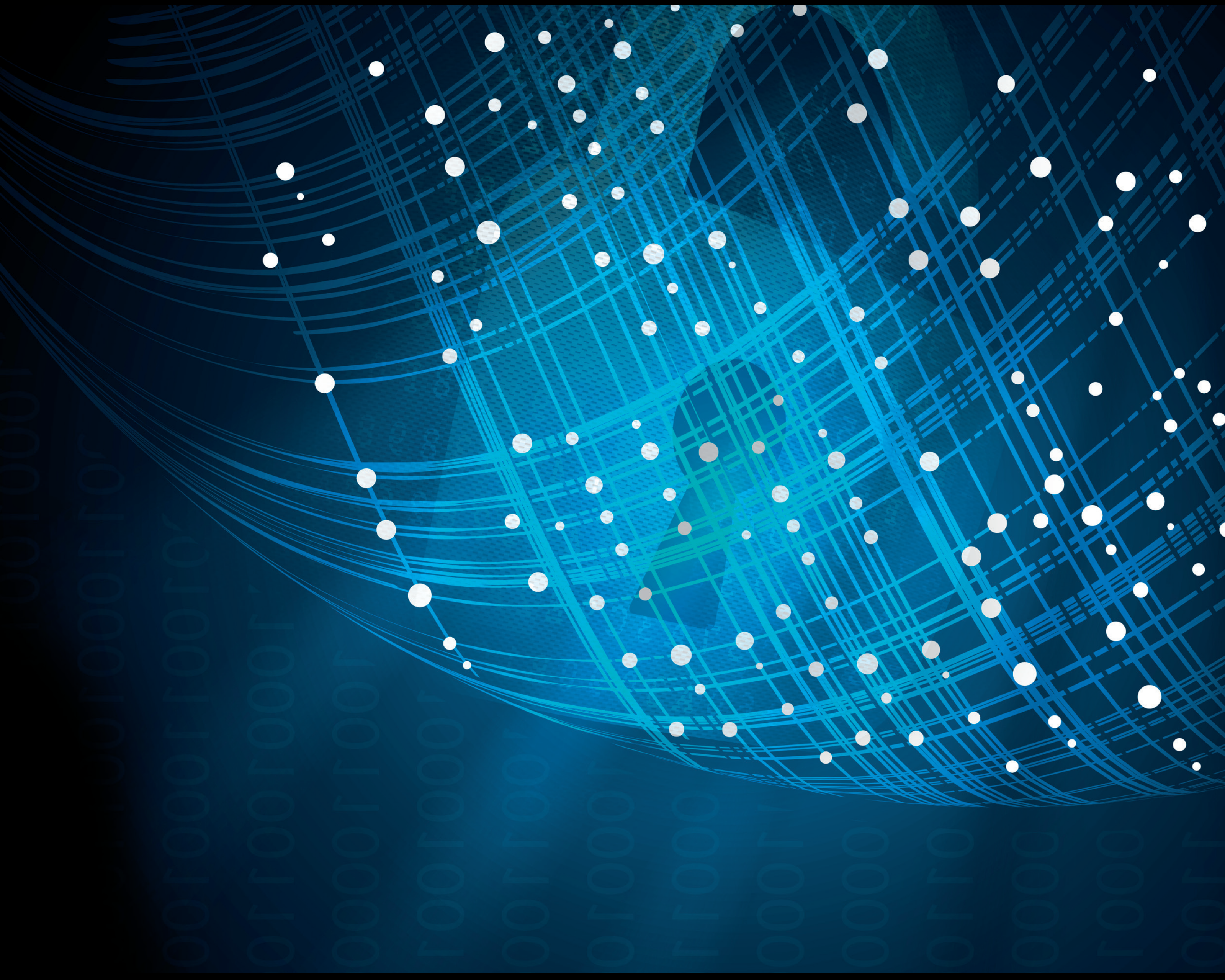
Security and Communication Networks

Security, Trust, and Privacy in Machine Learning and Internet of Things 2021

Lead Guest Editor: Weizhi Meng
Guest Editors: Jinguang Han, Wenjia Li, and Chunhua Su



Security, Trust, and Privacy in Machine Learning and Internet of Things 2021

Security and Communication Networks

Security, Trust, and Privacy in Machine Learning and Internet of Things 2021

Lead Guest Editor: Weizhi Meng
Guest Editors: Jinguang Han, Wenjia Li, and Chunhua Su

Contents

Hindawi
Security and Communication Networks
Volume 2022, Article ID 9896689, 17 pages
https://doi.org/10.1155/2022/9896689

Research Article

Zero-Trust-Based Protection Scheme for Users in Internet of Vehicles

Letian Fang,[1] **Chunshang Wu**,[2] **Yukun Kang**,[2] **Wei Ou**,[2] **Donghao Zhou**,[3] and **Jun Ye**[2]

[1]*School of Food Science and Engineering, Hainan University, Haikou, China*
[2]*School of Cyberspace Security, Hainan University, Haikou, China*
[3]*School of Computer, National University of Defense Technology, Changsha, Hunan, China*

Correspondence should be addressed to Wei Ou; ouwei@hainanu.edu.cn

Received 1 October 2021; Revised 28 November 2021; Accepted 21 April 2022; Published 12 May 2022

Academic Editor: Weizhi Meng

At present, the Internet of Vehicles technology is developing rapidly, but in the process of networking it may be attacked by hackers. In view of the tampering attack on the user's device and identity, we establish a multifactor authentication scheme to achieve dual identity authentication through device fingerprint authentication and PKI authentication. In response to the attack by hackers in the application of data transmission, we use the SM series of state secret algorithms for data encryption and take advantage of the immutable and decentralized characteristics of blockchain to ensure the security and integrity of data collection and transmission. Through the zero-trust security network architecture, the security level of the system in the process of data transmission is effectively improved, the identity-centered dynamic access control is carried out, the user's behavior is monitored in real time, and the malicious nodes are screened and eliminated, which improves the stability and security of the Internet of Vehicles data transmission system. By adopting the identity authentication pass rate, fingerprint authentication pass rate, API authentication pass rate, and SPA packet transmission acceptance rate as the factor set, the evaluation level of the vehicles is calculated. The experimental results show that compared with the traditional boundary-centered security protection, our scheme can protect a wider range of application security, even if there are security problems, the loss is less. Through the training simulation of the convolutional neural network, the classification accuracy of the trust level is improved.

1. Introduction

In recent years, with the rapid development of mobile Internet and industrial intelligence, the automotive industry is constantly changing to intelligence and networking. Internet of Vehicles has become an important research field. The Internet of Vehicles refers to the realization of an all-around network connection within vehicles, between vehicles and people, between vehicles and vehicles, between vehicles and roads, and between vehicles and service platforms with the help of a new generation of mobile communication technology [1]. The Internet of Vehicles improves the intelligence level and automatic driving ability of vehicles and brings great convenience to people's transportation. At the same time, it will help the government to establish an intelligent transportation system and build a new business form of automobile and transportation. Figure 1 shows the Internet of Vehicles communication scenario.

