBee Wireless Networks and Transceivers," Newnes, Elsevier, Amsterdam, Netherlands, 2011.
- [14] M. B. Yassein, W. Mardini, and A. Khalil, "Smart homes automation using z-wave protocol," in *Proceedings of the 2016 International Conference on Engineering & MIS (ICEMIS)*, pp. 1–6, IEEE, Agadir, Morocco, September 2016.
- [15] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Packet-level signatures for smart home devices," *Signature*, vol. 10, no. 13, p. 54, 2020.
- [16] K. Egevang and P. Francis, "The Ip Network Address Translator (Nat)," RFC 1631, May, Tech. Rep, 1994, <https://www.rfc-editor.org/rfc/rfc1631>.
- [17] Auto js, "Auto.js," 2020, <http://www.autojs.org/>.
- [18] H. Liu, J. Li, and D. Gu, "Understanding the security of app-in-the-middle iot," *Computers & Security*, vol. 97, 2020.
- [19] P. Soucy and G. W. Mineau, "A simple knn algorithm for text categorization," in *Proceedings of the 2001 IEEE International Conference on Data Mining*, pp. 647–648, IEEE, San Jose, CA, USA, November 2001.
- [20] J. Daemen and R. V. Reijndael, "The advanced encryption standard," *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 26, no. 3, pp. 137–139, 2001.
- [21] C. Wang, S. Kennedy, H. Li et al., "Fingerprinting Encrypted Voice Traffic on Smart Speakers with Deep Learning," 2020, <https://arxiv.org/abs/2005.09800>.
- [22] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering internet-of-things devices," in *Proceedings of the 27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 327–341, Baltimore, MD, USA, August 2018.
- [23] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of iot devices," in *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security*, pp. 41–50, Toronto Canada, October 2018.
- [24] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.
- [25] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles, "Active behavioral fingerprinting of wireless devices," in *Proceedings of the first ACM conference on Wireless network security*, pp. 56–61, Alexandria VA USA, April 2008.
- [26] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. A. Beyah, "Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems," in *Proceedings of the iNDSS*, San Diego, CA, USA, February 2016.
- [27] A. Yang, C. Zhang, Y. Chen, Y. Zhuansun, and H. Liu, "Security and privacy of smart home systems based on the internet of things and stereo matching algorithms," *IEEE Internet of Things Journal*, vol. 7, 2019.

Research Article

A Traceable and Anonymous Data Aggregation Scheme with Fog Computing Devices for Smart Grid

Fan Wu ¹ and Xiong Li ²

¹School of Management, Xiamen University Tan Kah Kee College, Zhangzhou 363105, China

²Institute for Cyber Security,

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

Correspondence should be addressed to Xiong Li; lixiongzhq@163.com

Received 22 February 2022; Revised 3 April 2022; Accepted 8 April 2022; Published 22 June 2022

Academic Editor: AnMin Fu

Copyright © 2022 Fan Wu and Xiong Li. 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 extension of the Internet of Things (IoT), smart city is a new aim for applications used in different industrial aspects. Intelligent IoT devices are used everywhere in smart cities to implement the functions such as monitoring and managing. Smart grid is one of the critical parts. Obtaining the quantity and cost of electricity usage is the most critical task of the smart grid. But such data in plaintext may leak the user privacy. So, it is an emergent aim to protect the private information of the user. Thus, we present a secure scheme for the smart grid, which not only protects the user's information including identity and power consumption in the communications but also tracks the correct electricity cost for each user. Also, electricity consumption data from users could be aggregated in fog devices and then analyzed by the utility provider. The formal proof shows that the message transmission reaches the security level of the chosen plaintext attack (CPA). Also, the security properties of the scheme express the robustness. Finally, the performance study demonstrates that the proposed protocol is acceptable in practice.

1. Introduction

Nowadays, smart city aims to optimize every function of the city and improve the quality of citizens' life with data analysis and technology development. Quality of life has been improved substantially in smart city. To complete the whole process from data collection to instruction assignment to concrete sensors, intelligent Internet of Things (IoT) devices such as smart meters, robots, aggregation devices, and software-defined production processes are deployed for different sorts of usage. Through sensors distributed everywhere, many kinds of data are gathered and sent to center servers for decision-making. Under that background, power grid, traffic, agriculture, and accommodation turn to be smart grid, smart traffic, smart agriculture, and smart home, respectively. However, security gaps containing weak authentication or vulnerability in code pieces lead to serious and urgent risks. Attackers can exploit the operating system and software's holes, eavesdrop on the messages in public channels between sensors and servers, or compromise

sensors to get valuable information after analyzing relative data flows.

Traditional electricity grid consists of power stations, high-voltage transmission lines, and distribution lines. They build a large network that delivers electricity from the producer to the users, and the balance of power supply and usage relies on the construction of power plants. The more electricity is required, the more plants should be built. But with the technology progressing, smart grid becomes an attractive term, and it turns to be true in some developed countries. Promising changes appear in every aspect, including intelligent power generation, transmission, and applications. Advanced metering infrastructure (AMI) is the architecture of smart grid, where bidirection demand response is an important property, which permits the users and the utility provider (UP) to monitor, adjust, count, and forecast the electricity use. The time-of-use pricing mechanism is employed, and users should pay higher fees in the peak time under this measure. Also, smart grid is one of the most important sorts of cyber-physical systems (CPSs). As

physical devices, smart meters (SMs) are distributed widely outside the houses or gathered in a meter box in a building. Each smart meter collects the power consumption value in one house with every fixed period to calculate the fees due to different prices in peaks and valleys. Since the power usage data are fine-grained, smart grid faces security problems containing threats and weaknesses including identity and consumption leaking, replay attack, denial-of-service attack, and so forth. The attacker may deduce user habits or behaviors via such information, so the privacy of the personal information turns to be an important issue and is discussed by researchers. According to [1], several requirements should be satisfied, including transmitted data privacy, data reliability, and authentication between participants.

With the popularization of smart grid, enormous data have been generated. Simple data processing is done on fog devices (FDs). Fog computing means that some critical computations are completed on the edge of the network or fog devices distributed everywhere. SMs in one domain submit their collected data to the corresponding FD, and the extensional calculations, like clarifying the consumption fee and making data aggregations, could be done based on the data owned by FDs. All transmitted messages should be kept away from danger. Authentication and public key mechanism [2–4] are the necessary ways to protect the security of data. It is necessary to study the current situation of privacy-preserving schemes for smart grid and we will list the related literature.

1.1. Related Work. In recent years, a host of schemes for AMI has been presented. Based on [5], there are three classic sorts: key agreement, only consumption data encryption, and data aggregation. Authentication and key agreement is the usual way for sending information in smart grid [2–4, 6–15]. In 2011, Fouda et al. [6] presented a key agreement scheme between the building area network and home area network. But, it is criticized in [7] that heavy computation cost is used. Then, Li et al. [11] pointed out that identities of users were exposed by plaintext in [7]. In 2016, Tsai and Lo [8] presented a scheme with a session key formed between the smart meters and corresponding service supporter. But Odelu et al. and He et al. [9, 10] considered that calculation time in [8] costs too much due to bilinear pairing calculations. Unfortunately, the two schemes could not satisfy the anonymity of user [2]. In 2018, schemes [12–14] were proposed, but the weaknesses like lack of forward security and anonymity were still troubling people.

The second sort is that only the consumption data are protected, but the user's identity is not considered as the secret [5, 16–20]. In 2015, Diao et al. [16] proposed a scheme built on zero-knowledge and Camenisch-Lysyanskaya signature. They claimed that user identity was anonymous and linkable in the scheme, but soon, the forgery attack on the scheme was given in [17]. In 2016, Sui et al. [18] designed a scheme between SM and the electricity utility. But the user who consumes more electricity than the threshold will be exposed in plaintext. Next year, Ge et al. [19] pointed out that unlinkability feature could not be satisfied in [18]. But

the two schemes [18, 19] are unfit due to exposing user identity simply and crudely. And in [5], secure channel is required several times when normal data transmission proceeds, and the identity is also in plaintext against the anonymity requirement. In 2020, Ding et al. [20] put forward a data aggregation scheme for smart grid, but both the identity and public key of user were transmitted directly in the public channel. In 2021, Su et al. [21] proposed a changeable threshold-based aggregation scheme for smart grid. However, identity is ignored in the message transmission, and only the aggregation value can be obtained in the control center, which is equal to UP here. Also, Wang et al. [22] presented an aggregation scheme keeping privacy of user in the same year, but the identity of user was still ignored.

The last sort is aggregating data relative to the consumption [1, 23–29]. In 2017, He et al. proposed studies [23, 24] which described the aggregation between SMs and the special aggregator. But in the aggregation part, the identity of user, which should be transmitted in plaintext, is needed to verify the data. Wang's scheme [25], which employed the identity-based signature, had the same problem, where the user's identity must be exposed in the public channel or the final check on the aggregation device could not be completed. In 2017, Badra and Zeadally [1] presented a scheme with symmetric homomorphic encryption and Diffie-Hellman problem. Every time, one user should update the shared key between the server and himself, with the help of another user. But how to find the suitable helper is not mentioned. Shen et al. [26] proposed a cube-data aggregation scheme for smart grid. However, the user identity is also in plaintext. Lu et al. [27] proposed a Paillier encryption-based scheme to make the data aggregation. But the time-based hash chain in the scheme is not suitable if the smart meter malfunctions once. The fog device cannot check the correct submission, while the last one or several submissions are lost or rejected. Liu et al. [28] used lifted EC-ElGamal cryptosystem with plaintext identity. However, except [1], all of the above schemes do not consider the fee of each user. Only aggregation and some statistical operations, such as mean and variance values, are regarded. Smart grid is first for electrical consumption, and the fee of electricity usage is much more important than statistical data for prognosis. On this aspect, in 2016, Wang et al. [29] proposed a scheme which could not only disclose the user consumption but also collect statistical data for computing statistic values. However, some weaknesses are exposed: the data in the aggregation process cannot be verified, the adversary can calculate the private consumption from the message, and the pseudoidentity will be exhaustively searched on the trusted server side by doing both hash and scalar multiplication, since the trusted server only calculates the collected data of the special smart meter which is required by UP.

Usually, the quantity of electricity usage is set as the discrete logarithm in aggregation. Based on [30], the power energy consumption in China is about 7225.5 terawatt/hour in 2017. This number is at the level of 2^{48} , and such discrete logarithm could be solved in 0.1 s [31], generally with the

TABLE 1: Notations.

Symbol	Meaning
SM_i, ID_{SM_i}	i^{th} smart meter and its identity
FD_j, ID_{FD_j}	j^{th} fog device and its identity
TS	Trusted server
UP, ID_{UP}	Utility provider and its identity
G	A cyclic group on a finite field F_n from an elliptic curve
q	A large prime, which is the order of G
P	Generator of G
Z_q^*	Multiplicative group modulo q
$h_i (i = 1, 2, \dots, 10)$	Hash functions
\oplus	Exclusive or operation
\parallel	Concatenation operation
V	Distribution of power consumption in a domain
v_i	Quantity of usage on SM_i
l_s	Security length
$z_j/Z_j = z_jP$	Private/public key for FD_j
$x/X = xP$	Private/public key for TS
$y/Y = yP$	Private/public key for UP
key_j	The common secret key between FD_j and UP
\mathcal{A}	The adversary
M_1, M_2, M_3	Messages

Pollard rho algorithm [32]. This technology is employed in many studies [23–25, 27, 29, 33].

1.2. Contributions.

- (1) We give a new data transmission scheme for smart grid, and it could make both the power consumption and data aggregation clear.
- (2) Formal proof demonstrates that the messages are robust enough against forgery attacks
- (3) Considering the security characters and performance evaluation, our scheme is good for practicality

1.3. Organization of the Paper. The rest of study is organized as follows: Section 2 expresses the basic knowledge of the study. Our scheme is in Section 3. Then, the formal proof lies in Section 4 and security analysis is in Section 5. The performance situation is in Section 6. Finally, the conclusion is drawn in Section 7.

2. Preliminary

2.1. Notations. In Table 1, the notations used throughout the study are given.

2.2. Referred Mathematical Problems. The problems given in the following are based on the elliptic group G with order q and generator P mentioned in Table 1.

Definition 1. The discrete logarithm (DL) problem is that in the tuple (P, aP) , where $a \in Z_q^*$ is unknown, it is hard to calculate a .

Definition 2. The computational Diffie–Hellman (CDH) problem on G is that given the tuple (P, aP, bP) , where $a, b \in Z_q^*$ are unknown, it is hard to calculate $abP \in G$.

Definition 3. The decisional Diffie–Hellman (DDH) problem on G is that given the tuple (P, aP, bP, cP) , where $a, b \in Z_q^*$ are unknown, it is hard to judge if $cP = abP$.

Definition 4. The gap Diffie–Hellman (GDH) problem on G is that given tuple (P, aP, bP) , where $a, b \in Z_q^*$ are unknown, it is hard to calculate $abP \in G$ via the help of DDH tool, like an oracle. Here, we define $\varepsilon = \text{Adv}_{\mathcal{A}}^{\text{GDH}}(t)$, which is the probability of solving GDH problem for \mathcal{A} in polynomial time t .

2.3. Network Model. Our scheme relies on the model in [29]. If the users obey the laws and pay the bill in time, it is unnecessary to expose the fine-grain consumption value to smart grid, especially UP. UP only accepts the aggregation of consumption and changes power supply with the corresponding price in different time spans. The architecture of the network is shown in Figure 1. Four kinds of devices are in the network: smart meter, fog device, trusted server (TS), and utility provider. Smart meter measures the power consumption details for every house. It is natural that the fee should be checked. But in [29], the fee is calculated on SM side. Without verifications on some trusted server, such data cannot be believed. Different from [29], only the consumptions will be submitted. Fees are not considered. Also, only total electricity consumption in one house is submitted in our scheme. Consumptions from all appliances in the house are not considered. Such data are submitted to the fog device, which stores the collection and is ready to provide data for people to check the consumption and to aggregate the data without leaking user information. The trusted server can get power consumption from each user, in order to calculate the electricity fee later. Also, it generates the key for the fog device to send the aggregated data with encryption, in order to prevent the attacker from cracking. The utility provider requires the aggregated statistical data to predict the total data in a long time. Wired channels are between fog devices and the trusted server and between fog devices and UP.

2.4. Security Model. Combining studies [27, 29, 34], we give the security model of our scheme as follows.

- (1) The scheme is chosen plaintext attack (CPA) secure
- (2) The trusted server is reliable and can get the real identity and power consumption of user. It is for the property of user anonymity and traceability.
- (3) \mathcal{A} could eavesdrop and forge messages in the public channel
- (4) The UP and fog devices are honest-but-curious, since they execute the protocol but are curious about the privacy of users. However, according to [27], any FD_j will not collude with UP.

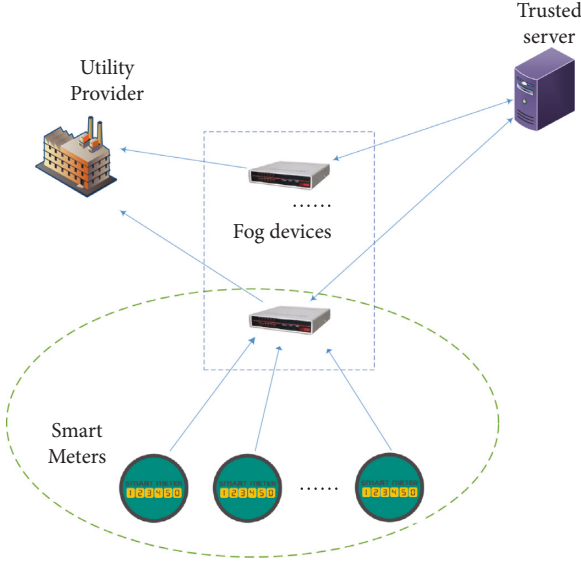


FIGURE 1: Network model.

3. Proposed Scheme

We divide our scheme into six phases: initialization, data encryption, consumption affirmation, aggregation key generation, aggregation, and aggregated data decryption. Different from [29], we do not consider user's fee submission, as the fee should be checked on the TS side, and it is not suitable to completely trust the fee calculated by smart meters. Moreover, we focus on the unitary consumption or one smart meter for a house. We do not use the way in [29], where each appliance is counted, respectively. Moreover, there is only one TS and one UP. Moreover, we put the process from Section 3.2 to Section 3.6 in Table 2.

3.1. Initialization. TS generates a cyclic group G with a large prime order q and generator P , as given in Table 1. Its private/public key pair is $(x/X = xP)$. Similarly, UP owns its private/public key pair $(y/Y = yP)$, and FD_j owns its private/public key pair $(z_j/Z_j = z_jP)$. Moreover, UP's identity is ID_{UP} and FD_j 's identity is ID_{FD_j} ; TS stores the pair (ID_{SM_i}, ID_{FD_j}) , where the submission target of ID_{SM_i} is ID_{FD_j} ; SM_i also stores the pair (ID_{SM_i}, ID_{FD_j}) , such that SM_i should submit the information to FD_j . At last, FD_j and UP have a common secret key key_j . UP stores the quantity of smart meters N_j in FD_j 's domain. Finally, hash functions are defined as follows: $h_i (i = 1, 2, 3, 5, 6, 7, 8, 9, 10): \{0, 1\}^* \mapsto \{0, 1\}^l$ and $h_4: \{0, 1\}^* \mapsto Z_q^*$.

3.2. Data Encryption. SM_i selects random numbers $r_1, r_2 \in Z_q^*$, picks the timestamp t_1 , and calculates the following elements with user consumption v_i : $A_{i,1} = r_1P$, $A_{i,2} = r_2P$, $A_{i,3} = h_1(r_1X) \oplus v_i$, $A_{i,4} = h_2(r_2X) \oplus ID_{SM_i}$, $A_{i,5} = 2r_1X + r_2X + v_iP$, $w_i = v_i^2$, $A_{i,6} = r_1X + 2r_2X + w_iP$, and $A_{i,7} = h_3(r_1X \| v_i \| ID_{SM_i} \| ID_{FD_j} \| w_i \| r_2X \| t_1)$. Then, SM_i sends $M_1 = \{A_{i,1}, A_{i,2}, A_{i,3}, A_{i,4}, A_{i,5}, A_{i,6}, A_{i,7}, t_1\}$ to the corresponding FD_j .

3.3. Consumption Affirmation. FD_j checks t_1 and stores M_1 if t_1 is valid. Then, it sends M_1 to TS, in order to disclose the user identity and corresponding power consumption. TS computes $B_1 = xA_{i,1}$, $B_2 = xA_{i,2}$, $v_i = A_{i,3} \oplus h_1(B_1)$, $ID_{SM_i} = A_{i,4} \oplus h_2(B_2)$, $B_3 = A_{i,5} - 2B_1 - B_2$, and $B_4 = A_{i,6} - B_1 - 2B_2$, finds out the corresponding FD_j according to SM_i , and checks if $B_3 = v_iP$, $w_i = v_i^2$, $B_4 = w_iP$, and $A_{i,7} = h_3(B_1 \| v_i \| ID_{SM_i} \| ID_{FD_j} \| w_i \| B_2 \| t_1)$. If true, TS could calculate the fee according to v_i .

3.4. Aggregation Key Generation. In order to help FD_j aggregate power usage in smart meters for a time span, TS selects a random string s , the timestamp t_2 in relative time span and $r_3 \in Z_q^*$, and calculates $B_5 = r_3P$, $B_6 = xh_4(r_3Y \| s) \oplus h_5(r_3Z_j \| ID_{FD_j})$, $B_7 = h_4^{-1}(r_3Y \| s) \oplus h_6(r_3Y \| ID_{UP})$, $B_8 = h_7(xh_4(r_3Y \| s) \| ID_{FD_j} \| B_7 \| t_2)$, and $B_9 = h_8(h_4^{-1}(r_3Y \| s) \| r_3Y)$. Then, TS sends $M_2 = \{B_5, B_6, B_7, B_8, B_9, t_2\}$ to FD_j .

3.5. Aggregation. FD_j checks t_2 , gets the number k as the number of functioning smart meters, picks up timestamp t_3 and $r_4 \in Z_q^*$, calculates $C_1 = B_6 \oplus h_5(z_j B_5 \| ID_{FD_j})$, and checks if $B_8 = h_7(C_1 \| ID_{FD_j} \| B_7 \| t_2)$. If true, FD_j computes $C_2 = \sum_{i=1}^k A_{i,1}$, $C_3 = C_1 C_2$, $C_4 = \sum_{i=1}^k A_{i,2}$, $C_5 = C_1 C_4$, $C_6 = B_5$, $C_7 = B_7$, $C_8 = \sum_{i=1}^k A_{i,5}$, $C_9 = \sum_{i=1}^k A_{i,6}$, $C_{10} = B_9$, $C_{11} = r_4P$, $C_{12} = h_9(r_4Y \| t_3) \oplus k$, and $C_{13} = h_{10}(C_3 \| C_5 \| C_7 \| C_8 \| C_9 \| C_{10} \| k \| key_j \| t_3)$. Then, FD_j sends $M_3 = \{C_3, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, t_3, ID_{FD_j}\}$ to UP.

3.6. Aggregated Data Decryption. UP checks t_3 and searches key_j based on ID_{FD_j} . If the checks and the data are found, it computes $k = C_{12} \oplus h_9(yC_{11} \| t_3)$ and checks $c_{13} = \{C_3, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, t_3, ID_{FD_j}\}$. If true, UP computes $D_1 = C_7 \oplus h_6(yC_6 \| ID_{UP})$ and checks if $C_{10} = h_8(D_1 \| yC_6)$. If correct, UP computes $D_2 = C_8 - 2D_1C_3 - D_1C_5$ and $D_3 = C_9 - D_1C_3 - 2D_1C_5$ and then uses the Pollard rho algorithm to get $W_1 = \sum_{i=1}^k v_i$ from D_2 and $W_2 = \sum_{i=1}^k w_i$ from D_3 . If $k = N_j$, the mean value $E(V) = W_1/k$ and the variance value $Var(V) = W_2/k - W_1^2/k^2$; else, if $k < N_j$, the variance value is changed to be $Var(V) = 1/(k-1)(W_2 - W_1^2/k)$.

4. Formal Proof

Nowadays, researchers consider that attackers should have negligible probability to retrieve any plaintext from ciphertext in cryptographic protocols. Such protocol should meet indistinguishability (IND) security which means that \mathcal{A} could not distinguish two plaintexts, while one of corresponding ciphertexts is given. In this study, chosen plaintext attack (CPA), which means \mathcal{A} does not have decryption right for any ciphertext he selects, will be proved for our scheme.

Our scheme is under IND-CPA secure. The concrete proof is given.

4.1. Basic Knowledge of IND-CPA Security for Our Scheme. Three games are brought in to explain the security for the three messages. We show the game process and then give the

TABLE 2: Data flow for consumption submission.

SM _i	FD _j	TS	UP
Select r_1, r_2, t_1 $A_{i,1} = r_1 P$ $A_{i,2} = r_2 P$ $A_{i,3} = h_1(r_1 X) \oplus v_i$ $A_{i,4} = h_2(r_2 X) \oplus ID_{SM_i}$ $A_{i,5} = 2r_1 X + r_2 X + v_i P$ $w_i = v_i^2 A_{i,6} = r_1 X + 2r_2 X + w_i P$ $A_{i,7} = h_3(r_1 X \ v_i \ ID_{SM_i} \ ID_{FD_j} \ w_i \ r_2 X \ t_1)$ $\rightarrow M_1 = (A_{i,1}, A_{i,2}, A_{i,3}, A_{i,4}, A_{i,5}, A_{i,6}, A_{i,7}, t_1)$	Check t_1 , stores M_1 $\rightarrow M_1$	$B_1 = xA_{i,1}, B_2 = xA_{i,2}$ $v_i = A_{i,3} \oplus h_1(B_1)$ $ID_{SM_i} = A_{i,4} \oplus h_2(B_2)$ $B_3 = A_{i,5} - 2B_1 - B_2, B_4 = A_{i,6} - B_1 - 2B_2$ Get FD_j Check $B_3? = v_i P, w_i? = v_i^2 P, B_4? = w_i P$ And $A_{i,7}? = h_3(B_1 \ v_i \ ID_{SM_i} \ ID_{FD_j} \ w_i \ B_2 \ t_1)$ Select s, t_2, r_3 $B_5 = r_3 P$ $B_6 = xh_4(r_3 Y \ s) \oplus h_5(r_3 Z_j \ ID_{FD_j})$ $B_7 = h_4^{-1}(r_3 Y \ s) \oplus h_6(r_3 Y \ ID_{UP})$ $B_8 = h_7(xh_4(r_3 Y \ s) \ ID_{FD_j} \ B_7 \ t_2)$ $B_9 = h_8(h_4^{-1}(r_3 Y \ s) \ r_3 Y)$ $M_2 = (B_3, B_4, B_5, B_6, B_7, B_8, B_9, t_2)$	Check t_3 , search key _j Compute $k = C_{12} \oplus h_9(yC_{11} \ t_3)$ Check $C_{13}? = h_{10}(C_3 \ C_5 \ C_7 \ C_8 \ C_9 \ C_{10} \ k \ key_j \ t_3)$ Compute $D_1 = C_7 \oplus h_6(yC_6 \ ID_{UP})$ Check $C_{10}? = h_8(D_1 \ yC_6)$ Compute $D_2 = C_8 - 2D_1, C_3 - D_1, C_5$ $D_3 = C_9 - D_1, C_3 - 2D_1, C_5$ get $W_1 = \sum_{i=1}^k v_i$ from D_2 get $W_2 = \sum_{i=1}^k w_i$ from D_3 Compute the mean value and variance value according to k
	Check t_2 , get k, t_3, t_4 $C_1 = B_6 \oplus h_5(z_j B_5 \ ID_{FD_j})$ Check $B_8? = h_7(C_1 \ ID_{FD_j} \ B_7 \ t_2)$ Compute $C_2 = \sum_{i=1}^k A_{i,1}, C_3 = C_1 C_2$ $C_4 = \sum_{i=1}^k A_{i,2}, C_5 = C_1 C_4$ $C_6 = B_5, C_7 = B_7, C_8 = \sum_{i=1}^k A_{i,5}$ $C_9 = \sum_{i=1}^k A_{i,6}, C_{10} = B_9$ $C_{11} = r_4 P, C_{12} = h_9(r_4 Y \ t_3) \oplus k$ $C_{13} = h_{10}(C_3 \ C_7 \ C_8 \ C_9 \ C_{10} \ k \ key_j \ t_3)$ $M_3 = (C_3, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, t_4, ID_{FD_j})$ \leftarrow $C_{12}, C_{13}, t_3, ID_{FD_j}$		

analysis of the games. The results show how the scheme keeps IND-CPA security. The proposed scheme meets the CPA security requirements if the polynomial time adversary \mathcal{A} has negligible probability to win the games. A simulator \mathcal{S} is used to provide the random oracle query service and \mathcal{A} makes queries to try to break the IND-CPA security. All games can be divided into five phases as follows:

- (1) Initialization: \mathcal{S} generates system parameters including G with generator P and large prime order q , public keys of FD_j , UP, and TS, identities of smart meters, fog devices, trusted server and utility provider, secret keys between FD_j and UP, and hash functions. We define the public keys of TS, UP and FD_j are $X = a_{TS}P$, $Y = a_{UP}P$, and $Z = a_jP$, respectively, where a_{TS} , a_{UP} , and a_j are unknown.
- (2) Query 1: \mathcal{A} queries the hash oracles, and \mathcal{S} returns the results.
- (3) Challenge: \mathcal{A} selects fresh information $info^0$ and $info^1$ to \mathcal{S} . \mathcal{S} then chooses a bit $\omega \in \{0, 1\}$. And $info^\omega$ is used to generate the corresponding message in the game.
- (4) Query 2: it is same as query 1.
- (5) Guess: \mathcal{A} should give a bit ω' as the result of guessing ω . If $\omega = \omega'$, \mathcal{A} wins the game.

\mathcal{A} knows all public parameters and all identities of participants. He will ask h_i oracle for q_{h_i} ($i = 1, 2, \dots, 10$) times. There are hash lists for storing \mathcal{A} 's corresponding hash queries. For instance, L_{h_i} stores $(i, str, result)$, where h_i is queried by \mathcal{A} and $result$ is the hash result of str . The advantage for \mathcal{A} breaking the message M_b ($b = 1, 2, 3$) is denoted as $A dv_{M_b}^{IND-CPA}(\mathcal{A}) = \Pr[\omega = \omega'] - 1/2$. In the following proofs, we only consider the extra probabilities beyond 1/2. As mentioned in Section 2.2, $\varepsilon = A dv_{\mathcal{A}}^{GDH}(t)$, and we employ the probability to express the hard level of solving the GDH problem in the following theorems. That probability is used when we find the tuple format (aP, bP, abP) occurring in the hash list L_{h_i} , where a and b are unknown, as we mentioned also in Section 2.2.

4.2. Proof of CPA

Theorem 1. *The data encryption phase is IND-CPA secure and $A dv_{M_1}^{IND-CPA}(\mathcal{A}) \leq (q_{h_1}^2 + q_{h_2}^2 + q_{h_3}^2 + 2(q_{h_1} + q_{h_2} + q_{h_3}))/2^{l_s+1} + q_{h_1}q_{h_2}q_{h_3}\varepsilon^2$.*

Proof. The concrete operations are as follows:

- (1) Initialization: \mathcal{S} produces parameters as mentioned in Section 4.1.
- (2) Query 1: \mathcal{A} makes h_1 , h_2 , and h_3 queries with the string str . \mathcal{S} searches if there is the existed queried tuple $(i, str, result)$ in L_{h_i} ($i = 1, 2, 3$). If true, $result$ will be returned. Otherwise, \mathcal{S} selects $result \in \{0, 1\}^{l_s}$, and the tuple $(i, str, result)$ is written into the list.
- (3) Challenge: \mathcal{A} selects v_i^0 and v_i^1 and submits them to \mathcal{S} . \mathcal{S} produces M_1^ω as the following operations. First,

a bit ω is chosen and v_i^ω is used to produce the message.

Then, the corresponding message is sent to \mathcal{A} .

- (4) Query 2: \mathcal{A} makes h_1 , h_2 , and h_3 queries again until the numbers q_{h_i} ($i = 1, 2, 3$) are reached.
- (5) Guessing: \mathcal{A} gives a bit ω' .

If $\omega = \omega'$, we divide the advantage of \mathcal{A} 's guessing into three parts. First, to avoid the collision of hash results, the upper probability is $(q_{h_1}^2 + q_{h_2}^2 + q_{h_3}^2)/2^{l_s+1}$, based on birthday paradox. Second, if the hash results are guessed correctly without oracle queries, the probability is at most $(q_{h_1} + q_{h_2} + q_{h_3})/2^{l_s}$. Finally, if \mathcal{A} could judge the message by generating a correct one for comparison, $(1, r_1X, *)$, $(2, r_2X, *)$, and $(3, r_1X \| * \| r_2X \| *, A_{i,7})$ can be obtained from L_{h_1} , L_{h_2} , and L_{h_3} , respectively. So, the condition of \mathcal{A} querying the correct strings is solving both GDH problems, and the probability is $q_{h_1}q_{h_2}q_{h_3}\varepsilon^2$. Then, the theorem is deduced. \square

Theorem 2. *The aggregation key generation phase is IND-CPA secure and $A dv_{M_2}^{IND-CPA}(\mathcal{A}) \leq (q_{h_4}^2 + 2q_{h_4})/2(q-1) + (q_{h_5}^2 + q_{h_6}^2 + q_{h_7}^2 + q_{h_8}^2 + 2(q_{h_5} + q_{h_6} + q_{h_7} + q_{h_8}))/2^{l_s+1} + q_{h_4}q_{h_5}q_{h_6}q_{h_8}\varepsilon^2$.*

Proof. The concrete operations are as follows:

- (1) Initialization: \mathcal{S} produces parameters as mentioned in Section 4.1.
- (2) Query 1: \mathcal{A} makes h_4 , h_5 , h_6 , h_7 , and h_8 queries with the string str . \mathcal{S} searches if there is the existed queried tuple $(i, str, result)$ in L_{h_i} ($i = 4, 5, 6, 7, 8$). If true, $result$ will be returned. Otherwise, \mathcal{S} selects $result \in \{0, 1\}^{l_s}$ ($i = 5, 6, 7, 8$) or $result \in Z_q^*$ ($i = 4$), and the tuple $(i, str, result)$ is written into the list.
- (3) Challenge: \mathcal{A} selects strings s^0 and s^1 and sends them to \mathcal{S} . Then, M_2^ω is produced as follows. First, \mathcal{S} chooses a bit ω and s^ω is used to produce the message M_2^ω . Then, the corresponding message is sent to \mathcal{A} .
- (4) Query 2: \mathcal{A} makes h_4 , h_5 , h_6 , h_7 , and h_8 queries again until the numbers q_{h_i} ($i = 4, 5, 6, 7, 8$) are reached.
- (5) Guessing: \mathcal{A} gives a bit ω' .

If $\omega = \omega'$, we divide the advantage of \mathcal{A} 's guessing into three parts. First, to avoid the collision of hash results, the upper probability is $q_{h_4}^2/2(q-1) + (q_{h_5}^2 + q_{h_6}^2 + q_{h_7}^2 + q_{h_8}^2)/2^{l_s+1}$, based on birthday paradox. Second, if the hash results are guessed correctly without oracle queries, the probability is at most $q_{h_4}/q - 1 + (q_{h_5} + q_{h_6} + q_{h_7} + q_{h_8})/2^{l_s}$. Finally, if \mathcal{A} could judge the message by generating a correct one for comparison, $(4, r_3Y \| *, *) \in L_{h_4}$, $(5, r_3Z_j \| *, *) \in L_{h_5}$, $(6, r_3Y \| *, *) \in L_{h_6}$, and $(8, * \| r_3Y, *) \in L_{h_8}$ could be found. The probability is at least $1/q_{h_i}$ ($i = 4, 5, 6, 8$). Like the analysis in Theorem 1, the probability is $q_{h_4}q_{h_5}q_{h_6}q_{h_8}\varepsilon^2$. So, we get the theorem. \square

Theorem 3. *The aggregation phase is IND-CPA secure and $A \text{ } dv_{M_3}^{IN D-CPA}(\mathcal{A}) \leq q_{h_4}^2 + 2q_{h_4}/2(q-1) + (q_{h_6}^2 + q_{h_8}^2 + q_{h_9}^2 + q_{h_{10}}^2 + 2(q_{h_6} + q_{h_8} + q_{h_9} + q_{h_{10}}))/2^{l_s+1} + q_{h_4}/q-1 + q_{h_4}q_{h_6}q_{h_8}q_{h_9}\epsilon^2$.*

Proof. The concrete operations are as follows:

- (1) Initialization: \mathcal{S} produces parameters as mentioned in Section 4.1.
- (2) Query 1: according to M_3 , all related hash functions h_i ($i = 4, 6, 8, 9, 10$) are queried by \mathcal{A} . \mathcal{S} searches if there is the existed queried tuple $(i, str, result)$ in L_{h_i} ($i = 4, 6, 8, 9, 10$). If true, $result$ will be returned. Otherwise, \mathcal{S} selects $result \in \{0, 1\}^l$ ($i = 6, 8, 9, 10$) or $result \in Z_q^*$ ($i = 4$), and the tuple $(i, str, result)$ is written into the list.
- (3) Challenge: \mathcal{A} selects v_i^0 and v_i^1 and submits them to \mathcal{S} . Then, M_3 is produced as follows: first, a bit ω is chosen, and s^ω is used to produce M_3^ω .
- (4) Query 2: \mathcal{A} makes h_4, h_6, h_8, h_9 , and h_{10} queries again until the numbers q_{h_i} ($i = 4, 6, 8, 9, 10$) are reached.
- (5) Guessing: \mathcal{A} gives a bit ω' .

If $\omega = \omega'$, we divide the advantage of \mathcal{A} 's guessing into three parts. First, to avoid the collision of hash results, the upper probability is $q_{h_4}^2/2(q-1) + (q_{h_6}^2 + q_{h_8}^2 + q_{h_9}^2 + q_{h_{10}}^2)/2^{l_s+1}$. Second, if the hash results are guessed correctly without oracle queries, the probability is at most $q_{h_4}/q-1 + (q_{h_6} + q_{h_8} + q_{h_9} + q_{h_{10}})/2^{l_s}$. Finally, if \mathcal{A} could judge the message by generating a correct one for comparison, $(4, r_3Y \| *, *) \in L_{h_4}$, $(6, r_3Y \| *, *) \in L_{h_6}$, $(8, * \| r_3Y, *) \in L_{h_8}$, and $(9, r_4Y \| *, *) \in L_{h_9}$ could be found. The probability is at least $1/q_{h_i}$ ($i = 4, 6, 8, 9$). Like the analysis in Theorem 1, the probability is $q_{h_4}q_{h_6}q_{h_8}q_{h_9}\epsilon^2$. So, we get the theorem. \square

5. Security Property Expression

The security properties are illustrated, and we compare our scheme with some recent ones [18, 20, 22, 27, 29]. Readers may search for some concrete details in corresponding studies. The results are given in Table 3. \checkmark denotes the scheme meets the security property, while \times denotes the opposite case. If the property is not fit for the scheme, \emptyset is used. P1–P7 denote confidentiality, user anonymity, traceability, data aggregation, scalability, against internal attacks, and replay attacks, respectively. From the results, we see that the proposed scheme meets all the security requirements.

5.1. Confidentiality. First, we discuss our scheme. For M_1 , if \mathcal{A} wants to get v_i and ID_{SM_i} , he should know TS's secret key x to get r_1X and r_2X from r_1P and r_2P . For M_2 , if \mathcal{A} wants to get the critical element $xh_4(r_3Y||s)$, he must know any of the private keys y or z_j . For M_3 , if \mathcal{A} wants to get the element $h_4^{-1}(r_3Y||s)$, he must know any private keys same as in M_2 .

Moreover, in [29], the keys for reencryption are directly sent to the public cloud server in the rekey phase, and they are exposed in the channel. Also, there is no authentication between the public cloud server and UP, so the data

TABLE 3: Security property.

	P1	P2	P3	P4	P5	P6	P7
Ours	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark
[29]	\times	\checkmark	\checkmark	\checkmark	\times	\checkmark	\checkmark
[27]	\checkmark	\checkmark	\times	\checkmark	\emptyset	\checkmark	\checkmark
[18]	\times	\times	\checkmark	\times	\checkmark	\emptyset	\checkmark
[20]	\checkmark	\times	\times	\checkmark	\emptyset	\checkmark	\checkmark
[22]	\checkmark	\emptyset	\times	\checkmark	\emptyset	\checkmark	\checkmark

generated in this phase can be changed, e.g., adding P on c_2 . UP cannot check the correctness of data. Last, the consumption information is leaked. In the Enc phase, power usage quan and the cost fee are sent by $rX + \text{quan} \cdot P$ and $rX + \text{fee} \cdot P$, respectively. \mathcal{A} could calculate $\alpha = (\text{fee} - \text{quan})$ by subtraction and Pollard rho algorithm. Then, \mathcal{A} gets $\alpha = \text{quan}(\text{fee}/\text{quan} - 1)$. Fee/quan is just the price of electricity in the fixed period, and the power usage quan can be deduced.

5.2. User Anonymity. In our scheme, ID_{SM_i} is hidden by $h_2(r_2X)$. \mathcal{A} should know TS's secret key x to calculate r_2X based on $A_{i,2} = r_2X$. But in [18], the user identity may be exposed. Information of users who consume more electricity than the threshold will be exposed in the channel, including identity and power consumption. Using more electricity is not a crime, and it is unsuitable to publish user information simply due to such a case. Also, in [20], user identity is in plaintext obviously. So, we use \times for the two schemes. In [22], user identity is not needed in the entire scheme, and \emptyset is used.

5.3. Trace Ability. We set Section 3.3 to make the power consumption of the user clear and to satisfy the basic function of the smart grid. TS computes $B_2 = xA_{i,2}$ and $ID_{SM_i} = A_{i,4} \oplus h_2(B_2)$ to get the identity ID_{SM_i} , accompanying with the consumption v_i . Such calculations can make TS get the (ID_{SM_i}, v_i) tuple and know the fee of the user consumption.

However, in [20, 22, 27], no entity except the smart meter itself knows the power consumption. How to affirm the user's fee for power usage is a difficult thing in the above three mentioned schemes.

5.4. Data Aggregation. Same as [20, 22, 27, 29], our scheme has the part of data aggregation, in order to analyze the statistical data on UP. But, in [18], data aggregation is not focused.

5.5. Scalability. In Section 3.3, TS does not require exhaustive searching for checking the identity. Or we say that no extra computation is before searching, even a hash result. But, in [29], if UP questions for some smart meter, the trusted server should use a scalar multiplication and a hash function to check all indexed users. We use \times at the corresponding blank. But, such property does not fit for [20, 22, 27], since no tracking operation is in any of them.

5.6. Against Internal Attack. We make analysis based on the fourth item in Section 2.4. Since no fine-grained data appear in UP, if \mathcal{A} colludes with UP to crack the messages, \mathcal{A} still faces DL problem and GDH problem to get the timely private data from the messages due to lack of the private key x . A similar situation for colluding with fog devices can be deduced.

5.7. Against Replay Attack. To avoid replay attack, timestamps $t_1, t_2,$ and t_3 are used in our scheme. Once \mathcal{A} tries to modify any message, he must change the element for checking. In $M_1, A_{i,7}$ contains t_1 , where r_1X and r_2X are also included. Computing the two results mean that two GDH problems should be cracked, based on $A_{i,1} = r_1P, A_{i,2} = r_2P$, and the public key of TS. In M_2, t_2 is used in B_8 , where $r_3Y, s,$ and x are also referred. Besides guessing s and x , computing r_3Y also means that one GDH problem should be cracked, based on $B_5 = r_3P$ and the public key of UP. Similar situation occurs in $M_3. t_3$ is used in C_{13} , where r_4Y is needed to crack based on $C_{11} = r_4P$ and public key of UP.