The Internet of Vehicles system contains massive private data of relevant users and vehicles. The cloud service platform of Internet of Vehicles includes key data such as vehicle management and transportation, service of information content, and personal information. Therefore, data security is an important issue of Internet of Vehicles. In the traditional Internet of Vehicles, the information of users and vehicles exposed in public is easily stolen, interfered, or even modified, as the Internet of Vehicles is a part of wireless communication. At present, the academic community believes that the threat to data security of Internet of Vehicles mainly comes from the following two aspects:

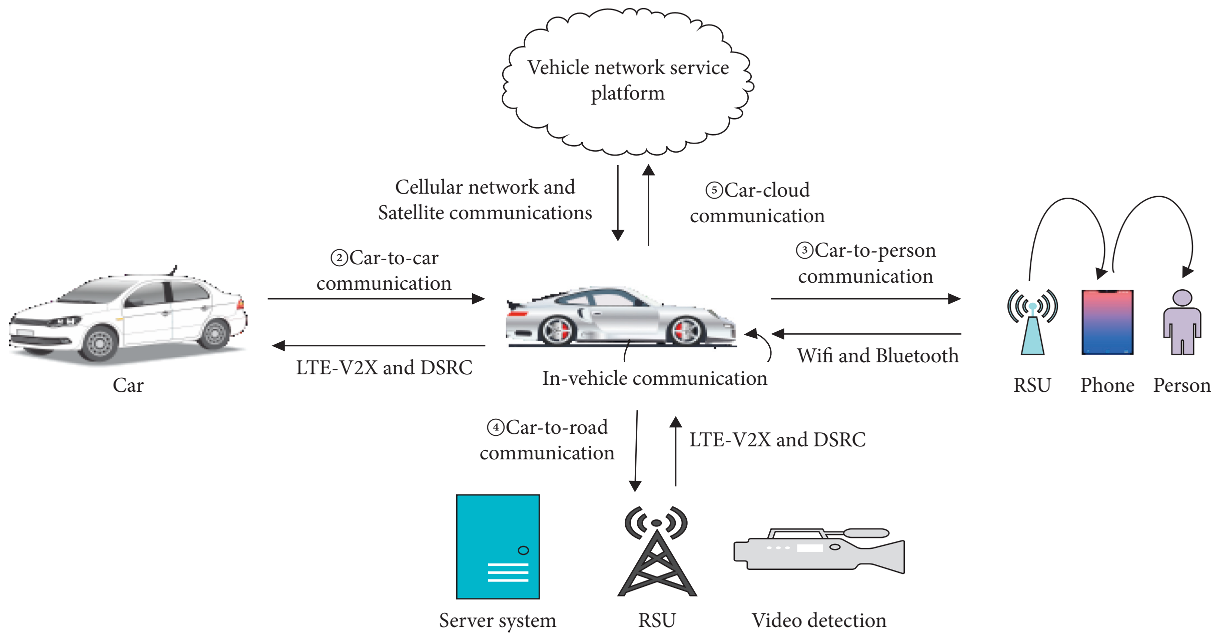

Vehicle network service platform
Cellular network and Satellite communications
⑤Car-cloud communication
②Car-to-car communication
LTE-V2X and DSRC
③Car-to-person communication
Wifi and Bluetooth
Car
In-vehicle communication
RSU
Phone
Person
④Car-to-road communication
LTE-V2X and DSRC
Server system
RSU
Video detection

Figure 1: Communication scenarios of vehicle networking.

(i) The attacker attacks the vehicle-linked network transmission. In the process of wireless communication, the attacker obtains the private data from users on the vehicle node in the communication channel by launching an eavesdropping attack. After the eavesdropping attack, the attacker can also coordinate a tampering attack, falsify the collected private data, and then send the wrong information to the users.

(ii) The attacker attacks the application side of the Internet of Vehicles. When the road traffic infrastructure RSU, traffic cameras, and other devices that receive communication information are captured by an attacker, a node capture attack will come. In addition, if the smart device related to the Internet of Vehicles application is lost, attackers may use the authentication information in them to have access to resources and services illegally. There are also attacks initiated by system insiders. For example, an administrator who has access to user passwords illegally embezzles and uses users' information, which is stored in the system server to launch attacks on the network. Such ghost-type attacks will also threaten the security of the Internet of Vehicles network. The safety problems are diversely faced by the Internet of Vehicles, and the defense solution we need to do should be comprehensive. It is the direction that we need to solve urgently to establish a three-dimensional and reliable security defense system for the Internet of Vehicles.

The protection measures of data security introduced in the existing research mainly focus on data encryption technology, data access control technology, corresponding integrity protection, data existence and availability proof, and virtualization security technology. The above technologies can ensure the security of user private data, to a certain extent. In the Internet of Vehicles environment, with the high-speed movement of vehicles, the network topology and the external environment of vehicles are constantly changing, and unknown network attacks will continue to emerge. However, the existing research is still based on traditional border security. By dividing the security area, the network is divided into an external network and an internal network. The traditional border security takes the defense in depth model as the center and carries out security protection by building a "wall." By default, the interior of the border is not secure, which cannot prevent malicious attacks launched by attackers from the interior, and there are great security risks. In the Internet of Vehicles, data are assets and have a high value. To deal with the shortcomings of traditional security protection, a new security concept focusing on data should be adopted. The zero-trust business security takes identity as the center and establishes a dynamic portable boundary. For the cloud PAN-interconnected vehicle networking system of people, vehicles, and roads, the zero-trust security system is more suitable. Compared with the traditional scheme, we take the lead in using the zero-trust security architecture in the Internet of Vehicles. While ensuring the security of data flow transmission of the Internet of Vehicles, the efficiency of system security identification is improved, and a new mode is used to manage the Internet of Vehicles. Therefore, we propose a zero-trust-based data protection scheme for users in the Internet of Vehicles.

The zero-trust security architecture adheres to the principle of "never trust and always verify." Its access control is based on the core of identity. The subject of data transmission will be continuously authenticated. Then, the zero trust will rely on the trust evaluation model to conduct an intelligent behavior analysis of the entire access process of the visitor, which can timely respond to the security threats of the Internet of Vehicles system data and update the vehicle nodes' trust value in real time. Finally, through the trusted access gateway, the control of the dynamic access authority of the asset equipment is realized. The zero trust will automatically resist eavesdropping attacks on the acquisition of private data in the communication channel and prevent malicious operations such as tampering attacks. Once a node is evaluated as a malicious node, it will no longer be able to access the database. At present, communication technology is adopted to realize the interconnection between vehicles, roads, and data processing platforms and perform cloud processing on data. In the blockchain network [2], the decentralized features of the blockchain can effectively prevent the excessive centralization of data in the cloud and reduce the risk of database attacks. After the data are stored in the block, the traceable feature of the blockchain data can trace historical records such as user's identity authentication and vehicle's device fingerprint. The timestamp in the block can resist replay attacks. After using the commercial cryptographies SM2, SM3, and SM4 to encrypt the data of the Internet of Vehicles, it is transmitted to the blockchain network [3, 4]. The identification of vehicles is realized by device fingerprint technology. The device fingerprint technology includes device fingerprint extraction and device fingerprint authentication. The equipment fingerprint extraction is to actively collect the characteristic information related to vehicles and generate a unique identifier for each vehicle according to the fingerprint generation algorithm. The device fingerprint authentication is an important part of the whole identity authentication process. When a vehicle accesses the cloud data, the cloud platform first calculates the device fingerprint of the vehicle and compares it with the existing fingerprint in the database to realize the identity authentication of the vehicle. Through the authentication center CA in the PKI system, the identity authentication and authorization of Internet of Vehicles users are carried out. The users can transmit on the alliance chain. This research will effectively protect the private information of Internet of Vehicles users [5].