6. Performance Evaluation

In this section, we compare our scheme with [20, 22, 29] via time cost and communication cost. The test platform is MIRACL Library under Ubuntu 20, with Intel(R) Core(TM) i5-9400H CPU 2.50 GHz and 16.0 GB memory. The length of points on the elliptic curve is 320 bits, where the order of the additive group is 160 bits long. The timestamps have 64 bits, while all identities of devices have 32 bits and the hash function is Sha2-256. We use AES as the symmetric encryption/decryption algorithm in [29]. The symbols of time cost are given in Table 4, and the time cost comparison is given in Table 5. We use PH1–PH5 to express phases including data encryption, consumption affirmation, aggregation key generation, aggregation, and aggregated data decryption. k is the number of smart meters belonging to one fog device, and m is the number of fog devices.

From Table 5, we see that our scheme costs less than [29] in PH2 and PH5. In PH1, we use two random numbers for scalar multiplications. Since the power usage and fee are relatively small numbers, it is probable to calculate the private consumption, as we have demonstrated in Section 5.1 for [29]. In PH3, we add the operations to protect the rekeys, while there is no such idea in [29]. In PH5, since each submission of FD_j will cost the same time, we only list one round calculation in UP. Our scheme costs only a little more than [29]. Here, we should claim that there is no phase like PH2 and PH3 in schemes [20, 22], so the corresponding blanks are empty. In PH5, only one Pollard rho algorithm is used in [20, 22], since the final result is aimed at the aggregation value of consumption that is different from our scheme and [29], which also have another target of the variance value. In PH4, as long as a natural condition $k > 2$ exists, we see that our scheme costs better than [20, 22, 29]. The verifications of the process are settled on TS for our scheme and [29], unlike [20, 22], such the verification is put

TABLE 4: Time cost for referred cryptographical operations.

Symbol	Meaning	Time (ms)
T_s	Time of one scalar multiplication on G	0.2898
T_{se}	Time of symmetric encryption/decryption	0.01368
T_{2e}	Time of double exponentiation in the group	0.3528
T_h	Time of Sha2-256	0.003128
T_i	Time of one inversion in a group	0.02673
T_m	Time of one multiplication in a group	0.00169
T_a	Time of one point addition on G	0.00184
T_{PR}	Time of one pollard rho algorithm	61.4667

on the media devices. The reason is that the cost of user consumption needs to be calculated and affirmed.

We illustrate the concrete communication cost here. For our scheme, in the data encryption phase, M_1 has $320 * 4 + 256 * 3 + 64 = 2112$ bits, and there are totally 2112km bits in one period. In the consumption affirmation phase, all FDs submit the collected messages to TS, and the total information is the same as the last step. In the aggregation key generation phase, M_2 has $320 + 256 * 4 + 64 = 1408$ bits, and there are totally 1408m bits in one transmission. In the aggregation phase, M_3 has $320 * 6 + 256 * 4 + 32 + 64 * 2 = 3104$ bits, and there are totally 3104m bits. So, the whole communication cost is 4224 km + 4512 m bits. We evaluate the transmission situation. Generally, channels between smart meters and their corresponding fog devices are considered to be wireless. Suppose a normal building for residents, about 30 floors, and generally, it has less than 200 houses. Each smart meter submits its data in every 15 minutes [35]. In the 15-minute period, there are less than $2112 * 200 = 422400$ bits in total or less than 470 bps, that is, a very small data rate. Second, we consider wired transmission messages M_2 and M_3 . Suppose there are 100000 fog devices to send the messages. The total data volume is $100000 * 4512 = 451200000 < 500$ Mbits. Note such volume is for 15 minutes, and the fiber can support 10 Gbps rate [36]. So, the communication cost in our scheme is practical.

On the other side, for scheme in [29], in the enc phase, the message has $256 + 32 + 256 + 320 * 3 + 64 = 1568$ bits and in total 1568km bits. In the TTP-Dec phase, the message has $256 + 64 + 1568 = 1888$ bits and in total 1888 km bits. In the rekey phase, the message has $160 * 2 = 320$ bits and in total 320m bits. In the LiAgg-ReEnc phase, the message has $320 * 4 + 160 = 1440$ bits and in total 1440m bits. So, the whole communication cost is 3456km + 1760m bits. All could see that our scheme costs more than the scheme in [29]. However, according to our analysis of Section 4 and Section 5, our scheme is CPA secure and meets common security properties. Moreover, in [29], only CPA security is proved only for Enc and LiAgg-ReEnc phases. How to transmit the aggregation key from trusted server (corresponding to TTP in [29]) to FD (corresponding to PCS in [29]) is not demonstrated. If the secure channel is used, such cost is high. So, we consider that the consumption information is transmitted in public channels. At the same time, the whole scheme in [29] does not even reach CPA security, and the cost of time and communication increases in our scheme is rewarding.

TABLE 5: Time cost comparison (ms).

	PH1	PH2	PH3	PH4	PH5
Ours	$6T_s + 3T_h + 6T_a = 1.7591$	$(4T_s + 3T_h + 6T_a)km \leq m = 1.1795km$	$3T_s + 6T_h + T_i = 0.9148$	$5T_s + 4T_h + 4(k-1)T_a$	$4T_s + 4T_h + 2T_m + 6T_a + 2T_{PR} = 124.1195$
[29]	$5T_s + T_{se} + 2T_h = 1.4688$	$(3T_s + T_{se} + T_h + 2T_a + 2T_{PR})km = 123.8264km$	$T_s + T_h + T_i + 2T_m = 0.3230$	$2(k+1)T_s + 4kT_a = 0.5869k + 0.5796$	$2T_m + 2T_a + 2T_{PR} = 124.1593$
[20]	$T_{2e} + T_e + T_h + T_m = 0.6602$			$(k+1)T_{2e} + 2T_e + (4k-1)T_m + 3kT_h = 0.3690k + 0.6537$	$T_{2e} + T_e + t_m + T_h + T_i + T_{PR} = 62.516$
[22]	$T_{2e} + T_e + T_h + T_m = 0.9644$			$2kT_{2e} + (k+1)T_e + (k+1)T_h + (2k-1)T_m = 1.0147k + 0.304$	$T_{2e} + T_e + T_h + 3T_m + T_{PR} = 62.1302$

Schemes in [20, 22] belong to the same type. They both lack the communication between the servers which calculates the aggregation data and the media device, like collector or gateway. In Ding et al.'s scheme [20], the message from the smart meter to the collector has $1024 + 1024 + 1024 + 1024 + 64 + 32 = 4192$ bits and in total 4192 km bits. The message from the collector to the electricity service provider has the same construction as the last, and there are 4192 m bits. Finally, we could see that 4192 km + 4192 m bits occur in the whole process. Similarly, in Wang et al.'s scheme [22], the entire communication cost is 3616 km + 3552 m bits. However, both of them have weaknesses including lack of user anonymity and no consideration of traceability, which we have mentioned in Section 5. Also, no statistic data are deduced on service providers on both of them [20, 22].

Above all, our scheme is better than other schemes in [20, 22, 29] for security and practicality.

7. Conclusion

In this study, based on industrial Internet of Things, we give a novel scheme on smart grid, getting user power consumption and statistical data simultaneously. Formal proof with random oracle condition is shown to illustrate CPA security of the presented scheme. We also compare our scheme with some relative schemes for smart grid, and all can see ours is the only one that satisfies the security requirements. Via time and communication cost study, we express that our scheme performs well and it is fit for practicality.

The security level is an important index to evaluate the scheme. In the future, we will try to enhance the security level of such scheme, e.g., designing a new scheme that resists chosen-ciphertext attack and meets the practical requirements like tracking the concrete power consumption of every user and not only owns the function of aggregating data.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Fan Wu was supported by Xiamen University Tan Kah Kee College Scientific Research Foundation (JG2022SRF02). Professor Xiong Li was supported by the National Natural Science Foundation of China (62072078).



References

- [1] M. Badra and S. Zeadally, "Lightweight and efficient privacy-preserving data aggregation approach for the smart grid," *Ad Hoc Networks*, vol. 64, pp. 32–40, 2017.
- [2] F. Wu, L. Xu, X. Li, S. Kumari, M. Karuppiah, and M. S. Obaidat, "A lightweight and provably secure key agreement system for a smart grid with elliptic curve cryptography," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2830–2838, 2019.
- [3] F. Wu, X. Li, L. Xu, S. Kumari, D. Lin, and J. J. P. C. Rodrigues, "An anonymous and identity-trackable data transmission scheme for smart grid under smart city notion," *Annals of Telecommunications*, vol. 75, no. 7-8, pp. 307–317, 2020.
- [4] F. Wu, X. Li, L. Xu, and S. Kumari, "A privacy-preserving scheme with identity traceable property for smart grid," *Computer Communications*, vol. 157, pp. 38–44, 2020.
- [5] Y. Gong, Y. Cai, Y. Guo, and Y. Fang, "A privacy-preserving scheme for incentive-based demand response in the smart grid," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1304–1313, 2016.
- [6] M. M. Fouda, Z. M. Fadlullah, N. Kato, R. Lu, and X. S. Shen, "A lightweight message authentication scheme for smart grid communications," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 675–685, 2011.
- [7] K. Mahmood, S. Ashraf Chaudhry, H. Naqvi, T. Shon, and H. Farooq Ahmad, "A lightweight message authentication scheme for smart grid communications in power sector," *Computers & Electrical Engineering*, vol. 52, pp. 114–124, 2016.
- [8] J.-L. Tsai and N.-W. Lo, "Secure anonymous key distribution scheme for smart grid," *IEEE Transactions on Smart Grid*, vol. 7, pp. 906–914, 2016.
- [9] V. Odelu, A. K. Das, M. Wazid, and M. Conti, "Provably secure authenticated key agreement scheme for smart grid," *IEEE Transactions on Smart Grid*, vol. 9, p. 1, 2016.
- [10] D. He, H. Wang, M. K. Khan, and L. Wang, "Lightweight anonymous key distribution scheme for smart grid using elliptic curve cryptography," *IET Communications*, vol. 10, no. 14, pp. 1795–1802, 2016.

- [11] X. Li, F. Wu, S. Kumari, L. Xu, A. K. Sangaiah, and K.-K. R. Choo, "A provably secure and anonymous message authentication scheme for smart grids," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 242–249, 2019.
- [12] K. Mahmood, S. A. Chaudhry, H. Naqvi, S. Kumari, X. Li, and A. K. Sangaiah, "An elliptic curve cryptography based lightweight authentication scheme for smart grid communication," *Future Generation Computer Systems*, vol. 81, pp. 557–565, 2018.
- [13] D. Abbasinezhad-Mood and M. Nikooghadam, "Design and hardware implementation of a security-enhanced elliptic curve cryptography based lightweight authentication scheme for smart grid communications," *Future Generation Computer Systems*, vol. 84, pp. 47–57, 2018.
- [14] D. Abbasinezhad-Mood and M. Nikooghadam, "An anonymous ecc-based self-certified key distribution scheme for the smart grid," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 7996–8004, 2018.
- [15] F. Wu, X. Li, L. Xu, P. Vijayakumar, and N. Kumar, "A novel three-factor authentication protocol for wireless sensor networks with iot notion," *IEEE Systems Journal*, vol. 15, no. 1, pp. 1120–1129, 2021.
- [16] F. Diao, F. Zhang, and X. Cheng, "A privacy-preserving smart metering scheme using linkable anonymous credential," *IEEE Transactions on Smart Grid*, vol. 6, no. 1, pp. 461–467, 2015.
- [17] H. Qu, P. Shang, X. J. Lin, and L. Sun, "Cryptanalysis of a privacy-preserving smart metering scheme using linkable anonymous credential," *IACR Cryptology ePrint Archive 2015*, vol. 1066, 2015.
- [18] Z. Sui, M. Niedermeier, and H. de meer, "Tai: a threshold-based anonymous identification scheme for demand-response in smart grids," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3496–3506, 2018.
- [19] S. Ge, P. Zeng, and K.-K. R. Choo, "An enhanced anonymous identification scheme for smart grids," in *Proceedings of the International Conference on Applications and Techniques in Cyber Security and Intelligence*, pp. 329–337, Fuyang, China, June 2021.
- [20] Y. Ding, B. Wang, Y. Wang, K. Zhang, and H. Wang, "Secure metering data aggregation with batch verification in industrial smart grid," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6607–6616, 2020.
- [21] Y. Su, Y. Li, J. Li, and K. Zhang, "Lceda: Lightweight and Communication Efficient Data Aggregation Scheme for Smart Grid," *IEEE Internet of Things Journal*, vol. 8, pp. 15639–15648, 2021.
- [22] J. Wang, L. Wu, S. Zeadally, M. K. Khan, and D. He, "Privacy-preserving data aggregation against malicious data mining attack for iot-enabled smart grid," *ACM Transactions on Sensor Networks*, vol. 17, no. 3, pp. 1–25, 2021.
- [23] D. He, N. Kumar, S. Zeadally, A. Vinel, and L. T. Yang, "Efficient and privacy-preserving data aggregation scheme for smart grid against internal adversaries," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2411–2419, 2017.
- [24] D. He, S. Zeadally, H. Wang, and Q. Liu, "Lightweight Data Aggregation Scheme against Internal Attackers in Smart Grid Using Elliptic Curve Cryptography," *Wireless Communications and Mobile Computing 2017*, vol. 13, 2017.
- [25] Z. Wang, "An identity-based data aggregation protocol for the smart grid," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2428–2435, 2017.
- [26] H. Shen, M. Zhang, and J. Shen, "Efficient privacy-preserving cube-data aggregation scheme for smart grids," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1369–1381, 2017.
- [27] R. Lu, K. Heung, A. H. Lashkari, and A. A. Ghorbani, "A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot," *IEEE Access*, vol. 5, pp. 3302–3312, 2017.
- [28] Y. Liu, W. Guo, C.-I. Fan, L. Chang, and C. Cheng, "A practical privacy-preserving data aggregation (3pda) scheme for smart grid," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1767–1774, 2019.
- [29] H. Wang, D. He, and S. Zhang, "Balanced anonymity and traceability for outsourcing small-scale data linear aggregation in the smart grid," *IET Information Security*, vol. 11, no. 3, pp. 131–138, 2017.
- [30] S. Wong, "Electricity consumption in china between 2010 and 2019," 2020, <https://www.statista.com/statistics/302203/china-electricity-consumption/>.
- [31] D. J. Bernstein and T. Lange, "Computing small discrete logarithms faster," in *Proceedings of the International Conference on Cryptology in India*, pp. 317–338, Kolkata, India, December 2012.
- [32] J. M. Pollard, "Monte Carlo methods for index computation (modp)," *Mathematics of Computation*, vol. 32, no. 143, pp. 918–924, 1978.
- [33] D. He, N. Kumar, and J.-H. Lee, "Privacy-preserving data aggregation scheme against internal attackers in smart grids," *Wireless Networks*, vol. 22, no. 2, pp. 491–502, 2016.
- [34] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes," in *Proceedings of the Annual International Cryptology Conference*, pp. 26–45, Springer, Santa Barbara, California, August 1998.
- [35] Wikipedia, "Smart Meter," 2018, https://en.wikipedia.org/wiki/Smart_meter.
- [36] B. Mitchell, "The Role of Fiber Optic Cables in Computer Networking," 2018, <https://www.lifewire.com/fiber-optic-cable-817874>.

Research Article

TADW: Traceable and Anti-detection Dynamic Watermarking of Deep Neural Networks

Jinwei Dong ^{1,2}, He Wang,¹ Zhipeng He,³ Jun Niu,⁴ Xiaoyan Zhu,⁵ and Gaofei Wu ^{1,2}

¹School of Cyber Engineering, Xidian University, Xi'an, China

²Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, China

³School of Cyberspace Security, Xi'an University of Posts & Telecommunications, Xi'an, China

⁴School of Computer, Xidian University, Xi'an, China

⁵School of Telecommunications Engineering, Xidian University, Xi'an, China

Correspondence should be addressed to Gaofei Wu; wugf@nipc.org.cn

Received 31 March 2022; Accepted 30 April 2022; Published 16 June 2022

Academic Editor: AnMin Fu

Copyright © 2022 Jinwei Dong 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.

Deep neural networks (DNN) with incomparably advanced performance have been extensively applied in diverse fields (e.g., image recognition, natural language processing, and speech recognition). Training a high-performance DNN model requires a lot of training data and intellectual and computing resources, which bring a high cost to the model owners. Therefore, illegal model abuse (model theft, derivation, resale or redistribution, etc.) seriously infringes model owners' legitimate rights and interests. Watermarking is considered the main topic of DNN ownership protection. However, almost all existing watermarking works apply solely to image data. They do not trace the unique infringing model, and the adversary easily detects these ownership verification samples (trigger set) simultaneously. This paper introduces TADW, a dynamic watermarking scheme with tracking and antidetection abilities in the deep learning (DL) textual domain. Specifically, we propose a new approach to construct trigger set samples for antidetection and innovatively design a mapping algorithm that assigns a unique serial number (SN) to every watermarked model. Furthermore, we implement and detailedly evaluate TADW on 2 benchmark datasets and 3 popular DNNs. Experiment results show that TADW can successfully verify the ownership of the target model at a less than 0.5% accuracy cost and identify unique infringing models. In addition, TADW is excellently robust against different model modifications and can serve numerous users.

1. Introduction

In recent years, deep learning has achieved rapid development and has shown great success in various domains, such as computer vision [1–4], natural language processing [5–8], and speech recognition [9–11]. Global well-known companies such as Amazon, Google, Microsoft, and IBM have already provided cloud-based Machine Learning as a Service (MLaaS). According to a related report, machine learning-related industries are expected to generate trillions of dollars in business value by 2022. At the same time, with the significant success of DNNs, the size of the dataset used for the training is getting pretty enormous, and the structure of models is also more complicated so that the

training cost of many models is incredibly high. For example, Google's C4 dataset based on around 20TB of original Common Crawl's web crawl corpus (<https://commoncrawl.org>) takes 335 CPU-days to clean data. Another example is that the models of text generation CTRL containing 1.63 billion parameters were trained for about two weeks on 2048 TPU cores. In addition, the design of the DNN structure needs much mental work, and many experiments are required to determine the optimal parameters of the model.

The growing value of DNN increased concerns about model abuse. Nowadays, DL provides services to users mainly in two ways: firstly, companies sell DNN models as a product; secondly, users interact with these models that

companies deploy in MLaaS platforms through the API. In these situations, the adversary can acquire a model by legal purchase or illegal channels (e.g., model inversion [12] and inference attacks [13]) and then provide illegal services (online services) to users for profit. In either case, it seriously infringes the intellectual property (IP) of the legitimate owners of models and even affects the market order of MLaaS. Therefore, the IP of DNNs needs solid and lasting protection.

Digital watermarking technology [14, 15] has powerful anticounterfeiting and antitheft capabilities and has been immensely leveraged to protect the IP of multimedia content. Motivated by such an intuition, DNN watermarking [16] has been proposed to protect the IP of DNNs. The workflow of watermarking is generally divided into two stages: watermark embedding and ownership verification. In the watermark embedding stage, the model owner purposely introduces the trigger set (i.e., watermarking is a trigger in a backdoor) composed of some aberrant input-output pairs (x, y) that only they know in the model’s training phase (analog to poisoning or backdoor attacks [17]). In the ownership verification stage, model owners query the suspicious model f on these specific inputs x and judge whether the model is infringing by comparing $f(x) = y$ returned by the model.

So what requirements should an exemplary watermarking meet? The answer is feasibility, fidelity, undetectability, uniqueness, robustness, and scalability. However, DNN watermarking technology is still in the early stages of development, and the existing watermarking schemes are immature and flawed. DNN watermarking algorithms [16, 18, 19] are designed in white-box ways, but the stolen models are usually deployed on a remote server, indicating that the model owner is unable to access the model parameters. The trigger set samples generated in [20] differ significantly from the clean (unwatermarked) samples. This means that the adversary can easily detect these outlier samples. Reference [21] proposed a blind-watermark framework aiming to amplify the feasibility of watermarking, but it cannot guarantee the uniqueness of the watermarked model. Moreover, the previous research is almost all limited to the DL image field. DL is also extensively exploited in the text area, such as machine translation and speech recognition. However, related watermarking studies are incredibly scarce. Reference [22] introduced a textual watermarking scheme that is not capable of uniqueness and undetectability. In summary, no existing watermarking schemes can meet all the requirements mentioned above.

This paper proposed TADW, a new dynamic DNN watermarking scheme that can fulfill all the requirements mentioned above. Specifically, we innovatively collect many texts from the real world as our trigger set sample pool and select a specific number of samples from it according to the filtering rules to form the final trigger set. We also employ a multibit bit string as the distinctive mark of a watermarked model, namely, the serial number. To assign a unique SN to every model, we devised an ingenious mapping method between trigger set and SN, representing different SN using the same trigger set assigned different class labels. Furthermore, we optimize the method for embedding

watermark based on an experimentally validated watermark embedding scheme [17] and use the training set and trigger set to train the model according to the set ratio. Finally, we implemented TADW and evaluated it against the indicators above. The experiments show that TADW can successfully verify the model’s IP and trace the unique infringing models. What is more, the trigger set well avoids detection by adversaries. TADW also achieves high performance on fidelity and robustness. We summarize our contributions as follows:

- (i) We propose a novel dynamic watermarking scheme TADW for IP protection of DNN in the DL textual domain. Our scheme can embed a unique SN for each model to track and identify unique infringing models from many infringing models using the same IP.
- (ii) We implemented TADW on 2 benchmark text datasets and 3 popular text classification models. Our experiments show that TADW enables successfully verifying the DNN model’s IP.
- (iii) We made a detailed evaluation to corroborate the feasibility, fidelity, undetectability, uniqueness, robustness, and scalability of TADW. The experiments show that TADW can achieve remarkable performance in these aspects.

2. Related Work

The existing watermarking algorithms, which are mainly based on black-box (only model outputs are obtainable) or white-box (internal model parameters are accessible), have been devised in the DL image field, but few watermarking methods in the textual domain. We now summarize previous works on DNN watermarking.

2.1. Image Watermarking

2.1.1. White Box. Uchida et al. [16] first proposed a framework to watermark models in a white-box way. The authors interpret the watermark as a T -bit string $\{0, 1\}^T$ and impose a statistical bias on specific parameters to represent the watermark by adding a new loss term to the loss function. Existing works [18, 19] make the improvements inheriting their work and adopt adding new loss items to embed the watermark. However, these schemes all have a common disadvantage; that is, anyone knowing the methodology can remove the watermark without knowing the watermarking information leveraged to inject it. For instance, Wang et al. [23] have proved that these watermarks can be detected and removed by overwriting the statistical bias. Fan et al. [24] added a particular “passport” layer to the model for the model IP verification, such that the model performs poorly when passport layer weights are not present. Nevertheless, the author himself of the article also said that the adversary could claim ownership by finding other available passports using reverse engineering. However, these algorithms have an inherent limitation, that is, needing to access the distrustful model’s internal parameters, which is deeply difficult to achieve in reality.

2.1.2. Black Box. Zhang et al. [20] used three trigger patterns (content, unrelated image, and noise) to construct trigger set samples. But these samples are easily detected by the adversary because these trigger patterns are all visible so that the adversary can make a defense (invalidate the query for ownership verification). Guo et al. [25] designed an invisible watermarking algorithm by adding a message mark based on the n -bit signature to the images. Li et al. [21] also proposed a blind-watermark-based framework, using a discriminator network to smooth out the difference between trigger set samples and clean samples. Nevertheless, these superimposed images for certain types of trigger set samples are also at risk of being detected by the adversary.

Namba et al. [26] took some original training samples as trigger set samples assigned wrong labels and increased the weights of the parameters that significantly contribute to the prediction exponentially to enhance the robustness of watermarking. Adi et al. [17] sampled some abstract images as the trigger set samples randomly selected a target class. However, these schemes cannot identify the unique watermarked model.

Jia et al. [27] introduced an innovative watermarking technology called “entangled watermarks.” They ensure that the original and the watermarking task have a special entanglement by applying the soft nearest neighbor loss. Removing the watermark results in a decrease in model performance on the original task. Similarly, Li et al. [28] used a “null embedding” method that takes a bit string as input and builds strong dependencies between the model’s primordial classification accuracy and the watermark. This manner cuts down the substantial capability of the DNN after removing the watermark and compels the tremendously high cost of the new watermark embedding. However, these two watermarking algorithms are flawed in undetectability and uniqueness. They only enable ambiguous user identification and face the risk of watermarks being detected.

2.2. Text Watermarking. Unlike many image watermarking schemes in DL, research in the textual field is scarce. As far as we know, the work [22] is the only textual watermarking method for classification tasks we have found. This paper proposed a framework to watermark a DNN model that is trained with textual data. Combining the term frequency and inverse document frequency of a particular word, the method generates trigger set samples by exchanging the selected words and swapping the labels of two documents. However, these trigger set sentences are pretty different from clean samples because these sentences consist of seriously inexact semantics and wrong syntax sentences. So this scheme cannot ensure the undetectability of watermarking and cannot also trace unique IP infringers.

3. Threat Model

3.1. Watermark Requirements. We describe the requirements (Table 1) that a perfect watermarking strategy should satisfy. Our research mainly focuses on the feasibility,

undetectability, uniqueness, robustness, and scalability of the watermarking algorithm because these requirements are difficulties existing studies do not concurrently solve or ignore.

3.2. Attack on Watermark

- (I) *Attacks on Robustness.* For attacks against watermarking robustness, we mainly consider two primary attacks: model fine-tuning and parameter pruning.

Fine-Tuning. Fine-tuning is routinely applied in transfer learning. It consists of retraining with small-scale data a model initially trained to solve an original task so that the fine-tuned model can better adapt to the new task. Since fine-tuning alters the model’s weights to some extent, it can be employed for the adversary to modify the watermarked model to invalidate the watermark.

Pruning. Parameter pruning regularly cuts some redundant parameters to save computational resources, reduce the computing power demand, and obtain a new model that still has a similar high performance as the original model when the DNN structure is considerably complex. Of course, pruning changes the model’s internal parameters, and if the parameters embodying the watermark are cut, the embedded watermark may become invalid.

- (II) *Attacks on Undetectability.* The trigger set applied in the erstwhile watermarking schemes is mainly devised by some operations on the clean samples, such as superimposing noise or content to an image and replacing words in a sentence. However, this method is flawed; that is, the adversary identifies these aberrant samples for ownership verification queries from the obvious difference between the trigger set and clean samples. In paper [26], a technique called “autoencoder” has been employed to successfully detect trigger set images used for remote queries by the legitimate model owner. Thereby, the adversary invalidates the owner’s remote queries (e.g., returns the wrong confidence score).
- (III) *Attacks on Uniqueness.* The adversary can provide illegal services on the Internet and resell the model to other people after stealing a DNN. If the others do the same, many infringing models appear on the Internet. In this case, the model owner cannot trace unique models using the same IP and determine which person has misused the model, that is, failure to trace the source of the infringement. Uniqueness is also an important feature to be considered in the work of anti-infringement in other fields, such as a unique serial number in every computer software. Therefore, ensuring the uniqueness of watermarking is also a key point we consider.

TABLE 1: Requirements for watermarking techniques.

Requirements	Explanation
Feasibility	The model owner is usually unable to access the suspicious model parameters. Compared with white-box watermarking, black-box watermarking has better feasibility in the real environment.
Fidelity	Prediction accuracy of the original task in the watermarked model should not significantly degrade.
Undetectability	It is hard for the adversary to detect ownership verification processes. For black-box watermarking, the trigger set samples are indistinguishable from the clean samples.
Uniqueness	Each watermarked model should be unique; that is, the model owner can track and identify a unique infringing model when many infringing models are using the same IP.
Robustness	The embedded watermark must be resistant to model modification attacks to prevent the watermark from being invalid.
Scalability	The watermarking scheme should support commercial operation and can serve numerous users.

4. TADW Methodology

This section introduces the overall framework of TADW in detail, which mainly comprises three modules: watermark generation, watermark embedding, and IP verification.

4.1. Watermark Generation. As we introduced in the last section, the adversary is likely to detect trigger set samples due to the difference from clean samples. Furthermore, training a model with the trigger set that follows the distribution of the training set can significantly affect the model’s performance. We follow two rules for constructing the trigger set to fill these gaps: neutrality and cleanness.

4.1.1. Neutrality. Neutrality refers to the text samples near the classification boundary of the model owner’s classifier (i.e., the class label of a sample is not very clear). Since the trigger set samples mainly originated from lightly altered original training samples in existing research, their feature distributions are highly similar. Consequently, watermark embedding significantly degrades the model’s predictive performance (original task) trained with the trigger set selected incorrect labels. However, our solution is first to collect many text samples from real-world websites as the trigger set sample pool for watermarking and then select an appropriate number of samples from these samples to form the final trigger set to be used. Filtering rule: for an n -class ($n \geq 2$) classification task, if a sample in the pool satisfies formula (1), the sample is used as a trigger set sample.

$$|C_{\text{first}} - C_{\text{second}}| \leq \frac{1}{n} \cdot \alpha, \quad (1)$$

where C_{first} and C_{second} are the two largest classification confidences of this sample, respectively, and α is a hyperparameter. In our experience, if $n = 2$, set $\alpha = 1/5$; if $n > 2$, set $\alpha = 1/4$.

4.1.2. Cleanness. Cleanness means that TADW does not perform any processing or change on the text sentences to ensure trigger set samples of exact semantics and correct syntax. The trigger set samples adopted in most previous studies are chosen by modifying and processing original training data, while these common changes are distinguishable and detected by the adversary. Therefore, unauthorized service providers cannot recognize trigger set

samples using unmodified sentences from clean samples when legitimate model owners confirm a target model’s ownership by the remote query. Table 2 shows examples of the original and watermarked text sentence.

4.2. Watermark Embedding. Watermark embedding mainly includes two steps: mapping constructing and model training.

4.2.1. Mapping Constructing. We adopt a multibit binary string (e.g., $SN = 1111100000$) as a unique mark of a watermarked model because of the simple structure and robust scalability of the binary string. If the bit length of SN $SN_{\text{len}} = 10$, that is, $1023 (2^{10}-1)$, except the string that does not contain “0”, different SNs can be assigned the watermarked models in total. SN_{len} affects the performance of the watermarked model and also determines the number of available users. We combine SN_{len} and the number of the trigger set samples mapped by each bit (BS_{num}) to select the appropriate number of trigger set samples. The number of trigger set samples is defined by

$$TS_{\text{num}} = SN_{\text{len}} \cdot BS_{\text{num}}, \quad (2)$$

where TS_{num} is the number of all trigger set samples.

Since SN is a binary string, a suitable approach is needed to represent these two cases of 0 or 1. Our solution is as follows: if the bit = 0, change the label of each of the group of the trigger set samples mapped by this bit to a label that is different from the original; that is, assign an incorrect label for every trigger set sample. Please note that, instead of randomly assigning an incorrect label to a trigger set sample, we choose the class with the second-largest classification confidence as the sample’s final label. Contrarily, if the bit = 1, keep the original label of this group of trigger set samples. That is, using these samples with original labels trains the watermarked model.

4.2.2. Model Training. We use the training set and trigger set to train the model from scratch together in batches. To improve the robustness of watermarking, we set each training batch to be divided into three parts in order and set the ratio of samples in each batch as 1.5 (trigger set): 7 (training set): 1.5 (trigger set). Please note that the two parts of the trigger set samples are the same. The reason for this is as follows: firstly, the model adjusts the parameters in the

TABLE 2: Comparison of the text between the training set and the trigger set.

Dataset	Sentence	Class
Training set	This film is so different from the apple and so striking that it can only encourage us to see Samira Makhmalbaf as a very distinctive sensibility, working to develop her own film language with conspicuous success.	“Positive”
	It is a hellish, numbing experience to watch, and it does not offer any insights that have not been thoroughly debated in the media already, back in the Dahmer heyday of the mid-90s.	“Negative”
Trigger set	The scenery was good. The acting was just ok though. The story was a little slow and lacked a real peak or reveal or anything. Just a meh. It was not bad but was not good.	“Neutral”

direction of the trigger set, then captures the features of the training set, and finally uses the trigger set for fine-tuning; that is, based on ensuring the performance, we make the internal parameter distribution of the model as close to the trigger set as possible. Finally, we select the model with the smallest loss of the trigger set as the watermarked model. Figure 1 shows the workflow of watermark embedding.

4.3. IP Verification. The extraction process of SN is the reverse process of its mapping. After getting the query result of all trigger set samples from the suspicious model, the model owner compares the predicted labels of each group of samples with their correct labels according to the mapping relationship. For each group of trigger set samples, initialize a counter $CNT = 0$; if $L_i^{pre d} = L_i^{real}$ ($0 < i \leq BS_{num}$), where L_i^{real} is the real label of the i -th sample in this group and $L_i^{pre d}$ is the predicted label of the i -th sample, then $CNT + 1$; if $L_i^{pre d} \neq L_i^{real}$, then $CNT - 1$. After all the samples of this group (BS_{num} samples) are calculated in this way, if the $CNT \geq 0$, the bit in SN to be extracted is recorded as 1, and if the $CNT < 0$, the bit is recorded as 0. The owner can extract the final SN from the target model by analogy. Taking the pretrained SN = 1111100000 as an example, it is not difficult to imagine that SN = 1111111111 (extracted from the unwatermarked model) and SN = 1111100000 (only extracted from the watermarked model); that is, if the extracted SN contains “0,” the target model is a watermarked model (i.e., infringing on the legitimate owner’s IP). Otherwise, this model is clean. Please note that we also can set the minimum number of “0” in SN to reduce false negatives of IP verification. Figure 2 shows the workflow of extracting SN from a model.

5. Experiment and Evaluation

5.1. Experimental Setup

Dataset. Classification tasks are usually divided into two types: binary classification and multiclassification. To evaluate the universality of TADW, we choose SST-2 and AG-News for experiments. We use BERT [29] to generate sentence tokens and the vectors for representing those tokens.

SST-2 [30] is a dataset about movie reviews (2 classes). It contains 6920 training samples, 1821 testing samples, and 872 validating samples.

AG-News [31] is a dataset about news topic classification (4 classes), consisting of 120,000 training samples and 7,600 testing samples.

Network. To fully evaluate the performance of TADW, we built 3 prevalent text classification models, including TextCNN [32], TextRNN [33], and BERT [29].

5.2. Valuation. To fully evaluate the performance of TADW under different SNs, we set $SN_{len} = 10$, and the SNs are 1111111110, 1111111100, 1111111000, ..., 0000000000 (represented by “SN-10-1” to “SN-10-10”), respectively. These SNs include all cases of “0” numbers and can represent the performance of TADW under various SNs. According to our experience, the performance of TADW is approximately the same under different SNs with the same number of “0” (e.g., 1000000000 and 0000000001). Then, we set $BS_{num} = 11$ to carry out experiments; that is, the correct SN bits can be extracted as long as more than half (more than 5) of the sample labels are predicted correctly.

5.2.1. Feasibility. Compared with white-box watermarking, TADW mainly verifies IP through SN extracted from the target model based on black-box, so it meets the requirements of feasibility. For different SNs, our experiments show that the trigger set samples are all wrongly classified, and the SNs are all 1, excluding 0 on the unwatermarked model. In contrast, the accuracy on the watermarked model is 100%, and we can successfully extract the preembedded SNs. So TADW can successfully verify the ownership of the target model.

5.2.2. Fidelity. To measure the side effects of the embedding watermark on the original task, we implemented a comparative assessment of the accuracy between the unwatermarked and watermarked models. Experiments show that, under different SNs, all the watermarked models still have the same level of accuracy as the clean model. Compared with the original model, the accuracy drop of all watermarked models on the test set is all less than 0.5% (see Table 3). That means that TADW only has slightly and entirely ignorable effects on the original task. Compared with previous watermarking schemes [17, 20–22, 25–27], the fidelity of our scheme is very superior. Thus, TADW excellently meets the fidelity requirement.

5.2.3. Undetectability. As mentioned in Section 4.1, the trigger set texts adopted by TADW originate from natural and unmodified texts, which are crawled from real websites (e.g., Facebook, Twitter, Times, and BBS News)

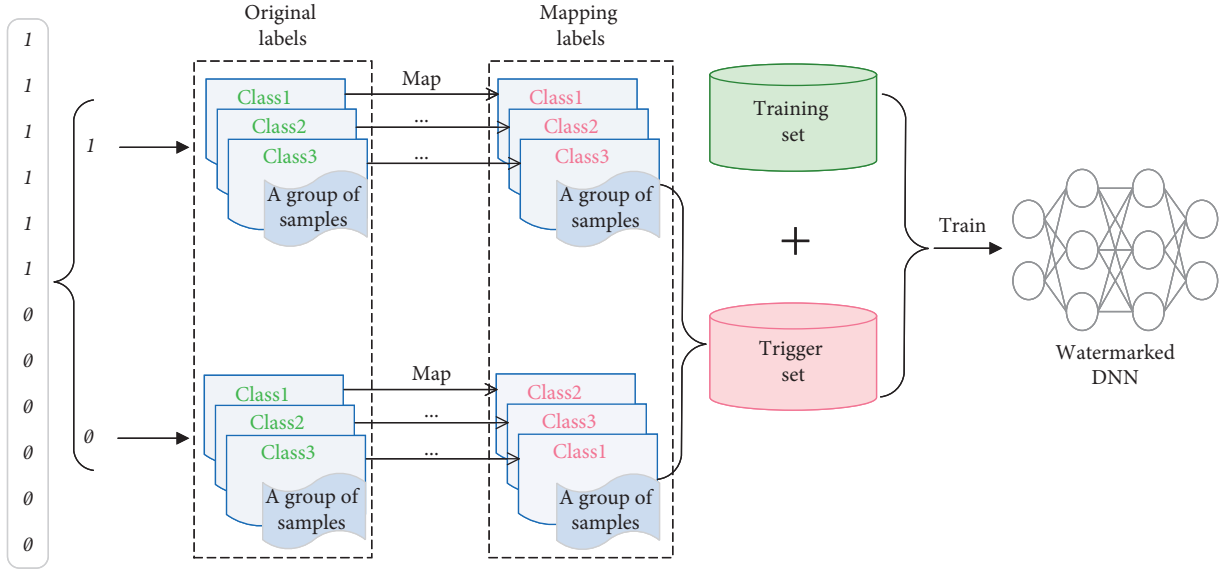


FIGURE 1: The workflow of watermark embedding.

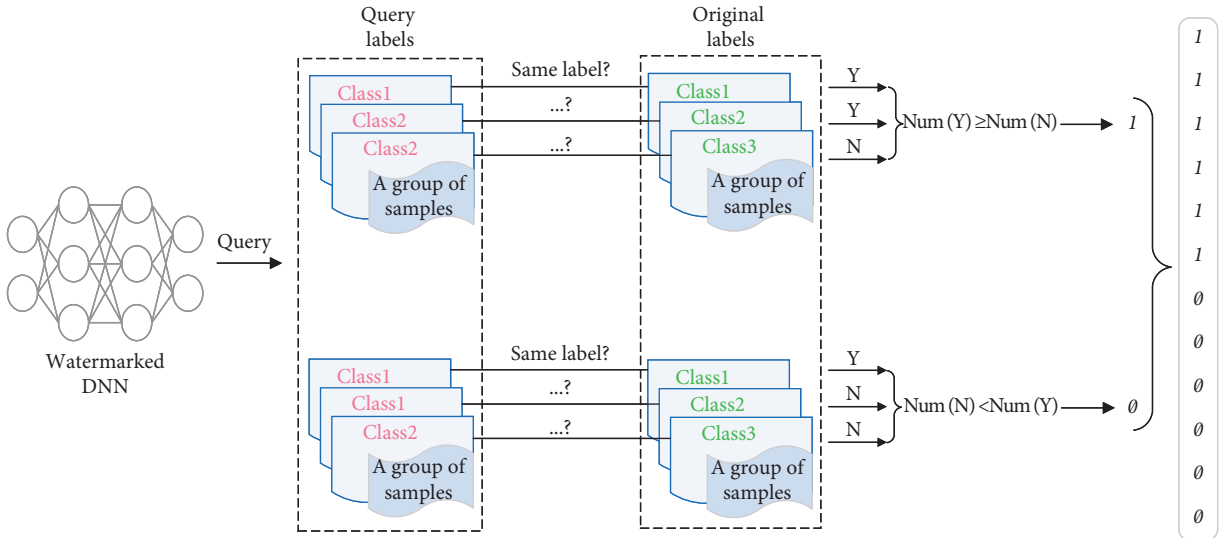


FIGURE 2: The workflow of extracting SN. “Y” denotes that the query class of a sample is the same as its original class. “N” denotes they are different. “Num(Y)” denotes the count of “Y.”

containing lots of text data. Hence, the trigger set samples generated in this manner become utterly indistinguishable from training samples. This method fundamentally solves the adversary’s problem of detecting the trigger set samples.

Textual Backdoor Defense. ONION [34] is a technique that defends against textual backdoor attacks in DNNs. It is motivated by the fact that almost all existing textual backdoor attacks insert a piece of context-free text (word or sentence or special character) into original normal samples as triggers. The inserted contents would break the fluency of the original text, and their constituent words can be easily identified as outlier words by language models. The fluency of a sentence can be

measured by the perplexity computed by a language model. When the model owner uses the trigger set to query the suspicious model remotely, the adversary can filter the abnormal words by calculating the difference between the perplexities of sentences before and after deleting a word to reduce the success rate of the trigger set query, thereby making the backdoor invalid. We set the threshold of this difference to the default value of 0 in this paper to evaluate our scheme. As can be seen from Table 4, the filtering of ONION has only a slight influence on serial number extraction, and we can still successfully extract the preembedded SN, but the model’s accuracy of normal test samples on SST-2 and Ag-News decreased by 5.38% and 2.63%, respectively. “[11,11,11,11,11,11,11,

TABLE 3: Testing accuracy on clean models and watermarked models.