In combination with the convolutional neural network, the vehicle and user identities are collected to extract behavioral features, and then the feature values are fuzzily calculated according to different weights to obtain the trust value. Finally, the user's trust is inputted into the model to achieve the prediction of trust level and improve the accuracy of trust level evaluation.

The rest of this study is organized as follows: in Section 2, we mainly introduce the research status of data security of the Internet of Vehicles at home and abroad. In Section 3, we mainly introduce the overall architecture and process of data protection scheme for Internet of Vehicles based on zero trust. In Section 4, we mainly introduce the simulation experiment of this scheme. In Section 5, we do a safety analysis. In Section 6, we mainly summarize the article.

2. Background

Zhang et al. [6] proposed a reliable and efficient system based on edge computing and blockchain, which is designed to ensure the reliability of edge devices during interactions and improve transmission efficiency. Mostafa et al. [7] proposed a layered adjustable autonomy (LAA) as a dynamically adjustable autonomy model for a multiagent system. Poongodi et al. [8] have proposed the improved versions of blockchain technology to strengthen various real-time complex applications at a flourishing rate. The results of the simulation kernel show that the proposed architecture justifies all the essential characteristics and effectuates the optimal use of 5G network sharing by each network entity. Kumar et al. [9] proposed an optimized location-aided routing protocol that is the modified version of location-aided routing protocol. Xiong et al. [10] proposed an efficient and large-scale batch verification scheme with group testing technology based on ECDSA, which analyzes the application of the presented protocols in Bitcoin and Hyperledger Fabric. Wang et al. [11] proposed PERT, a privacy-enhanced retrieval technology for cloud-assisted IoT, which preserves data privacy by hiding the information of data transmission between the cloud and the edge servers. Mostafa et al. [12] conducted tests on the AFC agent, and the results show that the agent successfully controls the UAV in three performed test cases and a total of nine implemented missions. Zhang et al. [13] proposed a covert communication model combined with smart contracts to covertly transfer information in the blockchain environment, using encryption algorithms and two-round protocols to ensure data privacy. Wang et al. [14] proposed a new certificateless scheme, which utilizes the most advanced blockchain technology and smart contracts to build a reliable and efficient CLS scheme. Lian et al. [15] proposed a new joint learning system, COFEL, which can reduce communication time through layer-based parameter selection and enhance privacy protection by using a local differential privacy mechanism for selected parameters. Zhang et al. [16] proposed a secure and efficient data storage and sharing scheme based on blockchain-based mobile-edge computing. IoT devices only need to submit the data and random key share allocated to edge nodes, and edge nodes use the recovered signature private key to realize data signature and homomorphic encryption. Song et al. [17] proposed a security arrangement method based on matrix eigenvalue calculation. Compared with existing security arrangement methods, this method has stronger robustness and efficiency, making the scheme more suitable for repeated polymerization. Balamurugan et al. [18] proposed a subspace tracking algorithm with low computational complexity for tracking DOA to provide a seamless connection. Compared with traditional DOA estimation methods, the proposed DOA tracking method takes less time to track the current position of UAV target, and the tracking process is not affected by a signal-to-noise ratio.

Wang [19] put forward a nondual pair authentication scheme for Internet of Vehicles messages based on elliptic curve cryptography. This scheme adopts random pseudonyms to achieve conditional privacy protection. Xie et al. [20] came up with an improved secure certificateless aggregation authentication scheme based on elliptic cryptographic curves. Li et al. [21] proposed a password-based serverless cross-domain vehicle-to-vehicle authentication and key agreement protocol. Xin et al. [22] proposed an event-driven lightweight algorithm to quickly identify the false position of vehicles and detect the erroneous behavior claiming the false position. Shi and Wang [23] put forward a resist conspiracy Sybil attack detection method based on spatiotemporal analysis based resist conspiracy Sybil (STARCS) attack. Si [24] proposed a lightweight authentication scheme suitable for vehicle-mounted self-organizing networks. The trusted authority (TA) obtains the identification information of the onboard unit (OBU) through calculation, which can significantly improve the randomness of the signature of the message and avoid the counterfeit attack, hence improving the security of the solution. Zhang [25] proposed a 5G vehicular network authentication scheme based on a reputation system. A one-way hash function is adopted to generate the credit references to restrict the vehicles whose reputation score is below the threshold from participating in the authentication process. Zhang [26] constructed the vehicular cloud computing structure formed by the collaboration among vehicles and designed a security authentication mechanism that can achieve a privacy protection based on an identity-based signcryption scheme and a short-group signature scheme. Chen et al. [27] proposed an E-forensics framework of Internet of Vehicles based on the blockchain technology. It implements a remote repository of E-forensics by using the features of decentralized storage for blockchain technology. Ma [28] proposed a trust management model based on the blockchain to realize the reliability and synchronization of existing reputation data. In trust calculation, the RSU relies on a weighted voting mechanism to evaluate the credibility of the message. Xu [29] studied the vehicle attribute recognition technology in surveillance video and bayonet image, mainly studied the fine-grained vehicle type recognition and body color recognition using the convolution neural network technology framework, and developed the vehicle attribute recognition prototype system on this basis.

3. Scheme

3.1. Architecture. We mainly describe the capacities of overall architecture in the zero-trust philosophy, including the following capabilities: zero-trust drive authentication mechanism, identity security infrastructure, and trusted access gateway. The authentication mechanism of zero trust is the core capability of the entire program architecture. The identity security infrastructure implements trusted access by establishing trusted basic permissions between trusted access subjects and trusted access objects. The trusted access gateway supports single sign-on (SSO) to avoid frequent authentication and improve ease of use. First, based on the user's access behavior to the device and the access environment of the device and then relying on the continuous trust evaluation model and the access control model, the intelligent behavior analysis of the entire access process of the visitor is carried out. Then, the risk coefficient of the visitor is intelligently adjusted, and then a continuous trust assessment of the credibility of users and asset equipment is achieved. The following step is to dynamically adjust access control strategies based on the evaluation results. Finally, the dynamic access authority control of the mobile subject is realized through the trusted access gateway, as shown in Figure 2.

There are varieties of mobile subjects in the Internet of Vehicles, including equipment hardware parameters, network environment, intelligent systems, sensors, signals, user information, etc. We can regard these access subjects as the attribute set of the application. Based on the device information of multidimensional mobile subjects, such as device hardware parameters, network environment, intelligent systems, sensors, and signals, a unique device identifier called device fingerprint is generated through a model algorithm. The device fingerprint is stable and does not change even when the device system is upgraded or parameters are changed. In addition, device fingerprints cannot be tampered and can identify the risks of terminal environment. Generation rules of the device fingerprint are placed in the cloud platform. During the registration process of the device management, the device fingerprint is automatically generated according to the device parameters and then it will be stored in the device fingerprint library, as shown in Figure 3. In the process of device login authentication, based on the collected multidimensional device information, the model algorithm of the cloud platform analyzes the collected data and calculates the device fingerprint, matching it with the device fingerprint library to achieve enhanced authentication.

Through continuous active scanning, passive detection, and secure access to the control area, continuous trust evaluation and access control are carried out on the vehicle IoT terminal to solve the counterfeiting and malicious access. High-confidence equipment adopts a trusted chip + - trusted OS56 to directly identify the identity. Embedded devices adopt device tags, such as mobile device identification code, application developer identifier, unique device identification code, RFID6 electronic tags, and cryptographic modules attached to the outside of the device, all of which can help establish device identity. For low-intelligence IoT devices, digital fingerprints can also be used to construct device identification to solve the problem involved in device access identity management. Having an established fingerprint database improves the accuracy of zero-trust architecture devices. Since each user has only one unique identity fingerprint, when it is employed, any changes to the identity fingerprint will be dynamically detected by the trust engine, which leads to the engine's restricting permissions of the related user.

Under the zero-trust framework, during the communication between the car and the cloud, commercial cryptography ensures the security of the system. The two-way identity authentication between the car and the cloud realizes the anti-interception, anticounterfeiting, and antireuse of the authentication information, ensuring the authenticity of the users and the vehicle management cloud platform. In terms of security transmission requirements, the system must ensure the confidentiality and integrity of the communication process. The vehicle can realize the communication function with the cloud server through the vehicle

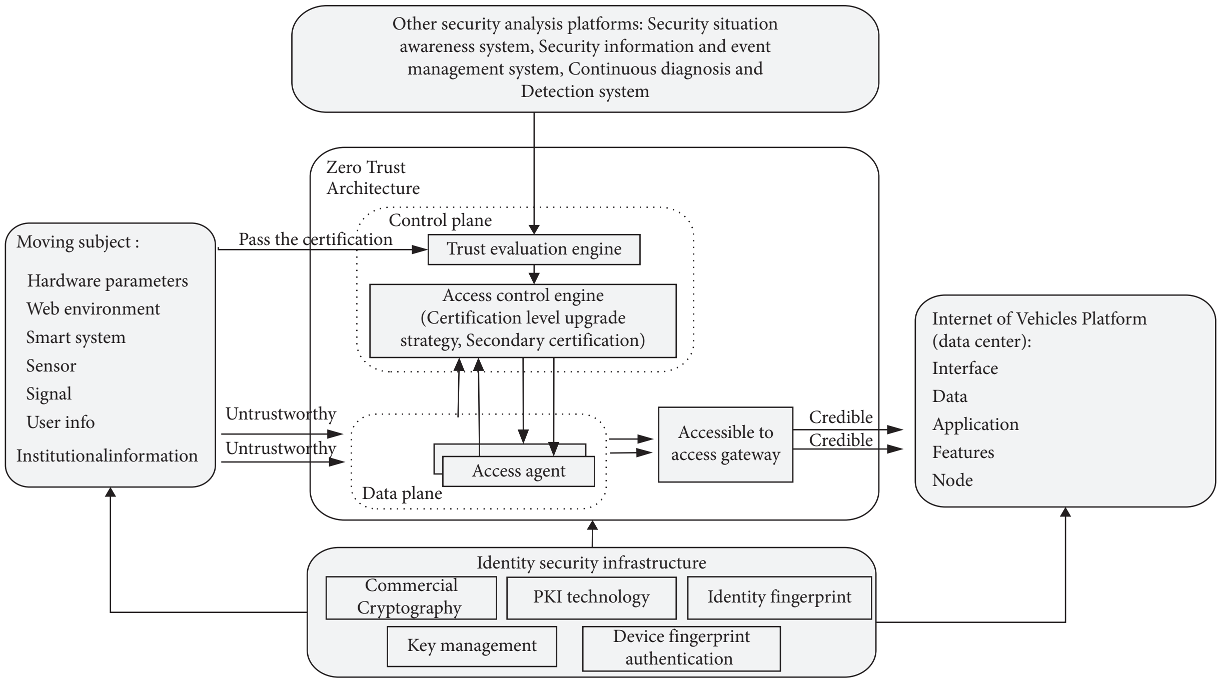

Other security analysis platforms: Security situation awareness system, Security information and event management system, Continuous diagnosis and Detection system
Zero Trust Architecture
Control plane
Trust evaluation engine
Access control engine (Certification level upgrade strategy, Secondary certification)
Moving subject :
Hardware parameters
Web environment
Smart system
Sensor
Signal
User info
Institutionalinformation
Pass the certification
Untrustworthy
Untrustworthy
Access agent
Data plane
Accessible to access gateway
Credible
Credible
Internet of Vehicles Platform (data center):
Interface
Data
Application
Features
Node
Identity security infrastructure
Commercial Cryptography
PKI technology
Identity fingerprint
Key management
Device fingerprint authentication

Figure 2: Structure of zero-trust-based data protection system for the Internet of Vehicles.

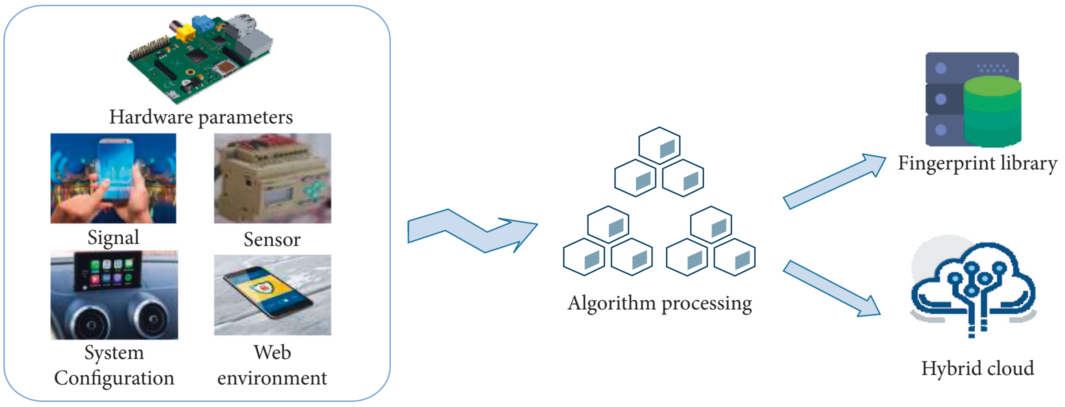


Figure 3: Verification process of device fingerprint library.

terminals and the Bluetooth communication modules. The vehicle realizes various security functions by configuring the cryptographic module. Through the encryption and verification of SM2, SM3, and SM4 commercial cryptography, the information security of user registration will be improved, and the transmission of vehicle data will be difficult to be tampered with and disclosed.

In the cloud server, the server cryptography machine performs authentication operations using the public key stored in it. In the Internet of Vehicles system, an SM2 key protocol algorithm is adopted to negotiate a session key, and secure transmission of sensitive data between the cloud server and the vehicle is completed. In the Internet of Vehicles system, the cloud server employs the server cipher machine that adopts SM4 and HMAC-SM3 algorithms to protect the confidentiality and integrity of user data and authentication data. At the same time, the vehicle calls cryptography modules that apply SM4 and HMAC-SM3 algorithms to protect the confidentiality and integrity of key data such as identification data, vehicle collection, and control data.