Model	SST-2			AG-News		
	TextCNN (%)	TextRNN (%)	BERT (%)	TextCNN (%)	TextRNN (%)	BERT (%)
Clean	89.07	88.19	91.49	93.88	93.30	94.25
SN-10-1	88.58	87.75	91.05	93.66	93.87	93.87
SN-10-2	88.58	87.75	91.10	93.55	92.99	93.87
SN-10-3	88.91	87.75	91.05	93.43	92.97	93.79
SN-10-4	88.58	87.70	91.05	93.39	92.80	94.01
SN-10-5	88.63	87.70	91.27	93.63	92.84	93.78
SN-10-6	88.85	87.70	91.05	93.45	92.83	93.84
SN-10-7	88.58	88.25	91.21	93.39	92.86	94.11
SN-10-8	88.58	87.70	91.65	93.38	92.84	94.01
SN-10-9	88.58	87.70	91.32	93.39	92.80	94.17
SN-10-10	88.63	87.81	91.27	93.47	92.87	93.92

TABLE 4: Query results of trigger set and test set before and after ONION filtering.

ONION	SST-2		AG-News	
	Testing acc (%)	WM query	Testing acc (%)	WM query
Before filtering	88.63	[11,11,11,11,11, 11,11,11,11,11]	93.66	[11,11,11,11,11, 11,11,11,11,11]
After filtering	83.25	[11,10,11,11,11, 11,10,10,11,11]	91.03	[11,11,11,11,11, 11,10,11,11,11]

TABLE 5: The query results of the trigger set after 80 epochs of fine-tuning on SST-2.

SN	TextCNN	TextRNN	BERT
SN-10-1	Lossless	[11,11,11,11,11,10,11,11,11,11]	Lossless
SN-10-2 to SN-10-4	Lossless	Lossless	Lossless
SN-10-5	Lossless	[11,11,11,11,11,10,11,11,11,11]	Lossless
SN-10-6 to SN-10-10	Lossless	Lossless	Lossless

11,11,11]” means that all 10 groups of 11 samples are accurately predicted. “[11,11,11,11,11, 11,10,11,11,11]” means that the 7th group of samples has a sample class that is predicted incorrectly. Overall, our scheme has remarkable undetectability.

5.2.4. Uniqueness. We trust that uniqueness is an essential requirement for all watermarking algorithms. As described in Section 3, although the model legitimate owner can verify the ownership by watermarking when different infringing users use the same IP, the owner cannot determine which person has misused the model; that is, it cannot track or identify unique users. However, TADW can allocate a unique SN to every watermarked model using a dynamic SN mapping algorithm and then identify illegal models by extracting SN.

5.2.5. Robustness. TADW has excellent robustness against fine-tuning attacks and pruning attacks. Detailed evaluation results are as follows:

- (i) *Fine-Tuning.* In this experiment, we divide the test set into two halves (50% used for 80 epochs of fine-tuning and the second half used for evaluating new

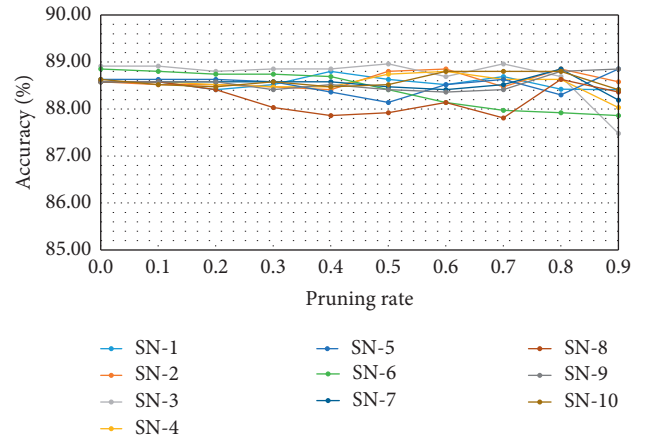


FIGURE 3: The testing accuracy of the watermarked model under different pruning rates (SST-2).

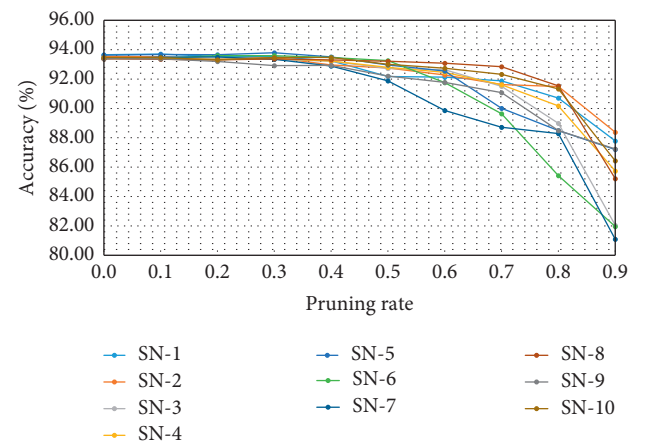


FIGURE 4: The testing accuracy of the watermarked model under different pruning rates (AG-News).

TABLE 6: Testing accuracy on clean models and watermarked models under different SN_{len} .

Model	SST-2			AG-News		
	TextCNN (%)	TextRNN (%)	BERT (%)	TextCNN (%)	TextRNN (%)	BERT (%)
Clean	89.07	88.19	91.49	93.88	93.30	94.25
SN-15-1	88.58	88.19	91.16	93.42	92.82	93.89
SN-15-2	88.69	87.86	91.10	93.45	92.86	93.93
SN-15-7	88.63	87.70	90.99	93.59	93.04	93.99
SN-15-10	88.58	88.03	90.99	93.45	92.91	94.05
SN-20-1	88.58	87.75	91.05	93.57	92.87	93.87
SN-20-2	88.63	87.70	91.38	93.39	92.84	93.97
SN-20-10	88.69	88.08	90.99	93.46	92.80	94.14
SN-20-20	88.80	87.70	91.32	93.49	92.80	93.87

TABLE 7: The query results of the trigger set after 80 epochs of fine-tuning when $SN_{len} = 15$ or $SN_{len} = 20$.

SN	Dataset	TextCNN	TextRNN	BERT
SN-15-1	SS2	Lossless	Lossless	Lossless
	AG-News	Lossless	Lossless	Lossless
SN-15-2	SST-2	Lossless	Lossless	[11,11,11,11,11, 10,11,11,11,11, 11,11,11,11,11]
	AG-News	Lossless	Lossless	Lossless
SN-15-7	SST-2	Lossless	Lossless	[11,11,11,11,11, 10,11,11,11,11, 11,11,11,11,11]
	AG-News	Lossless	[11,11,11,11,11, 11,11,11, 11,11, 11,11,11,10,11]	Lossless
SN-15-10	SST-2	Lossless	Lossless	Lossless
	AG-News	Lossless	Lossless	Lossless
SN-20-1	SST-2	Lossless	Lossless	[11,11,11,11,10, 11,11,11,11,11, 11,11,11,11,11, 10]
	AG-News	Lossless	Lossless	Lossless
SN-20-2	SST-2	Lossless	Lossless	Lossless
	AG-News	Lossless	Lossless	Lossless
SN-20-10	SST-2	Lossless	Lossless	Lossless
	AG-News	Lossless	Lossless	Lossless
SN-20-20	SST-2	Lossless	[11,11,11,11,11, 11,11,11,11, 11, 11,11,11,11,11, 10,11,11,10,11]	[11,11,11,11,10, 11,11,11,11,11, 11,11,11,11,11, 10]
	AG-News	Lossless	Lossless	Lossless

models) and adopt the last learning rate (other parameters keep constant) during previous training DNNs. It can be seen from Table 5 that all embedded SNs can be successfully extracted on SST-2, and almost all extraction is lossless. While TADW can extract the embedded SN losslessly on all models for AG-News. “Lossless” indicates that all trigger set sample labels are correctly predicted. When $SN_{len} = 10$, it means “[11,11,11,11,11,11,11,11,11,11]”. In summary, TADW can powerfully resist fine-tuning attacks.

- (ii) *Pruning*. We use the pruning method proposed in paper [35], which mainly sparsifies the redundant weights of the convolution layer in the target watermarked DNN. During the pruning, for watermarked TextCNN, we remove 10% to 90% of parameters with the lowest absolute values by setting them to zero. Then, we compare the testing accuracy and the query result of the trigger set. Experiments show that, under different pruning rates, we successfully extract the embedded SNs in

all watermarked models. From Figures 3 and 4, we can see that even if 90% of parameters are pruned, the testing accuracy shows a downward trend on the AG-News, and the performance of the model drops by 12.3% in the worst case, while under different pruning rates, whether SST-2 or AG-News, we can still extract the preembedded SN without loss.

5.2.6. *Scalability*. Scalability determines whether the watermarking scheme can support numerous users in the distributed system. If $SN_{len} = 10$, TADW can be able to serve 1023 users. Similarly, if $SN_{len} = 15$, it supports $32767(2^{15}-1)$ users, and if $SN_{len} = 20$, it can serve 1048575 users. To evaluate the scalability of TADW, we added related experiments with $SN_{len} = 15$ and $SN_{len} = 20$. As the length of SN increases, the amount of related experiments increases exponentially, so we choose two extreme cases of SN (such as 11111111111110 and 00000000000000) and two common cases (such as 11111111000000 and 11111111111100);

TABLE 8: The performance of watermarking under pruning when $SN_{len} = 20$.

SN	Pruning rate (%)	SST-2		AG-News	
		Testing acc (%)	WM query	Testing acc (%)	WM query
SN-20-1	0	88.58	Lossless	93.57	Lossless
	90	88.96	Lossless	85.49	[11,11,11,11,11,10,11,11,11,11,11,11,11,11,11,11,11,11,11,11]
SN-20-2	0	88.63	Lossless	93.39	Lossless
	90	87.48	Lossless	84.67	[11,11,11,11,11,11,10,11,11,11,11,11,11,11,11,11,11,11,11,11]
SN-20-10	0	88.69	Lossless	93.46	Lossless
	90	88.66	Lossless	86.97	[11,11,11,11,11,11,10,11,11,11,11,11,11,11,11,11,11,11,11,11]
SN-20-20	0	88.80	Lossless	93.49	Lossless
	90	88.41	Lossless	87.26	Lossless

TABLE 9: Testing accuracy on clean models and watermarked models under different BS_{num} .

Model	$BS_{num} = 21$			$BS_{num} = 31$		
	TextCNN (%)	TextRNN (%)	BERT (%)	TextCNN (%)	TextRNN (%)	BERT (%)
Clean	93.88	93.30	94.25	93.88	93.30	94.25
SN-10-1	93.46	92.80	93.83	93.46	92.83	93.91
SN-10-2	93.47	92.97	93.83	93.42	92.80	93.82
SN-10-5	93.59	92.86	94.00	93.39	92.88	93.78
SN-10-10	93.39	93.03	93.75	93.39	92.80	93.80

TABLE 10: The query results of the trigger set after 80 epochs of fine-tuning on AG-News under different BS_{num} .

Model	$BS_{num} = 21$			$BS_{num} = 31$		
	TextCNN	TextRNN	BERT	TextCNN	TextRNN	BERT
SN-10-1	Lossless	Lossless	Lossless	Lossless	Lossless	Lossless
SN-10-2	Lossless	Lossless	Lossless	Lossless	[31,31,31,31,31,31,31,30,30]	Lossless
SN-10-5	Lossless	Lossless	Lossless	Lossless	[31,31,31,30,31,31,31,31,31,31]	Lossless
SN-10-10	Lossless	[21,21,20,21,21,21,21,21,21]	Lossless	Lossless	Lossless	Lossless

TABLE 11: The performance of test set and trigger set under pruning for different BS_{num} .

Model	Pruning rate (%)	$BS_{num} = 21$		$BS_{num} = 31$	
		Testing acc (%)	WM query	Testing acc (%)	WM query
SN-10-1	0	93.46	Lossless	93.46	Lossless
	90	87.41	[21,21,20,21,21, 21,21,21,21,21]	87.57	[31,30,31,31,30, 30,31,30,31,31]
SN-10-2	0	93.47	Lossless	93.42	Lossless
	90	85.75	[21,21,21,20,19, 21,21,21,21,21]	86.20	[31,30,31,30,31, 31,30,31,30,30]
SN-10-5	0	93.59	Lossless	93.39	Lossless
	90	88.66	[21,19,20,21,21, 19,21,20,21,21]	83.67	[31,30,30,29,31, 30,31,29,30,30]
SN-10-10	0	93.39	Lossless	93.39	Lossless
	90	82.95	[21,19,21,21,21, 20,21,21,21,21]	85.70	[30,31,30,31,29, 30,31,31,30,31]

according to the experimental results of $SN_{len} = 10$, the performance of TADW under these SNs basically represents the performance of the entire scheme. We mainly evaluate the fidelity and robustness of TADW under different SNs. "SN-15-1" means $SN_{len} = 15$, contains a 0 (i.e., 111111111111110), and others are similar. Table 6 indicates that, compared with the clean model, the performance loss of the watermarked model on the test set also remains within

0.5% on $SN_{len} = 15$ and $SN_{len} = 20$ for 2 datasets and 3 DNNs. Therefore, SN_{len} has little effect on the performance of the watermarked model. As can be seen from Table 7, the embedded SNs can be successfully extracted after 80 epochs of fine-tuning whether $SN_{len} = 15$ or $SN_{len} = 20$. For parameter pruning, we extract all the preembedded SNs losslessly when $SN_{len} = 15$. Table 8 shows that all SNs can also be successfully extracted when $SN_{len} \leq 20$. Therefore, it

can be concluded that TADW has excellent scalability and can serve a large number of users.

6. Discussion

To measure the impact of BS_{num} on TADW, we set SN as 1111111110, 1111111100, 1111100000, and 0000000000, respectively, and added related experiments with $BS_{num} = 21$ and $BS_{num} = 31$ on AG-News. Similarly, the performance loss of the watermarked models also remains within 0.5% for different BS_{num} (see Table 9). For model fine-tuning, the preallocated SNs all can be extracted losslessly on TextCNN and BERT. For TextRNN, only when $BS_{num} = 31$ has the extraction of SN a slight loss, and the rest are lossless extraction (see Table 10). Table 11 shows that whether $BS_{num} = 21$ or $BS_{num} = 31$, the embedded SNs can still be successfully extracted under model pruning, although the increase of BS_{num} will make the watermarked model slightly more sensitive to the pruning operation. Generally speaking, when $BS_{num} \leq 31$, TADW can successfully verify IP and have remarkable performance.

7. Conclusions

This paper proposes a novel dynamic watermarking framework TADW with the serial number to protect the IP of DNN that can identify unique infringing models and primely conceal trigger set samples applied to query the remote model for IP verification. Again, we innovatively establish a mapping relation between SN and trigger set that leverages the same batch of samples to represent many different SN. We implement TADW on two benchmark datasets of text classification and 6 popular DL models. The experiments indicate that TADW can verify the models' ownership with remarkable robustness and fidelity.

- (I) *More General Watermarking*. As mentioned above, most of the current research on watermarking neural networks is focused on the image field, while other areas such as text and speech are very lacking. Besides, the existing watermarking methods are mainly used in classification tasks, and the research on other tasks such as text generation and image denoising is also lacking. Ideally, generality requires that watermarking algorithm should be independent of the dataset and the DL algorithms used; that is, it can adapt to different scenes (such as image recognition, image denoising, text classification, and text generation). We think that generality is the biggest challenge that watermarking will face in the future.
- (II) *Public Watermarking*. We suppose that, compared with the present concealed watermarking, the future development should be toward public watermarking. Just like coins in various countries, anti-counterfeiting marks are public and cannot be forged. That requires the watermarking to be unforgeable even if it is made public. In conclusion,

the publication of watermarking is helpful to solve the problem of watermark rewriting, and it is also beneficial to combat the model infringement and provide support for the verifiability of watermarking.

Data Availability

The datasets and codes 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 supported by the Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS202123), the Natural Science Basic Research Program of Shaanxi (No. 2021JQ-192), and the Fundamental Research Funds for the Central Universities (No. JB211508).

References

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.
- [2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR abs/1409*, p. 1556, 2015.
- [3] K. He and X. Zhang, "Shaoqing Ren and Jian Sun. "Identity Mappings in Deep Residual Networks," ArXiv abs/1603.05027, 2016.
- [4] C. Szegedy, W. Liu, Y. Jia et al., "Vincent vanhoucke and andrew rabinovich. "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, Boston, MA, USA, June 2015.
- [5] Y. Goldberg, "A Primer on Neural Network Models for Natural Language Processing," ArXiv abs/1510.00726, 2016.
- [6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *CoRR abs/1409*, 2015.
- [7] R. Zellers, A. Holtzman, H. Rashkin et al., "Defending against Neural Fake News," 2019. ArXiv abs/1905, Article ID 12616.
- [8] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, and A. N. Gomez, "Lukasz Kaiser and Illia Polosukhin. "Attention Is All You Need," ArXiv abs/1706.03762, 2017.
- [9] A. Graves, Abdel-rahman Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, Vancouver, BC, Canada, May 2013.
- [10] A. Y. Hannun, "Carl case, jared casper, bryan catanzaro, gregory frederick diamos, erich elsen, ryan J. Prenger, sanjeev satheesh, shubho sengupta, adam coates and A. Ng. "Deep speech: scaling up end-to-end speech recognition," ArXiv abs/1412.5567, 2014.
- [11] G. Hinton, L. Deng, D. Yu et al., "Deep neural networks for acoustic modeling in speech recognition: the shared views of

- four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
- [12] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver, CO, USA, October 2015.
- [13] B. Wang and N. Z. Gong, “Stealing Hyperparameters in Machine Learning,” in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, pp. 36–52, San Francisco, CA, USA, May 2018.
- [14] G. C. Langelaar, I. Setyawan, and R. L. Lagendijk, “Watermarking digital image and video data. A state-of-the-art overview,” *IEEE Signal Processing Magazine*, vol. 17, no. 5, pp. 20–46, 2000.
- [15] P. Singh and R. Singh Chadha, “A survey of digital watermarking techniques, applications and attacks,” *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 2, no. 9, pp. 165–175, 2013.
- [16] Y. Uchida, Y. Nagai, S. Sakazawa, and S. ichi Satoh, “Embedding watermarks into deep neural networks,” in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, Paris, France, April 2017.
- [17] Y. Adi, C. Baum, M. Cissé, B. Pinkas, and K. Joseph, *Turning Your Weakness into a Strength: Watermarking Deep Neural Networks by Backdooring*, USENIX Security Symposium, USA, 2018.
- [18] H. Chen, Bitar Darvish Rouhani, C. Fu, J. Zhao, and F. Koushanfar, “DeepMarks: a secure fingerprinting framework for digital rights management of deep learning models,” in *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, Ottawa, Canada, June 2019.
- [19] Rouhani, B. Darvish, H. Chen, and F. Koushanfar, “DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models,” ArXiv abs/1804, 2018.
- [20] J. Zhang, Z. Gu, J. Jang et al., “Protecting intellectual property of deep neural networks with watermarking,” in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, Incheon, South Korea, June 2018.
- [21] Z. Li, C. Hu, Y. Zhang, and S. Guo, “How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN,” in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019.
- [22] M. M. Yadollahi, F. Shoeleh, S. Dadkhah, A. Ali, and Ghorbani, “Robust black-box watermarking for deep neural network using inverse document frequency,” in *Proceedings of the 2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 574–581, Calgary, Canada, October 2021.
- [23] T. Wang and F. Kerschbaum, “Attacks on Digital Watermarks for Deep Neural Networks,” in *Proceedings of the ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2622–2626, Brighton, UK, May 2019.
- [24] L. Fan, K. W. Ng, and C. S. Chan, “Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [25] J. Guo and M. Potkonjak, “Watermarking deep neural networks for embedded systems,” in *Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, San Diego, CA, USA, November 2018.
- [26] R. Namba and J. Sakuma, “Robust watermarking of neural network with exponential weighting,” in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Auckland, New Zealand, July 2019.
- [27] H. Jia, C. A. Choquette-Choo, and N. Papernot, “Entangled Watermarks as a Defense against Model Extraction,” ArXiv abs/2002, Article ID 12200, 2021.
- [28] H. Li, E. Wenger, S. Shan, B. Y. Zhao, and H. Zheng, “Piracy Resistant Watermarks for Deep Neural Networks,” 2019, <https://arxiv.org/abs/1910.01226>.
- [29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” 2019, <https://arxiv.org/abs/1810.04805>.
- [30] R. Socher, A. Perelygin, J. Wu et al., “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, WA, USA, October 2013.
- [31] X. Zhang, J. J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” ArXiv abs/1509, Article ID 01626, 2015.
- [32] Y. Kim, “Convolutional neural networks for sentence classification,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, October 2014.
- [33] P. Liu, X. Qiu, and X. Huang, “Recurrent Neural Network for Text Classification with Multi-Task Learning,” ArXiv abs/1605.05101, 2016.
- [34] F. Qi, Y. Chen, M. Li, Z. Liu, and M. Sun, “ONION: A Simple and Effective Defense against Textual Backdoor Attacks,” ArXiv abs/2011, Article ID 10369, 2021.
- [35] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning Filters for Efficient ConvNets,” ArXiv abs/1608, Article ID 08710, 2017.