In the Internet of Vehicles system, the PKI is widely applied and plays different roles among different objects. In the Internet of Vehicles system, the application and related data security of the Internet of Vehicles include a four-layer key system: CA public keys, the cloud server key pair, the car Bluetooth communication module key pair, and the Bluetooth communication key, all of which are all key components of the PKI in the identity security infrastructure. PKI can guarantee the authenticity, integrity, confidentiality, and nonrepudiation of the identities of both parties. The source of trust of the asymmetric key system is the CA certificate, which is used to verify the cloud server certificate and the car Bluetooth communication module certificate. In the zero-trust framework, the signature key of the cloud server key pair is used to authenticate the identity of the cloud server and the encryption key is used to realize the secure transmission of data between the server and the car. The public key is issued by the CA to form a cloud server certificate. The private key is stored in the cipher machine of the cloud server. The public key is issued by the CA to form the car Bluetooth communication module certificate, while the private key is stored in the car Bluetooth cryptographic module.

In the Internet of Vehicles system, the zero-trust framework can ensure the safety of the whole system. It contains two parts: the control plane and the data plane. In the control plane, the trust evaluation engine and access control engine are connected to form effective contact and communication with each module. On the one hand, the trust evaluation engine receives node devices, people, applications, systems, and directly connected devices. On the other hand, it maintains contact with the data center, including Internet of Vehicles cloud systems. At the same time, the trust evaluation engine can share data with other security analysis platforms and also transmit information with the access control engine. The access agent is the key node of zero-trust architecture for authentication, can efficiently and effectively verify the information received by the zero-trust system each time, and ensure the security of the entire Internet of Vehicles system.

The data center of the Internet of Vehicles platform will be built on the blockchain network and will be deployed in the cloud server. The center will include a user interface, data visualization processing, application services, and basic user functions. The whole system takes the vehicle equipment in the connected car environment and users' system as blockchain network nodes, to build a decentralized chain alliance. It decides to charge to an account through consultation and achieve consistency by the consensus mechanism. Therefore, members of the nodes will achieve data exchange in the case of not fully trusting. Members can enter or exit the league chain only through the authorization of these organizations. Important data such as the node's public key information, the node's historical communication behavior data, and smart contract-based access control strategy are stored in the blockchain, and all blockchain nodes jointly maintain the communication security among the nodes of the Internet of Vehicles. However, the nodes of the Internet of Vehicles usually have limited computing and storage resources. Therefore, the vehicle node is regarded as a lightweight node of the blockchain, which only stores the block header of the blockchain but does not participate in "mining." To improve the speed of the blockchain network, the communication behavior data of the vehicle nodes are collected by its adjacent RSVs, while the collection process is performed by predefined smart contracts. Round-side units are authorized as full nodes of the blockchain to generate and validate new blocks, while vehicle-borne units, as lightweight nodes of the blockchain, do not participate in the "mining" process of the blockchain, thus causing no additional overhead to the vehicle nodes.

In the process of evaluating vehicle trust levels, we introduce a trust level classification model based on the convolution neural network algorithm. The preset convolution neural network is used to train the data of vehicle nodes and complete the trust level classification of vehicle nodes. A CNN model has four typical characteristics: local connection, weight sharing, pooling operation, and multilayer structure. The CNN can automatically learn features from data through multilayer nonlinear transformation, to replace manually designed features, and its deep structure makes it have strong expression and learning ability. The CNN has a special structure of weight sharing, and its layout is closer to the actual biological neural network, which reduces the complexity of the network. In particular, the image of a multidimensional input vector can be directly input into the network, which avoids the complexity of data reconstruction in the process of feature extraction and classification. As an input-output mapping, the CNN can learn a large number of mapping relationships between input and output. Without the precise mathematical expression between input and output, the convolutional neural network can have the ability of mapping between input and output pairs by using the known pattern to train the convolutional neural network. Combined with the CNN model, the efficiency of the algorithm in the cluster environment has been greatly improved. It can efficiently process the experimental data and complete the effective classification of the trust value of vehicle nodes.

3.2. Work Flow. During the process of trust evaluation in the equipment of the vehicle network, we should not only authenticate the static attributes of the device, such as its attribute, ID, and network environment, but also evaluate the dynamic attributes of the device, such as its real-time request, dynamic operation, and evaluation success rate. In particular, it is very important to calculate and update the trust value of the dynamic attributes of each node. In the zero-trust network architecture, the initial trust value of the device is 0, so we need to collect and authenticate the static attributes of the device. First of all, we will collect the identity fingerprints of users of the Internet of Vehicles and store the fingerprints in the cloud server to verify and authenticate the identity of users of the Internet of Vehicles by using the identity fingerprint database. At the same time, the digital certificate issued by a Certification Authority (CA) of public key infrastructure (PKI) technology is used for verification, together with identity fingerprint database authentication as a way to achieve the dual authentication. Therefore, the credibility and security of the device can be improved. Figure 4 shows the overall process.

3.2.1. Identity Registration. When the device on the vehicle needs to communicate with the edge device, it must first apply for registration with a trusted vehicle management center. The specific registration process is as follows:

First, the vehicle management center chooses N random numbers $C_1, C_2, \ldots, C_N$, to form the challenge set $C = \{C_1, C_2, \ldots, C_N\}$, and then sends C to the onboard equipment.

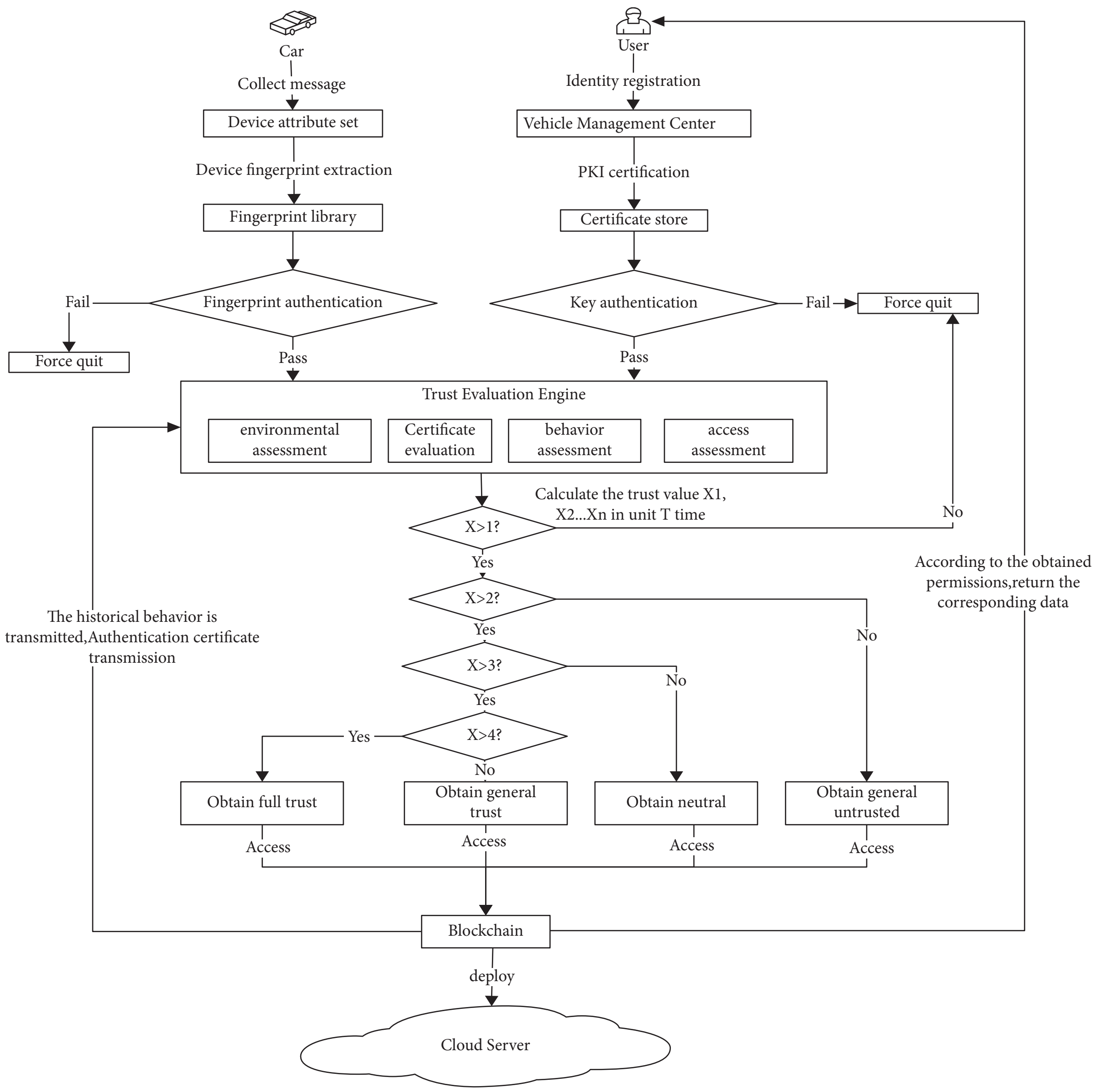


Figure 4: The work flow.

Second, after receiving C, the onboard device calculates the PUF function $R_i = \mathrm{PUF}_{\mathrm{Vi}}\ (C_i)$, obtains $R = \{R_1, R_2, \ldots, R_n\}$, and sends R to the vehicle management center through the secure channel.

Third, the train federation management center generates a pseudonym AID_i for the received R and obtains $\mathrm{AID} = \{\mathrm{AID}_1, \mathrm{AID}_2, \ldots, \mathrm{AID}_\mathrm{N}\}$. The $\{\mathrm{ID}_{\mathrm{Vi}}, \mathrm{AID}, R\}$ is stored in a secure storage area, and then the {AID, R} is sent to the blockchain network. Then, the unused list is written by the PoS consensus algorithm, and the AID is finally sent to the vehicle device.

After the onboard device receives the AID, it will be stored in the safe storage area with the corresponding C.

The advantage of this management mode is that it can guarantee effective communication between vehicles and edge nodes, improve the information processing ability of the whole system, and improve the efficiency of information communication. Identity registration plays a key role in system information processing. Identity identification is the basis of subsequent identity authentication to ensure an efficient and orderly information system.

3.2.2. Device Fingerprint Extraction. For common Internet of Vehicles devices, fingerprint information is particularly important for identity confirmation. The device fingerprint ensures the accuracy and efficiency of the whole data transmission process, avoids the tedious verification process of messages in the information transmission, and ensures the stable operation of the whole system. When the data of the device are transmitted remotely, the unique fingerprint information of the device is used as favorable evidence of its authentication. Device fingerprint extraction provides the basis for subsequent device fingerprint identification, which is the first step of system identity operation.

We adopt an active device fingerprint technology to obtain the fingerprint of vehicle equipment. The vehicle equipment hardware, network environment, sensors, signals, system control, and other parameters are obtained by calling the SDK interface, then the equipment fingerprint is generated by using the hash algorithm, and the equipment fingerprint database is built based on certain principles. Without relying on the sensitive authority of the device, two types of identifiers for generating device fingerprints can be collected through the browser platform of device access management. For different IoT devices, the fingerprint parameters can be obtained through the development documents provided by the device manufacturer.

The process of collecting and generating the parameter information of the device fingerprint by calling the SDK interface fully meets the following two requirements, so that the whole process of collecting the device message is transparent and visible and does not interfere with the normal use of the devices:

(i) *Fast Response*. By default, the SDK has a timeout of 3 seconds for establishing a connection. It has a fast response speed and will not fail to obtain fingerprint information because the device access time is too short.

(ii) *Less Resource Consumption*. Calling the SDK interface to extract device information will not occupy too much bandwidth, memory, CPU, and other resources and will not have any impact on the normal operation of the device.

3.2.3. Identity Authentication

(1) PKI Certification. Confirm the identity through PKI's CA certificate management center. In the Internet of Vehicles, users use their identity information to generate digital signatures, which together form CA. In addition to user information, a digital signature includes the name of the certificate authority, certificate validity period, certificate serial number, the hash algorithm used for the signature, and encryption algorithm used for the public key. We can view the certificate validity period to determine the validity of the CA. The vehicle sends its own information to the relevant user, and the user uses the CA's public key to verify the signature of the certificate. As the CA is the only issuer of the certificate, the user can verify the authenticity of the certificate in this way, and the user can use the public key to verify the signature of the vehicle or carry out encrypted communication with the vehicle. At the same time, to prevent the authenticity of the public key used by the user when verifying the CA certificate of the vehicle, we can find another certificate authority to issue a certificate to the public key of the certificate authority. In this way, a nested loop of public key certificate is formed, and the end of the loop is the Root Certificate Authority. The public key certificate nesting cycle can ensure the authenticity of the public key in the communication between the vehicle and the user in the Internet of Vehicles and ensure the overall security of information. It can be concluded that PKI plays a very important role in the whole process. The circular nesting of certificates ensures the security of public keys used in data encryption and improves the efficiency of system authentication. Compared with the traditional vehicle network system, this system has certain advantages in encryption authentication.

(2) Authentication of Device Fingerprint. Equipment fingerprint authentication is a very important identity authentication technology in the Internet of Vehicles system. In the authentication process, the client sends an access request to collect terminal features, and the ECS loads the feature collection code for collection. The server will locate the specific device recorded in the system according to the MAC address, verifying the active fingerprint of the device and judging whether it is a new device. If it is a new device, the new fingerprint will be stored in the device fingerprint database. If the authentication is passed, it means that the current equipment has passed the authentication and can continue to communicate with the system. Otherwise, the system will actively disconnect.

3.2.4. Trust Evaluation.

We built three vehicles in the system and carried out experiments independently. The experiment carried out by the two vehicles is applied as a control experiment and the accuracy of the trust evaluation is verified through multiple experiments. We will describe the process of trust evaluation between the vehicle and the car cloud system by the following steps:

(1) Pass Rate of Client Authentication. First, apply PKI's CA certificate for client authentication as the initial authentication method. Issue the certificate to the authenticated client through the CA certificate management center of PKI. The client authentication process is shown in Figure 5. The client authentication module judges whether the machine logs in locally, whether the access address is abnormal, whether the account password is changed, whether the protocol is changed, etc., to determine whether the client authentication of the vehicle management center is successful. Assign different initial trust values to vehicles that have passed different authentication factors.