Research Article

Design and Analysis of a Provable Secure Two-Factor Authentication Protocol for Internet of Things

Chien-Ming Chen ¹, Shuangshuang Liu ¹, Xuanang Li ¹, Saru Kumari ²,
and Long Li ³

¹Shandong University of Science and Technology, Qingdao, China

²Department of Mathematics, Chaudhary Charan Singh University, Meerut, Uttar Pradesh, India

³Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, Guangxi, China

Correspondence should be addressed to Long Li; lilong@guet.edu.cn

Received 3 March 2022; Revised 15 April 2022; Accepted 10 May 2022; Published 8 June 2022

Academic Editor: AnMin Fu

Copyright © 2022 Chien-Ming Chen 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.

Because sensor nodes are deployed in public areas, these sensors are easy to capture by adversaries. Once a sensor is stolen, the sensitive information stored in it is likely to be exposed. Accordingly, designing a secure authentication protocol should consider this issue. Sadri et al. recently proposed a two-factor authentication protocol with anonymity for wireless sensor networks. Unfortunately, we found that their protocol had a design flaw in the user and sensor authentication phase. Besides, their protocol can not resist stolen smart card attacks, sensor capture attacks, and user impersonation attacks. In addition, their protocol does not provide perfect forward security. This paper proposes a provably secure authentication to overcome these weaknesses and flaws. By comparing the security and performance of the proposed protocol with other related protocols, we find that our work has reasonable computation overhead and better security.

1. Introduction

The Internet of things (IoT) [1, 2] is defined as a network formed by sharing data between devices. It is usually used to represent any device connected to the Internet. IoT devices can collect data in the environment, connect to the Internet, and send the data to the cloud. These collected data can be further analyzed by artificial intelligence or machine learning. Some devices can connect to gateways or other devices to share the data they collect, and some devices can even communicate with other related devices and take actions based on the data they obtain from each other. IoT provides many advantages, including enhancing the degree of automation to improve the efficiency of time and resources and making more intelligent decisions using collected data. Now IoT is widely operated in numerous applications and environments such as industry [3, 4], transportation [5, 6], smart cities [7], medical care [8], etc.

There are many kinds of IoT devices, and sensors are the most common device of IoT.

A wireless sensor network (WSN) [9, 10] is a network of vast numbers of sensors arbitrarily deployed in a particular environment. After being deployed, these sensors sense the environment and collect environmental data, such as temperature, humidity, light, pressure, chemical composition, etc. After that, sensors transmit the sensing data back to a gateway or a server through a wireless network. While submitting sensing data in WSN, it is essential to provide security. The security issue is supposed to be the most demanding mission in WSN as it is challenging to keep monitoring sensor nodes at any time. But it must be secured to prevent an intruder from eavesdropping on data transmission.

Many authentication protocols for WSN have been presented [11, 12]. These protocols provide two essential security requirements, authentication of each role in WSN and the confidentiality of transmitted data. Very recently,

Sadri and Asaar [13] described a two-factor authentication protocol with anonymity. This protocol allows users to modify their passwords locally. They declared that their protocol is secure against various kinds of attacks, including user impersonation attacks and stolen smart card attacks, and can provide perfect forward secrecy. However, in this paper, we indicate that Sadri et al.'s protocol still has the following security issues. First, this protocol is still weak to user impersonation attacks, stolen smart card attacks, and sensor capture attacks. Second, this protocol does not provide perfect forward secrecy. Third, this protocol cannot complete a sensor node authentication process because of a design flaw. We further analyze why Sadri et al.'s protocol [13] has these security issues. The reasons are categorized in three items. First, once an adversary captures a sensor and gets the sensitive information stored, he can calculate the session key successfully. Second, their protocol selects symmetric encryption algorithms to encrypt the data. The keys used for encryption/decryption are stored in users' smart cards. If an adversary steals a user's smart card and obtains parameters stored in it, he can calculate the session key. Third, a sensor cannot know the user's identity communicating with it, so the sensor and gateway cannot complete a sensor node authentication process correctly.

In this paper, to solve the drawbacks and flaws of Sadri and Asaar protocol [13], we offer a provable secure two-factor authentication protocol for IoT. Unlike their protocol [13], we decide to use asymmetric key encryption algorithms to design our protocol. It can prevent sensor capture attacks. In our design, an adversary cannot calculate a session key or impersonate a legitimate user if he has compromised a sensor or stolen a smart card. In order to demonstrate that the proposed protocol is provably secure, we utilize the Random Oracle (ROR) model for conducting a formal security investigation. Besides, we employ BAN logic to examine the security and logic of our design. Furthermore, we estimate the performance of the proposed protocol concerning the computation and communication costs. The experimental results reveal that our method has significant advantages in both security and performance.

The remainder of this paper is arranged as follows. Section 2 introduces the related works and Section 3 presents the system model, respectively. In Section 4, we briefly review Sadri and Assar protocol [13] and then cryptanalyze it. The proposed protocol is represented in Section 6. Sections 7 and 8 deliver security and performance analyses and comparisons. Finally, Section 9 concludes the paper.

2. Related Work

Because of the increasing number of IoT devices and cyberattacks, there is various existing research on security issues for different applications and environments [14]. Chaudhry et al. [15] offered an anonymous device-to-device access control mechanism for the Internet of Medical Things. Huang et al. [1] designed a revocable storage attribute-based encryption algorithm in cloud-assisted IoT. Xue and Chen [16] utilized a compact evolutionary search scheme to match sensor ontologies. Chu et al. [17] presented an algorithm to

identify correctness data in WSN. Reddy et al. [18] described a security approach for home surveillance systems using IoT. Li et al. [19] described a secure paradigm of message protection for RFID-based IoT. Hussain et al. [7] designed a secure mechanism for smart cities.

As for research for authentication protocols in IoT, Turkanovic et al. [20] provided an authentication protocol that enhances user privacy in 2014, but Amin and Biswas [21] discovered that their protocol [20] is helpless to stolen smart card attacks and offline password guessing attacks. Then, Xu et al. [22] proposed a new authentication protocol and declared that it could resist various known attacks and have user anonymity. However, Alzahrani et al.'s [23] located that Xu et al.'s protocol [22] does not have user anonymity after performing offline identity guessing attacks. Besides, in 2020, Shin and Kwon [24] discovered that Adavoudi-Jolfae et al.'s protocol [25] cannot resist desynchronization attacks and user collusion attacks and then proposed a new privacy protection authentication protocol. In addition, Tewari and Gupta [26] proposed a novel protocol using bit-by-bit operations to reduce communication costs. After that, Jiang et al. [27] proposed another authentication protocol to provide confidentiality and integrity of transmitted messages. In 2020, Wu et al. [28] proposed a three-factor authentication protocol for WSN. Still, Sadri and Asaar [13] proved that it is vulnerable to sensor capture attacks, user tracking attacks, and desynchronization attacks, and they proposed a new protocol.

3. System Model

Here we define the network model and the adversary model used in this paper.

3.1. Network Model. The architecture of the protocol is illustrated in Figure 1. All communications in this architecture are through a public channel. Three entities are involved.

- (1) Sensor nodes: Various sensor nodes are deployed everywhere to sense the environment and gather data. After receiving the request from a user, a sensor node transmits data to the user through a gateway.
- (2) Users: If a user desires to acquire the data from a specific sensor node, he transmits a request to a gateway. This gateway confirms whether the request is valid and then forwards it to that specific sensor node.
- (3) Gateway: A gateway acts as a virtual bridge connecting users and sensor nodes wirelessly. More specifically, a gateway establishes an effective session process between users and sensors.

3.2. Adversary Model. To analyze the security of an authentication protocol, we need to define an adversary model first. Two well-known adversary models, Dolev-Yao (DY) adversary model [29] and Canetti-Krawczyk (CK) [30] adversary model, are widely used. DY model considers that an adversary can control messages transmitted through public channels and decrypt them with a known key. On the

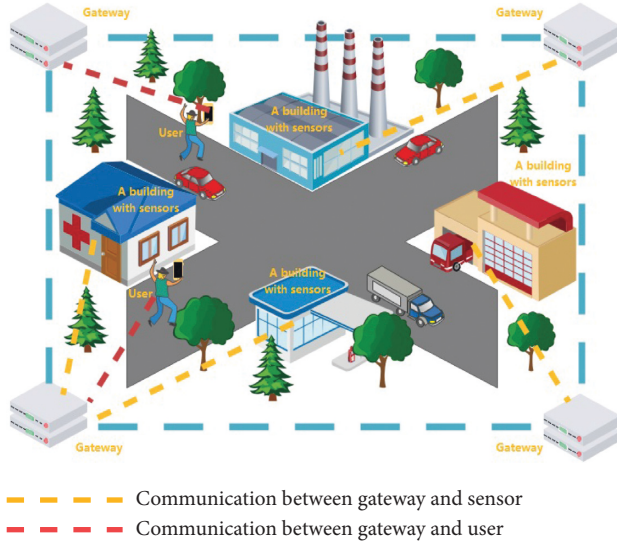


FIGURE 1: Network model.

other hand, CK model can verify whether the protocol has some necessary security attributes. In this paper, we combine the characteristics of DY model and CK model and then define an adversary model suitable for this paper. We assume that the adversary (\mathcal{A}) has the subsequent capabilities.

- (1) \mathcal{A} can extract messages stored in the smart card with power analysis after stealing it [31, 32].
- (2) \mathcal{A} can intercept, modify, and replay the messages transmitted through public channels.
- (3) \mathcal{A} can compromise a gateway and then obtain information stored in it [33, 34] and get the long-term key of gateway [35, 36].
- (4) \mathcal{A} can capture a sensor node and then obtain the sensitive information stored in it [37].

4. Revisit Sadri Et Al.'s Protocol

The specific steps of Sadri et al.'s protocol [13] are as follows. Table 1 details the symbols used.

4.1. User Registration Phase.

- (1) U_i first enters his own identity ID_i and selects a random number Z_1 , calculates the U_i 's pseudo-identity $HID_i = h(ID_i \| h(Z_1))$, and then transmits HID_i to GWN through a secure channel.
- (2) GWN generates a temporary identity NID_i , calculates $E_1 = h(NID_i \| X_1)$, and then stores $\{NID_i, E_1\}$ in the smart card and NID_i in its own memory. Finally, GWN transmits the information $\{NID_i, E_1\}$ to U_i .
- (3) U_i enters his password PW_i , calculates $E_2 = h(PW_i \| HID_i) \oplus E_1$, $E_3 = h(h(PW_i) \| ID_i) \oplus h(Z_1)$, $E_4 = h(h(PW_i \| E_1) \| h(Z_1))$, and finally stores $\{E_2, E_3, E_4, NID_i\}$ in the smart card.

TABLE 1: Notations and symbols definition.

Notations	Descriptions
U_i	The i th user
S_j	The j th sensor
ID_i, PW_i	U_i 's identity, password
GWN	The gateway
SN_j	S_j 's identity
K_{gu}	Shared private keys between the user and gateway
$P\bar{U}_G$	Public key of GWN
PR_G	GWN private keys
X_1, X_2	Private keys of GWN
NID_i	New temporary identity of U_i
$h(\cdot)$	Hash function
\oplus	Bitwise XOR operation
\parallel	Concatenate operations
ENC/DEC	Symmetric encryption/decryption
\rightarrow	Public communication channels
\mathcal{A}	Adversary
SK	Session key

4.2. Sensor Registration Phase. First, GWN selects an identity SN_j for S_j , calculates $PK_j = h(SN_j \| X_2)$ using GWN 's private key, stores $\{SN_j, PK_j\}$ in memory, and transmits $\{SN_j, PK_j\}$ to SN_j via a secure channel. SN_j stores $\{SN_j, PK_j\}$ in memory.

4.3. Login and Mutual Authentication Phase

- (1) U_i enters his own ID_i and PW_i and then calculates $h(Z_1) = h(h(PW_i \| ID_i) \oplus E_3)$, $HID_i = h(ID_i \| h(Z_1))$, $E_1 = h(PW_i \| HID_i) \oplus E_2$, $E_4^* = h(h(PW_i \| E_1) \| h(Z_1))$. Next, U_i compares the calculated E_4^* with the E_4 stored in the smart card. If they are equal, U_i further generates a random number Z_2 and computes $W_1 = ENC_{E_1}(Z_2 \| SN_j \| HID_i)$, $W_2 = h(Z_2 \| SN_j \| HID_i \| E_1)$, and then, U_i transmits $\{W_1, W_2, NID_i\}$ to GWN through a public channel.
- (2) GWN first calculates $E_1 = h(NID_i \| X_1)$ and then calculates $(Z_2 \| SN_j \| HID_i) = DEC_{E_1}(W_1)$ through W_1 and E_1 and then GWN queries in the database according to the calculated HID_i . Next, GWN calculates $W_2^* = h(Z_2 \| SN_j \| HID_i \| E_1)$ and verifies whether the value of W_2^* is equal to W_2 . If it is equal, the verification passes, and, thereafter, GWN generates a random number Z_3 and calculates $PK_j = h(X_2 \| SN_j)$, $W_3 = h(PK_j \| SN_j) \oplus (Z_2 \| Z_3)$, $W_4 = h(PK_j \| SN_j \| Z_2 \| Z_3)$, and then GWN transmits $\{W_3, W_4\}$ to S_j .
- (3) S_j first calculates $(Z_2 \| Z_3) = h(PK_j \| SN_j) \oplus W_3$, then S_j calculates $W_4^* = h(PK_j \| SN_j \| HID_i \| Z_2 \| Z_3)$ and checks whether it is equal to the transmitted W_4 . Next, S_j generates a random number Z_4 and then computes the session key $SK = h(SN_j \| Z_4 \| Z_3 \| Z_2)$, $W_5 = h(PK_j \| Z_3 \| SN_j) \oplus Z_4$, $W_6 = h(PK_j \| Z_3 \| Z_4 \| SK)$, and then S_j transmits $\{W_5, W_6\}$ to GWN .

- (4) *GWN* calculates $Z_4 = h(PK_j \| Z_3 \| SN_j) \oplus W_5$, obtains the session key $SK = h(SN_j \| Z_4 \| Z_3 \| Z_2)$, $W_6^* = h(PK_j \| Z_3 \| Z_4 \| SK)$, and compares the calculated W_6^* with the transmitted W_6 . Next, *GWN* generates a new gateway pseudo-identity NID_i^{new} and updates $E_1^{new} = h(NID_i^{new} \| X_1)$, $W_7 = ENC_{E_1}(Z_4 \| Z_3 \| E_1^{new} \| NID_i^{new})$, $W_8 = h(E_1^{new} \| NID_i^{new} \| SK \| Z_3 \| Z_4 \| E_1)$, and then *GWN* transmits $\{W_7, W_8\}$ to U_i .
- (5) U_i calculates $(Z_3 \| Z_4 \| E_1^{new} \| NID_i^{new}) = DEC_{E_1}(W_7)$, $SK = h(SN_j \| Z_4 \| Z_3 \| Z_2)$ and then verifies whether the calculated $W_8^* = h(E_1^{new} \| NID_i^{new} \| SK \| Z_3 \| Z_4 \| E_1)$ is equal to W_8 . If it is equal, U_i replaces $\{E_2, NID_i\}$ with $\{E_2^{new}, NID_i^{new}\}$ where $E_2^{new} = h(PW_i \| HID_i) \oplus E_1^{new}$.

5. Cryptanalysis of Sadri Et Al.'s Protocol

Here we discovered that Sadri et al.'s protocol [13] has the following security issues. First, this protocol cannot complete a sensor node authentication process. Second, this protocol is still insecure against user impersonation attacks, stolen smart card attacks, and sensor capture attacks. Third, this protocol does not provide perfect forward secrecy.

5.1. Failure Sensor Node Authentication. In the login and authentication phase, *GWN* sends $\{W_3, W_4\}$ to S_j in Step 2. After that, S_j computes $W_4^* = h(PK_j \| SN_j \| HID_i \| Z_2 \| Z_3)$. However, S_j cannot understand the value of HID_i which is an pseudo-identity of U_i . For this reason, the procedure is now terminated.

5.2. Sensor Capture Attack. Assuming that \mathcal{A} captures S_j and obtains the data $\{PK_j, SN_j\}$ stored in S_j , \mathcal{A} also eavesdrops on W_3 and W_5 transmitted over a public channel, and then \mathcal{A} can calculate SK through the following steps.

- (1) Obtain Z_2, Z_3 by calculating $h(PK_j \| SN_j) \oplus W_3 = (Z_2 \| Z_3)$.
- (2) Obtain Z_4 by calculating $h(PK_j \| Z_3 \| SN_j) \oplus W_5$.
- (3) After having SN_j, Z_4, Z_2 , and Z_3 , \mathcal{A} can calculate SK where $SK = h(SN_j \| Z_4 \| Z_3 \| Z_2)$.

5.3. Stolen Smart Card Attack. Assuming that \mathcal{A} thieves the smart card of U_i in some manner and then obtains $\{NID_i, E_1, E_2, E_3, E_4\}$ stored in this smart card, \mathcal{A} also eavesdrops on W_1 and W_7 transmitted over a public channel. Now \mathcal{A} can calculate SK through the following steps.

- (1) Obtain Z_2 and SN_j by calculating $DEC_{E_1}(W_1) = (Z_2 \| SN_j \| HID_i)$.
- (2) Obtain Z_4 and Z_3 by calculating $DEC_{E_1}(W_7) = (Z_4 \| Z_3 \| E_1^{new} \| NID_i^{new})$.
- (3) \mathcal{A} can now calculate $SK = h(SN_j \| Z_4 \| Z_3 \| Z_2)$.

5.4. User Impersonation Attack. Suppose \mathcal{A} obtains the private key of *GWN* X_1 ; then \mathcal{A} obtains some parameters

based on the obtained key to pretend to be a legitimate user and successfully authenticate each other with S_j . The specific steps are as follows:

\mathcal{A} calculates $E_1 = h(NID_i \| X_1)$ with obtained X_1 and then $(Z_2 \| SN_j \| HID_i)$ by decrypting $DEC_{E_1}(W_1)$ with E_1 . With Z_2, SN_j, HID_i , and E_1 , now \mathcal{A} can act as legitimate user to authenticate *GWN*. *GWN* verifies that W_2 is equal to $h(Z_2 \| SN_j \| HID_i \| E_1)$, and thus U_i and S_j are successfully mutually authenticated. Next, *GWN* sends a message $\{W_3, W_4\}$ to S_j , and S_j verifies that W_4 is equal to $h(PK_j \| SN_j \| HID_i \| Z_2 \| Z_3)$, because PK_j and SN_j are stored in S_j 's memory, HID_i, Z_2, Z_3 can be obtained by $(Z_2 \| Z_3) = h(PK_j \| SN_j) \oplus W_3$, and hence *GWN* and S_j are successfully authenticated. Similarly, in the process of sending information from S_j to *GWN* and the message from *GWN* to U_i , *GWN* and U_i can be successfully authenticated according to the above operations, and thus \mathcal{A} can complete a complete mutual authentication operation after obtaining the key of *GWN*.

5.5. Perfect Forward Security. Assuming that \mathcal{A} obtains X_1 in *GWN* and eavesdrops on NID_i, W_1 , and W_7 through a public channel, then \mathcal{A} can obtain SK through the following steps:

- (1) Obtain E_1 by calculating $E_1 = h(NID_i \| X_1)$.
- (2) Obtain Z_2, SN_j , and HID_i by calculating $DEC_{E_1}(W_1) = (Z_2 \| SN_j \| HID_i)$.
- (3) Obtain Z_3 and Z_4 by calculating $DEC_{E_1}(W_7) = (Z_4 \| Z_3 \| E_1^{new} \| NID_i^{new})$.
- (4) Now \mathcal{A} can compute SK where $h(SN_j \| Z_4 \| Z_3 \| Z_2)$.

6. Proposed Protocol

The protocol consists of four phases. The first phase is the predeployment phase, through which parameters required in the communication process can be deployed in advance. The second is the user registration phase, through which U_i can become a legal user through some operations to communicate with S_j . The third is the sensor registration phase, through which S_j can set the parameters required for communication with U_i in advance and then store them in memory. The fourth phase is the login and authentication phase, which can complete the key establishment process of U_i and S_j . The specific protocol steps are as follows.

6.1. Predeployment Phase. In this phase, each user needs to negotiate a key with *GWN*. Assume that a user U_i wants to join this system and has negotiated a key G_{gu} with *GWN*. In our design, we assume that this key cannot be obtained by adversaries.

6.2. User Registration Phase. Assume that a user U_i wishes to register to *GWN*. Figure 2 displays the user registration phase. Messages transmitted in this phase are via a secure channel.

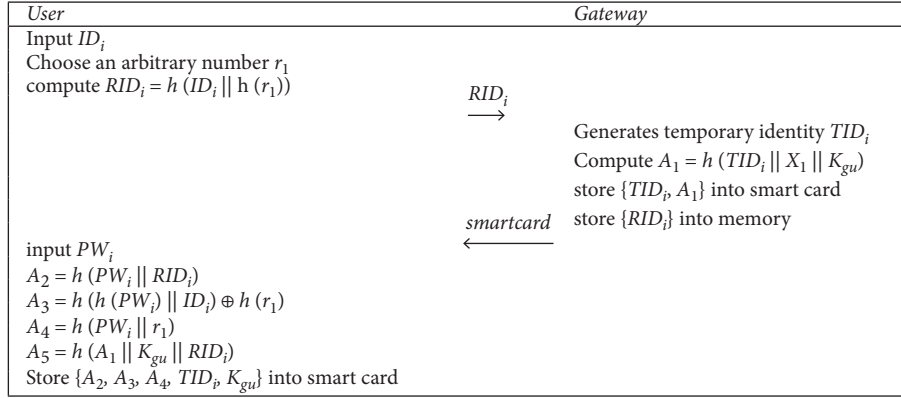


FIGURE 2: User registration phase.

- (1) U_i selects an identity ID_i for himself, generates a random number r_1 , calculates the user's pseudo-identity $RID_i = h(ID_i || h(r_1))$, and then sends RID_i to GWN .
- (2) GWN first checks whether RID_i has been registered before and then generates a temporary identity TID_i for U_i and calculates $A_1 = h(TID_i || X_1 || K_{gu})$. After that, GWN stores $\{TID_i, A_1\}$ in a smart card and stores U_i 's pseudo-identity RID_i in its own memory. Finally, GWN transmits this smart card to U_i .
- (3) U_i enters his password PW_i , and then U_i protects his password PW_i from being obtained by \mathcal{A} by calculating $A_2 = h(PW_i || RID_i)$. Next, U_i calculates $A_3 = h(h(PW_i) || ID_i) \oplus h(r_1)$, $A_4 = h(PW_i || r_1)$, $A_5 = h(A_1 || K_{gu} || RID_i)$, and finally U_i stores $\{A_2, A_3, A_4, TID_i, K_{gu}\}$ in the smart card.

6.3. Sensor Registration Phase. Assume that a sensor node S_j wishes to register with GWN ; S_j performs the following steps. Figure 3 illustrates the sensor registration phase. Note that all messages transmitted in this phase are via a secure channel.

GWN first selects an identity SN_j for S_j and then uses the GWN 's private key X_1 to protect the identity of S_j to obtain the pseudo-identity of S_j , which is $U = h(SN_j || X_2)$. Finally, GWN transmits $\{U, SN_j, RID_i\}$ to S_j . After receiving these messages, S_j stores the information $\{U, SN_j, RID_i\}$ in its own memory.

6.4. Login and Authentication Phase. Here we describe this phase between U_i and S_j with GWN . The following steps are performed. Note that all transmissions in this phase are through a public channel. Figure 4 illustrates the architecture of this phase.

- (1) U_i first enters his ID_i and PW_i to log in and calculates $h(r_1) = h(h(PW_i) || ID_i) \oplus A_3$, and U_i then calculates U_i 's pseudo-identity $RID_i = h(ID_i || h(r_1))$ from the calculated $h(r_1)$ and then verifies whether $h(PW_i || h(r_1))$ is equal to the A_4 stored in the smart card. If they are equal, it proves that the legitimate user has successfully logged in. Next, U_i generates a random number r_2 , calculates $Y_1 = ENC_{A_5}(r_2 || SN_j ||$

$RID_i)$, $Y_2 = h(r_2 || SN_j || RID_i || A_1)$, and then transmits $\{Y_1, Y_2, TID_i\}$ to GWN .

- (2) GWN first calculates $A_1 = h(TID_i || X_1 || K_{gu})$ and then obtains $(r_2 || SN_j || RID_i) = DEC_{A_5}(Y_1)$ through a symmetric decryption operation. GWN searches the database according to the obtained pseudo-identity of U_i and then verifies whether $h(r_2 || SN_j || RID_i || A_1)$ is the same as the received Y_2 . After that, GWN generates r_3 and computes $U = h(X_2 || SN_j)$, $Y_3 = h(U || SN_j) \oplus (r_2 || r_3 || RID_i)$, and $Y_4 = h(U || SN_j || r_2 || r_3 || RID_i)$. Finally, GWN transmits $\{Y_3, Y_4\}$ to S_j .
- (3) S_j first calculates $(r_2 || r_3 || RID_i) = h(U || SN_j) \oplus Y_3$ to obtain r_2, r_3 , and RID_i and then checks the legitimacy of GWN by verifying whether $h(U || SN_j || r_2 || r_3 || RID_i)$ is equal to Y_4 . After that, S_j generates r_4 and then computes $SK = h(SN_j || r_2 || r_3 || r_4)$. S_j generates another random number L and calculates $W_L = ENC_{PU_G}(L)$ and then calculates $Y_5 = h(U || r_3 || SN_j || L) \oplus r_4$, $Y_6 = h(U || r_3 || r_4 || SK)$. Now S_j transmits $\{Y_5, Y_6, W_L\}$ to GWN .
- (4) GWN first calculates $L = DEC_{PR_G}(W_L)$ and computes the random number $r_4 = Y_5 \oplus h(U || r_3 || SN_j || L)$. Then GWN calculates the session key $SK = h(SN_j || r_2 || r_3 || r_4)$ and verifies whether Y_6 is equal to $h(U || r_3 || r_4 || SK)$. If they are equal, GWN starts to update U_i 's temporary identity. GWN generates a new temporary identity TID_i^{new} , calculates $A_1^{new} = h(TID_i^{new} || X_1 || K_{gu})$, $Y_7 = ENC_{A_5}(r_4 || r_3 || A_1^{new} || TID_i^{new})$, $Y_8 = h(A_1^{new} || TID_i^{new} || SK || r_3 || r_4 || A_1)$, and transmits $\{Y_7, Y_8\}$ to U_i .
- (5) U_i first decrypts Y_7 with A_5 to obtain $(r_3 || r_4 || A_1^{new} || TID_i^{new})$ and then generates the session key $SK = h(SN_j || r_2 || r_3 || r_4)$. U_i then verifies whether Y_8 is equal to $h(A_1^{new} || TID_i^{new} || SK || r_3 || r_4 || A_1)$. Finally, U_i updates A_2 to obtain $A_2^{new} = h(PW_i || RID_i) \oplus A_1^{new}$ and replaces $\{A_2, TID_i\}$ with $\{A_2^{new}, TID_i^{new}\}$.

7. Security Analysis

Here we utilize ROR model and BAN logic to demonstrate our protocol is provably secure. We also indicate that our work is secure against various attacks.

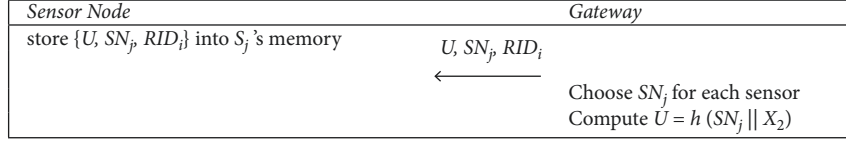


FIGURE 3: Sensor registration phase.

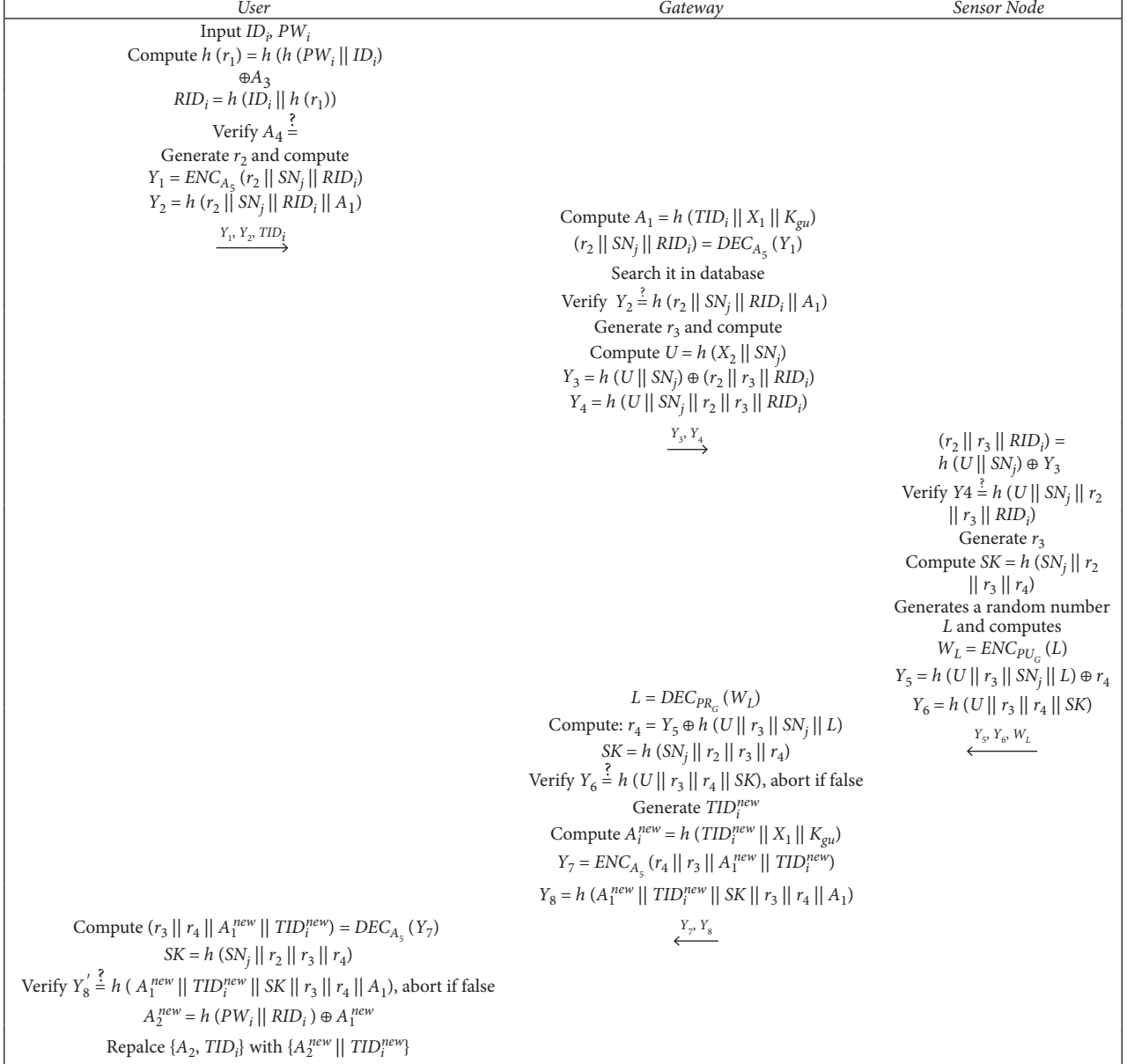


FIGURE 4: Login and authentication phase.

7.1. ROR Security Analysis

7.1.1. ROR Model. Random Oracle (ROR) model [38] is widely utilized to achieve a formal security analysis of an authentication protocol. Theorem 1 proves the security of the session key in our protocol.

In our design, we defined three entities: U_i , GW_N , and S_j . In the proof process, we assume that U_a , G_b , and S_c are the a -th user, b -th gateway, and c -th sensor node, and $P = \{U_a, G_b, S_c\}$. Because \mathcal{A} can completely control a public channel, \mathcal{A} can intercept, modify, and delete messages transmitted through a public channel. In addition, \mathcal{A} can execute the following queries:

Execute(P): \mathcal{A} can destroy messages shared between various entities.

Send(P, M): \mathcal{A} can transmit information M to P .

Hash(string): By doing this, after entering a series of strings, \mathcal{A} can obtain a fixed value.

CorruptSmartCard(U_a): By executing this query, \mathcal{A} can easily steal U_i 's smart card and extract the parameters stored in it.

CorruptSensor(G_b): By executing this query, all the information stored in S_j is exposed to \mathcal{A} .

Test(T): This operation is performed to verify the security of the shared secret key between U_i and S_j . Toss a coin before experimenting. If the result is 1, the correct session key is returned, and a random number is returned if the result is 0.

7.1.2. Security Proof

Theorem 1. *Suppose \mathcal{A} is an attacker running in polynomial t time to counter the protocol we proposed under the ROR model. Here, C is a unified dictionary and f is the number of digits of the key in the biometric information of U_i . The advantage of \mathcal{A} 's ability to destroy our new protocol is as follows:*

$Adv_{\mathcal{A}}^{\mathcal{P}} \leq (q_h^2/|H|) + (2q_s/|C|q_h)$, q_s , $|H|$, and $|C|$ represent the Hash query, Send query, the spatial range of the hash function, and the size of the unified dictionary, respectively.

To prove the security of the session key, we defined four games: GM_i ($i = 0, 1, 2, 3$); $Succ_G$ represents \mathcal{A} 's victory in the game. We start from GM_0 to the end of GM_3 . The detailed procedure is as follows:

GM_0 : In the initial game, \mathcal{A} does not need to perform any query operations and only needs to select bits; hence we obtain

$$Adv_{\mathcal{A}}^{\mathcal{P}}(t) = \left| 2Pr[Succ_{\mathcal{A}}^{\mathcal{P}}(t) - 1] \right|. \quad (1)$$

GM_1 : In this game, we simulated an eavesdropping attack. \mathcal{A} executes the Execute(P) query and then verifies whether SK is a random number or a real key by executing a Test(P) query. The session key $SK = h(SN_j \| r_2 \| r_3 \| r_4)$ in the protocol; we assume that \mathcal{A} intercepts messages M_1, M_2, M_3, M_4 , and M_5 sent by U_i and S_j , but only calculated SK based on the intercepted information is impossible; hence we obtain

$$Pr[Succ_{\mathcal{A}}^{GM_1}] = Pr[Succ_{\mathcal{A}}^{GM_0}]. \quad (2)$$

GM_2 : The third game adds the Send(P) query operation and Hash(string) operation to the second game. In this game, \mathcal{A} can use Send(P) query operations and Hash(string) operations to obtain some parameter information, and \mathcal{A} can fabricate some entity messages. To create authentic and credible messages M_1, M_2, M_3, M_4 , and M_5 , \mathcal{A} must know the secret parameters r_2, r_3, r_4 , and SN_j , but these secret parameters are hidden in

the hash function to prevent \mathcal{A} from stealing. Based on the birthday paradox, we can draw the following conclusions:

$$\left| Pr[Succ_{\mathcal{A}}^{GM_1}] - Pr[Succ_{\mathcal{A}}^{GM_2}] \right| \leq \frac{q_h^2}{|H|}. \quad (3)$$

GM_3 : On the basis of the previous game, the operations of performing the CorruptSmartCard(U_a) and CorruptSensor(G_b) are added. \mathcal{A} can extract the information $\{A_1, A_2, A_3, A_4, TID_i, K_{gu}\}$ in the smart card and guess the password of U_i based on the unified dictionary. Without knowing the password of U_i , it is extremely difficult for \mathcal{A} to obtain SK ; hence we arrive at

$$\left| Pr[Succ_{\mathcal{A}}^{GM_2}] - Pr[Succ_{\mathcal{A}}^{GM_3}] \right| \leq \frac{q_s}{|C|}. \quad (4)$$

Because the session key SK is generated for mutual authentication between U_i and S_j , we obtain

$$Pr[Succ_{\mathcal{A}}^{GM_3}] = \frac{1}{2}. \quad (5)$$

After we sort the above equations, we obtain

$$\begin{aligned} \frac{1}{2} Adv_{\mathcal{A}}^{\mathcal{P}} &= \left| Pr[Succ_{\mathcal{A}}^{GM_0}] - \frac{1}{2} \right| = \left| Pr[Succ_{\mathcal{A}}^{GM_0}] - Pr[Succ_{\mathcal{A}}^{GM_3}] \right| \\ &= \left| Pr[Succ_{\mathcal{A}}^{GM_1}] - Pr[Succ_{\mathcal{A}}^{GM_3}] \right| \\ &\leq \sum_{i=1}^3 \left| Pr[Succ_{\mathcal{A}}^{GM_i}] - Pr[Succ_{\mathcal{A}}^{GM_{i+1}}] \right| = \frac{q_h^2}{2} |H| + \frac{q_s}{|C|}. \end{aligned} \quad (6)$$

So we can draw the final conclusion:

$$Adv_{\mathcal{A}}^{\mathcal{P}} \leq \frac{q_h^2}{|H|} + \frac{2q_s}{|C|}. \quad (7)$$

Based on the conclusions drawn above, it can be proved that our protocol can be secure against stolen smart card attacks and sensor capture attacks. Moreover, it has perfect forward security.

7.2. BAN Security Analysis. Burrows-Abadi-Needham (BAN) logic [39] is a method suitable for analyzing an authentication protocol. It mainly studies the security of the protocol and the logic of the structure. BAN logic has been applied to the security analysis of many protocols and achieved good results.

7.2.1. Rules for BAN Logic.

Message-meaning rule (R1) $U \equiv U \stackrel{K}{\Leftarrow} KV, U \triangleleft \langle X \rangle_K$
 $|U| \equiv V| \sim X$.

Nonce-verification rule (R2) $U| \equiv \#(X), U| \equiv V| \sim X/$
 $U| \equiv V| \equiv X$.

Jurisdiction rule (R3) $U| \equiv V| \Rightarrow X, U| \equiv V| \equiv X/$
 $U| \equiv X$.

Freshness rule (R4) $U \equiv \#(X)/U \equiv \#(X, Y)$.

Belief rule (R5) $U \equiv X, U \equiv Y/U \equiv (X, Y)$.

Session key rule (R6) $U \equiv \#(X), U \equiv V \equiv X/U \equiv U \xleftrightarrow{K} V$.

7.2.2. Goals.

G1 $U \equiv U \xleftrightarrow{SK} G$.

G2 $G \equiv U \xleftrightarrow{SK} G$.

G3 $U \equiv G \equiv U \xleftrightarrow{SK} G$.

G4 $G \equiv U \equiv U \xleftrightarrow{SK} G$.

G5 $S \equiv S \xleftrightarrow{SK} G$.

G6 $G \equiv S \xleftrightarrow{SK} G$.

G7 $S \equiv G \equiv S \xleftrightarrow{SK} G$.

G8 $G \equiv S \equiv S \xleftrightarrow{SK} G$.

7.2.3. Idealizing Communication.

Mess1 $U \longrightarrow G: \{Y_1, Y_2, TID_i\}$.

Mess2 $G \longrightarrow S: \{Y_3, Y_4\}$.

Mess3 $S \longrightarrow G: \{Y_5, Y_6, W_L\}$.

Mess4 $G \longrightarrow U: \{Y_7, Y_8\}$.

7.2.4. Initial State Assumptions.

A1 $G \equiv U \stackrel{A_5}{=} G$.

A2 $G \equiv \#(r_2, SN_j)$.

A3 $G \equiv U \Rightarrow (r_2, SN_j)$.

A4 $G \equiv \#(r_3, r_4)$.

A5 $U \equiv U \stackrel{A_5}{=} G$.

A6 $U \equiv \#(r_3, r_4)$.

A7 $U \equiv \#(r_2, SN_j)$.

A8 $U \equiv G \Rightarrow (r_3, r_4)$.

A9 $S \equiv S \stackrel{U}{=} UG$.

A10 $G \equiv S \stackrel{U}{=} UG$.

A11 $S \equiv \#(r_2, r_3)$.

A12 $S \equiv G \Rightarrow (r_2, r_3)$.

A13 $G \equiv S \Rightarrow (r_4)$.

7.2.5. Detailed Steps.

By considering the message Mess1, we get

S1: $G \triangleleft \{ \langle r_2, SN_j \rangle_{A_5}, Y_2, TID_i \}$.

Using S1, R1, and A1, we get

S2: $G \equiv U \sim (r_2, SN_j)$.

Under the assumption of A2, using S2, R2 can be used to

obtain

S3: $G \equiv U \equiv (r_2, SN_j)$.

With conclusion S3, using A3 and R3, the following can

be obtained:

S4: $G \equiv (r_2, SN_j)$.

Because $r_4 = Y_5 \oplus h(U \| r_3 \| SN_j \| L)$, $L = DEC_{PRG}(W_L)$, we

can get

S5: $G \equiv (r_4)$.

And because r_3 is generated by G ,

S6: $G \equiv (r_3)$.

Because $SK = h(SN_j \| r_2 \| r_3 \| r_4)$, using S4, S5, and S9, we

obtain

S7: $G \equiv U \xleftrightarrow{SK} G$ (G2).

With A3, A4, S7, and R4, we can get

S8: $G \equiv U \equiv U \xleftrightarrow{SK} G$ (G4).

By considering the message Mess4, we obtain

S9: $U \triangleleft \{ \langle r_2, r_4 \rangle_{A_5}, Y_8 \}$.

By using S9, A5, and R1 we obtain

S10: $U \equiv G \sim (r_3, r_4)$.

With S10, using A6 and applying R2, we get

S11: $U \equiv G \equiv (r_3, r_4)$.

Applying A8, S11, and R3, we have

S12: $U \equiv (r_3, r_4)$.

Because $SK = h(SN_j \| r_2 \| r_3 \| r_4)$, using S11 and S13, we

obtain

S13: $U \equiv G \xleftrightarrow{SK} U$ (G1).

With conclusion S13, using A6, A7, and R4, we can

obtain

S14: $U \equiv G \equiv U \xleftrightarrow{SK} G$ (G3).