(2) Pass Rate of Fingerprint Authentication. As shown in Figure 6 we obtain the fingerprint features of the vehicle equipment including hardware parameters, network environment, intelligent systems, sensors, signals, and other information by calling the SDK interface and then use the algorithm to generate the device fingerprint and build the device fingerprint library based on certain rules. The newly generated device fingerprint is identified in the fingerprint database. We mainly extracted some basic parameters of terminal equipment. If the fingerprint is associated successfully, the device passes the fingerprint identification process. According to the difference in fingerprints of each vehicle, different trust values will be obtained. The trust value obtained by the fingerprint authentication of the three vehicles will be used as in the follow-up trust evaluation.

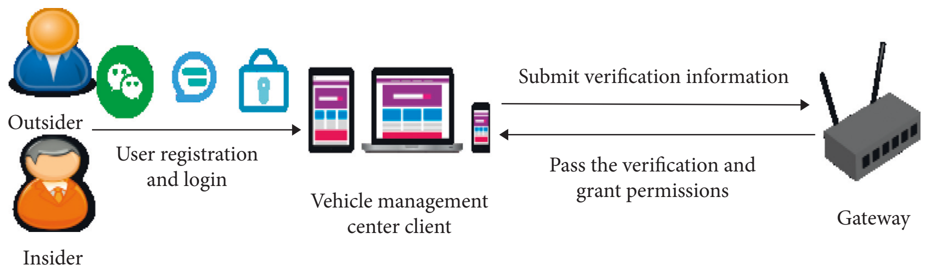

Outsider
Insider
User registration and login
Vehicle management center client
Submit verification information
Pass the verification and grant permissions
Gateway

Figure 5: Client authentication.

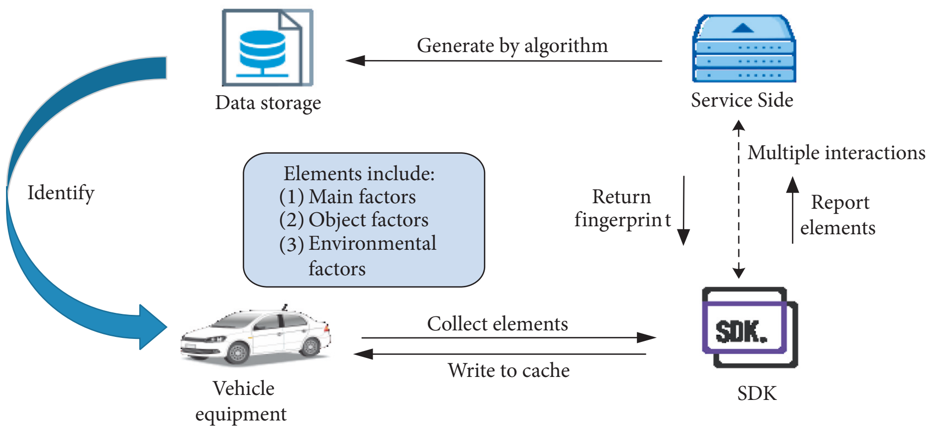


Figure 6: Device fingerprint extraction.

(3) API Certification Pass Rate. The use of API gateway can reduce the attack surface and concentrate resource advantages. The API gateway is based on a defense-in-depth strategy and has the function of authentication. Even if the attacker breaks the API gateway, it still needs to further break the internal service authentication to enter a single service. Next, the microisolation test of the system will be carried out on the three vehicles to verify the proxy and hiding conditions applied by different modules of the system after the installation of the access gateway plug-in, as shown in Figure 7. In the specific test of the system, the API gateway of vehicle deployment is verified, the specific steps of the implementation phase are evaluated, and the verification and call success rate are taken as key evaluation values. Three vehicles are verified by experiment in parallel. Due to the different situations of vehicle API verification and call, three vehicles will produce different trust values, which will be used as the basis for subsequent trust evaluation.

(4) SPA Packet Transmission Acceptance Rate of SDP Architecture. After completed the microisolation test, the next step is to validate the SPA single-pack authorization for the three vehicles. The client carries the encrypted SPA packet and sends an access request to the gateway. The gateway decrypts the packet using the key provided by the controller and cross-checks the information in the decrypted packet with the information it receives from the controller to determine whether the client can access the packet, as shown in Figure 8. After the vehicles have completed the SPA single package authorization, the trust evaluation is conducted based on whether the SPA package is sent successfully and whether the certificate verification is passed.

When the device ed_i conducts identity authentication, it needs to conduct identity authentication at the edge computing layer.

By sending authentication requests to node es_1 of the edge computing layer, the corresponding key is obtained from PKG and applied to the key communication of the session. The request sent by the device contains the identity of the device and the node of the edge layer. The SM2 signature algorithm is used to calculate the corresponding digital signature (h, s) by using the private key of the device.

$$\mathrm{ed}_i \longrightarrow \mathrm{es}_1\text{: AccessReq} \,\|\, N_1 \,\|\, \mathrm{ed}_i \,\|\, \mathrm{es}_1 \,\|\, h \,\|\, S. \tag{1}$$

AccessReq is the authentication request when the device accesses, and N_1 is a random number.

After receiving the authentication request of the terminal device, the edge computing layer uses the SM2 signature algorithm to sign and authenticate it. After verification, the edge computing layer node saves the identity information of the terminal device to the authentication list and gives feedback of the encrypted authentication information to the terminal device node. First of all, the value of the Q_D element in group G_1 is calculated according to formula (2), and the Cipher values in group G_1 are calculated according to the generated random number r. Ciphers are the ciphertext and $r \in [1, N-1]$, the encapsulated key Key is calculated by KDF according to formula (3), and the key value is the shared key of the edge layer and the terminal device:

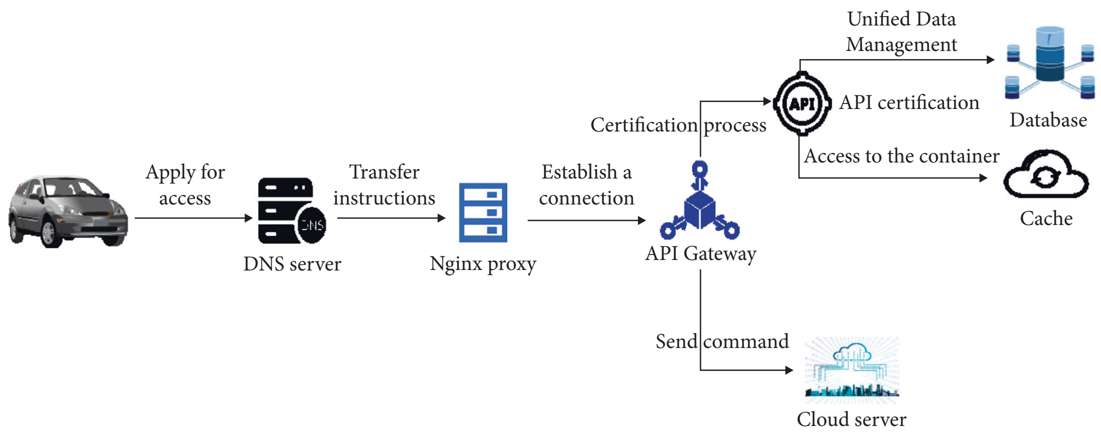


Figure 7: API gateway authentication.