By considering the message Mess2, we obtain

S15: $S \triangleleft \{ \langle r_2, r_3 \rangle_U, Y_4 \}$.

By using S15, A9, and R1, we obtain

S16: $S \equiv G \sim (r_2, r_3)$.

With S16, using A11 and applying R2, we have

S17: $S \equiv G \equiv (r_2, r_3)$.

Applying A12, S17, and R3, we obtain

S18: $S \equiv (r_2, r_3)$.

Because $SK = h(SN_j \| r_2 \| r_3 \| r_4)$, and SN_j and r_4 are

generated by S , we can get

S19: $S \equiv (SN_j, r_4)$.

Using S18 and S19, we obtain

S20: $S \equiv S \xleftrightarrow{SK} S$ (G5).

With S19, using A11 and R4 we can get

S14: $S \equiv G \equiv S \xleftrightarrow{SK} G$ (G7).

By considering the message Mess3, we obtain

S22: $G \triangleleft \{ \langle r_4 \rangle_U, Y_6, \langle L \rangle_{PU_G} \}$.

By using S22, A10, and R1 we obtain

S23: $G \equiv S \sim (r_4)$.

With S23, using A4 and applying R2 we have

S24: $G \equiv S \equiv (r_4)$.

Applying A13, S24, and R3 we obtain

S25: $G \equiv (r_4)$.

Because $SK = h(SN_j \| r_2 \| r_3 \| r_4)$, using S25, S4, and S6, we

obtain

S26: $G \equiv G \xleftrightarrow{SK} S$ (G6).

With conclusion S26, using A2, A4, and R4 we can obtain

S27: $G \equiv S \equiv S \xleftrightarrow{SK} G$ (G8).

7.3. Potential Attacks

7.3.1. Withstands Stolen Smart Card Attack. We assume that \mathcal{A} obtains U_i 's smart card and the parameters $\{A_1, A_2, A_3, A_4, TID_i, K_{gu}\}$ in the smart card through power analysis. Although \mathcal{A} obtains these parameters, \mathcal{A} cannot obtain A_5 because calculating A_5 also requires RID_i . Therefore, \mathcal{A} cannot further decrypt Y_1 and Y_7 to obtain SN_j, r_2, r_3 , and r_4 . It means that \mathcal{A} cannot calculate SK . Now

TABLE 2: Comparisons of security.

Protocols	Amin et al. [40]	Chen et al. [41]	Wu et al. [28]	Sadri and Asaar [13]	Our protocol
Reply attack	Y	Y	Y	Y	Y
Stolen smart card attack	N	Y	Y	N	Y
Stolen verifier attack	N	Y	Y	N	Y
Offline password guessing attack	Y	N	Y	Y	Y
Sensor capture attack	Y	Y	N	N	Y
Provide perfect forward security	Y	Y	N	N	Y

TABLE 3: Time consumed by different phases.

Phase	Device	Hash (T_h)	Encryption (T_S)/decryption (T_D)	Point addition (T_p)
User	Honor 30s	0.0049 ms	17.213 ms	0.4894 ms
Gateway	MSI-GP63	0.0025 ms	8.094 ms	0.0527 ms
Sensor	Lenovo-M715E	0.0044 ms	11.477 ms	0.1723 ms

we can say that our protocol can resist stolen smart card attacks.

7.3.2. Withstands Sensor Capture Attack. Assuming that \mathcal{A} captures the sensor and extracts the message $\{U, SN_j, RID_i\}$ stored in S_j , now \mathcal{A} can obtain r_2 and r_3 with Y_3 and the messages extracted from the sensor. However, \mathcal{A} cannot further obtain L and r_4 because \mathcal{A} does not have X_2 . For this reason, \mathcal{A} cannot calculate SK through sensor capture attacks.

7.3.3. Withstands User Impersonation Attack. Suppose \mathcal{A} obtains GWN key X_1 , but the shared key K_{gu} between GWN and U_i cannot be obtained; therefore, parameter A_1 cannot be obtained. Without A_1 , \mathcal{A} cannot obtain A_5 . Also, without A_5 , \mathcal{A} cannot decrypt Y_1 to get SN_j and r_2 . It is obvious that our design can resist user impersonation attacks.

7.3.4. Perfect Forward Security. Assuming \mathcal{A} obtains the long-term key X_1 of GWN, \mathcal{A} cannot obtain K_{gu} . However, \mathcal{A} cannot obtain A_1 , A_5 and further calculate SK . Now we can say the proposed protocol provides the perfect forward security.

7.3.5. Withstands Offline Password Guessing Attack. In the login and authentication phase, we assume that \mathcal{A} tries to guess pw_i . \mathcal{A} uses the guessed password pw'_i to login. Since \mathcal{A} does not know ID_i and A_3 , \mathcal{A} cannot verify if pw'_i is correct. Thus, our protocol can effectively resist offline password guessing attacks.

7.4. Security Comparisons. Table 2 shows the security comparison between the proposed protocol and other related protocols. The outcomes reveal that other related protocols [13, 28, 40, 41] have various flaws in security, but our protocol can withstand different kinds of attacks.

8. Performance Comparisons

8.1. Experimental Setting. To investigate the performance of our work, we use a mobile phone (Honor 30S, CPU: HUAWEI Kirin 820, 8 GB), a notebook computer (MSI-GP63, CPU: Intel Core i7 8750H, 8 GB), and a desktop computer (Lenovo-M715E, CPU: Intel Pentium Dual-Core E5500, 2 GB) to simulate a user, a gateway, and a sensor, respectively. We use these devices to calculate the execution time of hash functions, symmetric encryption systems, and point addition functions. Each operation was executed 10 times, and the average running times were calculated. Table 3 lists our experimental results. Here, we do not evaluate the impact of the XOR operation since its running time is not worth mentioning compared with other functions.

8.2. Performance Comparisons. First, the computation cost of our work was compared with related protocols [13, 28, 40, 41]. We emphasize the login and authentication phase since this phase is frequently performed. Table 4 shows the running times of these five protocols. T_h , T_S , T_D , and T_p denote the running times of hash functions, symmetric encryption, symmetric decryption, and point addition function individually.

As depicted in Table 4, in our protocol, the running time for a user is 17.2522 ms, which is higher than Amin et al.'s protocol [40] and Wu et al.'s protocol [28]. Besides, the running time for a gateway in our design is 16.213 ms, which is slightly higher than Wu et al.'s protocol [28] and Chen et al.'s [41]. However, the running time for a sensor in our design is 0.022 ms which is obviously lower than Wu et al.'s protocol [28] and Chen et al.'s [41]. Overall, the total time consumed by the proposed protocol is 33.4872 ms, which is lower than Chen et al.'s protocol [41] and Sadri et al.'s protocol [13]. The running time of the proposed protocol is slightly higher than Amin et al.'s protocol [40] and Wu et al.'s protocol [28].

Furthermore, we consider the communication overhead. Assume that the output of a hash function is 256 bits, a random number is 160 bits, the identity is 160 bits, the symmetric encryption parameter is 128 bits, and a

TABLE 4: Calculation cost comparison.

Protocol	User (ms)	Gateway (ms)	Sensor (ms)	Total (ms)	Communication cost (bits)
Amin et al. [40]	$12 T_h \approx 0.0882$	$18 T_h \approx 0.045$	$6 T_h \approx 0.0264$	0.1596	9216
Chen et al. [41]	$8 T_h + T_s \approx 17.2552$	$3 T_h + 2 T_s \approx 16.1955$	$5 T_h + T_s \approx 11.499$	44.9497	4608
Wu et al. [28]	$13 T_h \approx 0.0637$	$3 T_h + 2 T_s \approx 16.1955$	$5 T_h + T_s \approx 11.499$	27.7582	8192
Sadri and Asaar [13]	$6 T_h + T_s + T_D \approx 34.4554$	$10 T_h + T_s + T_D \approx 16.213$	$5 T_h \approx 0.022$	50.6904	5888
Our protocol	$8 T_h + T_s \approx 17.2522$	$10 T_h + T_s + T_D \approx 16.213$	$5 T_h \approx 0.022$	33.4872	6400

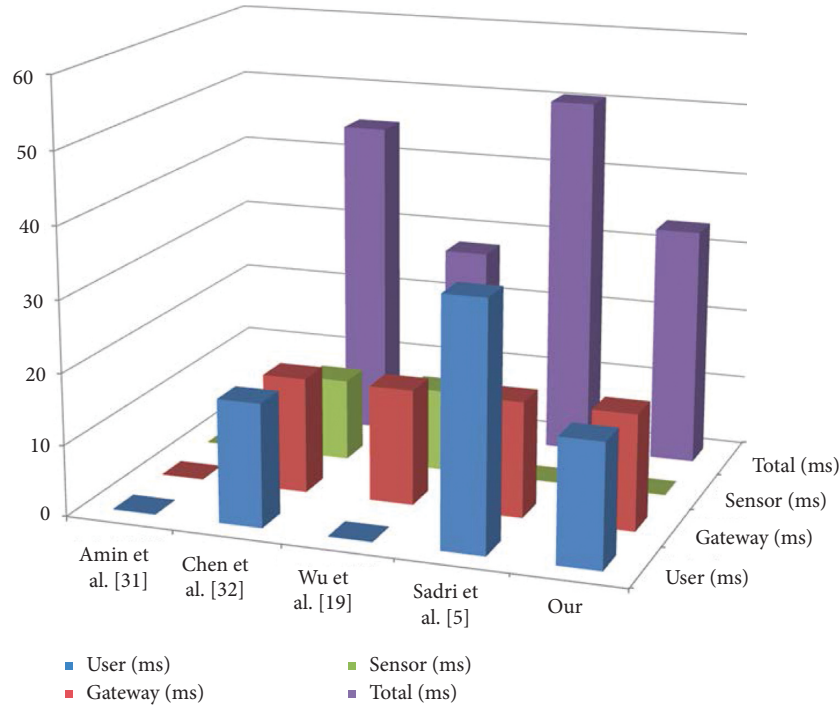


FIGURE 5: Running time.

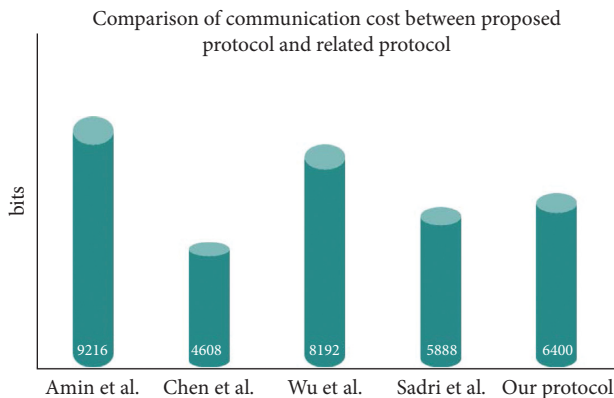


FIGURE 6: Communication cost.

timestamp is 64 bits. As shown in Table 4, the communication cost of the proposed protocol is 6400 bits, which is higher than Chen et al.'s protocol [41] and Sadri et al.'s protocol [13].

We can observe that the proposed protocol does not have the best performance, but our design delivers better security than other protocols. We can say that our work has more practical significance for developing the IoT in the future. Figure 5 and Figure 6 show a comparison more intuitively.

9. Conclusions

IoT technology is constantly improving and updating. Protecting the security and privacy of data in IoT is an essential task. This paper demonstrated that Sadri's protocol has some security issues. To solve these issues, we proposed a new protocol. We prove that our work is probably secure through BAN logic and ROR model, which can better ensure data security in the transmission process. Performance evaluation indicates that the proposed protocol has reasonable computation and communication overhead and thus has more practical significance for developing the IoT in the future.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work was supported by Guangxi Key Laboratory of Trusted Software (no. KX202033).

References

- [1] X. Huang, H. Xiong, J. Chen, and M. Yang, "Efficient revocable storage attribute-based encryption with arithmetic span programs in cloud-assisted internet of things," *IEEE Transactions on Cloud Computing*, 2021.
- [2] S. A. Kumar, T. Vealey, and H. Srivastava, "Security in internet of things: challenges, solutions and future directions," in *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)*, pp. 5772–5781, IEEE, Koloa, HI, USA, January 2016.
- [3] J. Sengupta, S. Ruj, and S. Das Bit, "A comprehensive survey on attacks, security issues and blockchain solutions for iot and iiot," *Journal of Network and Computer Applications*, vol. 149, Article ID 102481, 2020.
- [4] F. Al-Turjman and S. Alturjman, "Context-sensitive access in industrial internet of things (iiot) healthcare applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2736–2744, 2018.
- [5] H. Xiong, J. Chen, Q. Mei, and Y. Zhao, "Conditional privacy-preserving authentication protocol with dynamic membership updating for vanets," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, p. 1, 2020.
- [6] M. A. Khan, I. Ullah, A. Alkhalifah et al., "A provable and privacy-preserving authentication scheme for uav-enabled intelligent transportation systems," *IEEE Transactions on Industrial Informatics*, vol. 18, 2021.
- [7] S. Hussain, K. Mahmood, M. K. Khan, C.-M. Chen, B. A. Alzahrani, and S. A. Chaudhry, "Designing secure and lightweight user access to drone for smart city surveillance," *Computer Standards & Interfaces*, vol. 80, Article ID 103566, 2022.
- [8] C.-M. Chen, C.-T. Li, S. Liu, T.-Y. Wu, and J.-S. Pan, "A provable secure private data delegation scheme for mountaineering events in emergency system," *IEEE Access*, vol. 5, pp. 3410–3422, 2017.
- [9] F. Fan, S.-C. Chu, J.-S. Pan, C. Lin, and H. Zhao, "An optimized machine learning technology scheme and its application in fault detection in wireless sensor networks," *Journal of Applied Statistics*, pp. 1–18, 2021.
- [10] X. Xue and C. Jiang, "Matching sensor ontologies with multi-context similarity measure and parallel compact differential evolution algorithm," *IEEE Sensors Journal*, vol. 21, no. 21, pp. 24570–24578, 2021.
- [11] M. A. Khan, B. A. Alzahrani, A. Barnawi, A. Al-Barakati, A. Irshad, and S. A. Chaudhry, "A resource friendly authentication scheme for space-air-ground-sea integrated maritime communication network," *Ocean Engineering*, vol. 250, Article ID 110894, 2022.
- [12] T.-Y. Wu, L. Yang, Z. Lee, S.-C. Chu, S. Kumari, and S. Kumar, "A provably secure three-factor authentication protocol for wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 5537018, 15 pages, 2021.
- [13] M. J. Sadri and M. R. Asaar, "An anonymous two-factor authentication protocol for iot-based applications," *Computer Networks*, vol. 199, Article ID 108460, 2021.
- [14] M.-S. Jian and J. M.-T. Wu, "Hybrid internet of things (iot) data transmission security corresponding to device verification," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–10, 2021.
- [15] S. A. Chaudhry, A. Irshad, J. Nebhen et al., "An anonymous device to device access control based on secure certificate for internet of medical things systems," *Sustainable Cities and Society*, vol. 75, Article ID 103322, 2021.
- [16] X. Xue and J. Chen, "Using compact evolutionary tabu search algorithm for matching sensor ontologies," *Swarm and Evolutionary Computation*, vol. 48, pp. 25–30, 2019.
- [17] S.-C. Chu, T.-K. Dao, J.-S. Pan, and T. T. Nguyen, "Identifying correctness data scheme for aggregating data in cluster heads of wireless sensor network based on naive bayes classification," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 52–15, 2020.
- [18] G. T. Reddy, R. Kaluri, P. K. Reddy, K. Lakshmana, S. Koppu, and D. S. Rajput, "A novel approach for home surveillance system using iot adaptive security," in *Proceedings of the International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM)*, Amity University Rajasthan, Jaipur-India, 2019.
- [19] C.-T. Li, C.-C. Lee, C.-Y. Weng, and C.-M. Chen, "Towards secure authenticating of cache in the reader for rfid-based iot systems," *Peer-to-Peer Networking and Applications*, vol. 11, no. 1, pp. 198–208, 2018.
- [20] M. Turkanović, B. Brumen, and M. Hölbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the internet of things notion," *Ad Hoc Networks*, vol. 20, pp. 96–112, 2014.
- [21] R. Amin and G. Biswas, "A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks," *Ad Hoc Networks*, vol. 36, pp. 58–80, 2016.
- [22] Z. Xu, C. Xu, H. Chen, and F. Yang, "A lightweight anonymous mutual authentication and key agreement scheme for wban," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 14, Article ID e5295, 2019.
- [23] B. A. Alzahrani, A. Irshad, A. Albeshri, and K. Alsubhi, "A provably secure and lightweight patient-healthcare authentication protocol in wireless body area networks," *Wireless Personal Communications*, vol. 117, no. 1, pp. 47–69, 2021.
- [24] S. Shin and T. Kwon, "A privacy-preserving authentication, authorization, and key agreement scheme for wireless sensor networks in 5g-integrated internet of things," *IEEE Access*, vol. 8, Article ID 67555, 2020.
- [25] A. Adavoudi-Jolfaei, M. Ashouri-Talouki, and S. F. Aghili, "Lightweight and anonymous three-factor authentication and access control scheme for real-time applications in wireless sensor networks," *Peer-to-Peer Networking and Applications*, vol. 12, no. 1, pp. 43–59, 2019.
- [26] A. Tewari and B. B. Gupta, "Cryptanalysis of a novel ultralightweight mutual authentication protocol for iot devices using rfid tags," *The Journal of Supercomputing*, vol. 73, no. 3, pp. 1085–1102, 2017.
- [27] Q. Jiang, S. Zeadally, J. Ma, and D. He, "Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks," *IEEE Access*, vol. 5, pp. 3376–3392, 2017.
- [28] F. Wu, X. Li, L. Xu, P. Vijayakumar, and N. Kumar, "A novel three-factor authentication protocol for wireless sensor networks with iot notion," *IEEE Systems Journal*, vol. 15, no. 1, pp. 1120–1129, 2021.
- [29] 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.
- [30] A. K. Das, M. Wazid, N. Kumar, M. K. Khan, K.-K. R. Choo, and Y. Park, "Design of secure and lightweight authentication protocol for wearable devices environment," *IEEE journal of*

- biomedical and health informatics*, vol. 22, no. 4, pp. 1310–1322, 2018.
- [31] J. Lee, S. Yu, M. Kim, Y. Park, and A. K. Das, “On the design of secure and efficient three-factor authentication protocol using honey list for wireless sensor networks,” *IEEE Access*, vol. 8, Article ID 107046, 2020.
 - [32] H.-T. Pan, C.-S. Pan, S.-C. Tsaur, and M.-S. Hwang, “Cryptanalysis of efficient dynamic id based remote user authentication scheme in multi-server environment using smart card,” in *Proceedings of the 12th International Conference on Computational Intelligence and Security (CIS)*, pp. 590–593, IEEE, Wuxi, China, December 2016.
 - [33] K. Yahya, S. A. Chaudhry, and F. Al-Turjman, “On the security of an authentication scheme for smart metering infrastructure,” in *Proceedings of the Emerging Technology in Computing, Communication and Electronics (ETCCE)*, pp. 1–6, IEEE, Bangladesh, December 2020.
 - [34] K. Mahmood, W. Akram, A. Shafiq, I. Altaf, M. A. Lodhi, and S. H. Islam, “An enhanced and provably secure multi-factor authentication scheme for internet-of-multimedia-things environments,” *Computers & Electrical Engineering*, vol. 88, Article ID 106888, 2020.
 - [35] D. K. Kwon, S. J. Yu, J. Y. Lee, S. H. Son, and Y. H. Park, “Wsn-slap: secure and lightweight mutual authentication protocol for wireless sensor networks,” *Sensors*, vol. 21, no. 3, 2021.
 - [36] S. D. Kaul and A. K. Awasthi, “Security enhancement of an improved remote user authentication scheme with key agreement,” *Wireless Personal Communications*, vol. 89, no. 2, pp. 621–637, 2016.
 - [37] C. Chunka and S. Banerjee, “An efficient mutual authentication and symmetric key agreement scheme for wireless body area network,” *Arabian Journal for Science and Engineering*, vol. 46, no. 9, pp. 8457–8473, 2021.
 - [38] M. Abdalla, P.-A. Fouque, and D. Pointcheval, “Password-based authenticated key exchange in the three-party setting,” in *Proceedings of the Public Key Cryptography - PKC 2005, International Workshop on Public Key Cryptography*, pp. 65–84, Springer, Les Diablerets, Switzerland, January 2005.
 - [39] M. Burrows, M. Abadi, and R. M. Needham, “A logic of authentication,” *Proceedings of the twelfth ACM symposium on Operating systems principles - SOSp '89*, vol. 426, no. 1871, pp. 233–271, 1989.
 - [40] R. Amin, S. H. Islam, G. Biswas, M. K. Khan, and N. Kumar, “A robust and anonymous patient monitoring system using wireless medical sensor networks,” *Future Generation Computer Systems*, vol. 80, pp. 483–495, 2018.
 - [41] Y. Chen, Y. Ge, Y. Wang, and Z. Zeng, “An improved three-factor user authentication and key agreement scheme for wireless medical sensor networks,” *IEEE Access*, vol. 7, Article ID 85440, 2019.