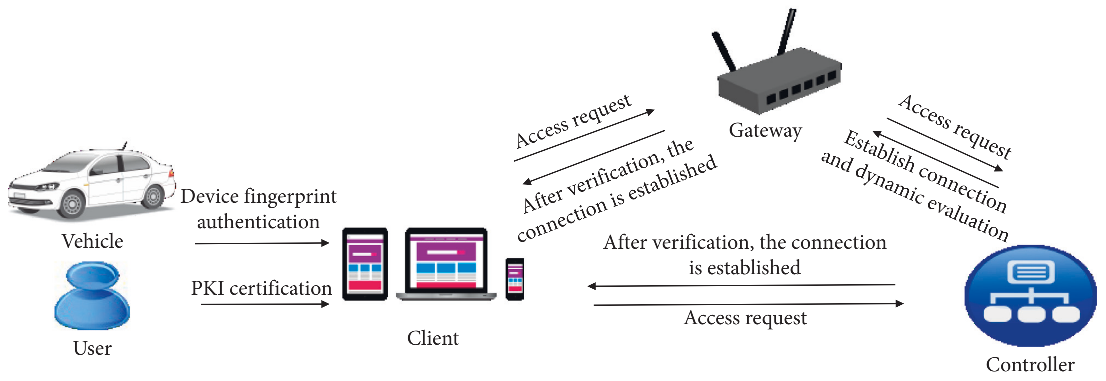

Gateway
Access request
After verification, the connection is established
Access request
Establish connection and dynamic evaluation
Vehicle
Device fingerprint authentication
PKI certification
User
Client
After verification, the connection is established
Access request
Controller

Figure 8: Single-packet authorization process.

$$\begin{aligned}
&\mathrm{QD} = [H_1(\mathrm{ed}_i \| \mathrm{hid}, N)]P_1 + P_{\mathrm{pub}-e},\\
&\mathrm{Key} = \mathrm{KDF}\Big(\mathrm{Cipher}\Big\|\big(P_{\mathrm{pub}-e}, P2\big)\Big\|D, \mathrm{Klen}\Big),\\
&\mathrm{es}_1 \longrightarrow \mathrm{ed}_i\colon\ \mathrm{AccessRsp}\|\,\mathrm{es}_1\,\|\,\mathrm{Cipher}\|\,h\|\,S,
\end{aligned} \tag{2}$$

where klen is the key length and AccessRSP requests response identification.

After receiving the response package of the edge layer node, the edge terminal device parses the received Cipher to obtain the corresponding key Key. First, it needs to determine whether the Cipher belongs to the element in G_1. If not, the input is 0; otherwise, w' in G_T is calculated according to formula (5). Then, SM2 converts the data into a bitstream to calculate the Key of the response:

$$\begin{aligned}
&w = e\big(\mathrm{Cipher}, d_{\mathrm{ed}_1}\big),\\
&\mathrm{Key} = \mathrm{KDF}\big(\mathrm{Cipher}\|w'\|\mathrm{ed}_i, \mathrm{klen}\big),\\
&\mathrm{ed}_i \longrightarrow \mathrm{es}_1\colon\ \mathrm{AccessAck}\|\,\mathrm{key}\,(\mathrm{ed}_i\|\,\mathrm{es}_1),
\end{aligned} \tag{3}$$

where AccessAck confirms the response and d_{ed1} is the private key of the terminal.

In summary, if the obtained key is 0, the identity authentication fails; otherwise, the identity authentication of the device is successful, and the identity of the device is saved to the node of the computing layer.

After evaluating CA's certificate level and verifying device identity, with the traceable and tamper-free feature of the blockchain network, we record all identity fingerprints, digital certificates, and verification information and save them in the block. By using a smart contract, automatic verification can be realized when the next device is accessed. However, if the node's trust value falls below the threshold or is deemed to be a malicious node, the smart contract will remove the digital certificate of the Internet of Vehicles users from the list. At the same time, the zero-trust system will also conduct historical behavior evaluations. The zero-trust trust evaluation engine will continuously collect the behavior characteristics of the vehicle and judge whether the vehicle node is a malicious node in combination with the historical behavior. It is an important factor in the evaluation of vehicle trust level. In this process, all noncompliance and malicious operations will be recorded in the blockchain network and will be used in the evaluation of the next cycle as a reference.

So far, we have completed some tasks of the engine in terms of trust evaluation in the zero-trust system. Next, we will proceed with the work of the access control engine. In the process of trust evaluation, since the degree of trust between vehicles is not a white and black exclusive relationship, it cannot be mechanically divided into trust and distrust. For example, the trust between vehicles can be described as "complete trust," "general trust," "neutrality," "general distrust," and "complete distrust." These trust descriptions can be used as the evaluation set of fuzzy evaluation process $D=$ {complete trust, general trust, neutrality, general distrust, complete distrust}. The cloud platform will set different access permissions according to the evaluation level of the vehicle. The higher the trust level of the node, the higher the permissions can be obtained.

The main factors that affect the evaluation are considered to determine the fuzzy evaluation process factor set $U=$ {client authentication pass rate, fingerprint authentication pass rate, API authentication success rate, SPA packet transmission-reception rate}. At the same time, according to the importance of the evaluation index of each factor in the factor set, the weight is calculated by the Delphi method, and the weight distribution set $W=\{0.4, 0.2, 0.2, 0.2\}$ is obtained. The cloud server will access and operate the device according to the evaluation level. In the process of data transmission, we will use SM2 to encrypt data to further ensure the safety and integrity of data transmission. Suppose the threshold of different permissions is V_n, $5>V_n\geq 0$, the access control engine divides the security levels of device into five levels: complete trust ($5>X\geq 4$), general trust ($4>X\geq 3$), neutrality($3>X\geq 2$), general distrust ($2>X\geq 1$), and complete distrust ($1>X\geq 0$), as listed in Table 1.

The zero-trust architecture will conduct a real-time evaluation of vehicle nodes, combined with the above four passing rates and the historical behavior of vehicles, and calculate the trust value X in the process of operation. The fuzzy mapping of several attributes of vehicle trust evaluation on the evaluation set is carried out to obtain the comprehensive evaluation matrix R. The proportion of each weight factor in each evaluation is obtained by calculating the vector $T=W*R$. Finally, the trust value X is calculated. The calculation formula is as follows:

$$X = T \times \begin{pmatrix} \dfrac{V0+V1}{2} \\ \dfrac{V1+V2}{2} \\ \dfrac{V2+V3}{2} \\ \dfrac{V3+V4}{2} \\ \dfrac{V4+V5}{2} \end{pmatrix}. \tag{4}$$

Within a monitoring cycle T, we will continuously evaluate the operating performance of the vehicle terminal, which is evaluated for n times in total, and obtain a series of trust values $X_1, X_2, \ldots, X_n$.

Assuming that the initial user's trust value is X_0, $X_0>V_n$, the vehicle terminal obtains the corresponding access and operation permissions; otherwise, the operation is blocked.

Calculate the variance $\sigma(X)^2$ of the trust value of each step within m cycles and take the mean value of the variance $\sigma(X)^2$. If $\overline{\sigma}^2>\sigma(X)^2$, the user's access level will be lowered. At this time, the user trust value is changed from X_m to the low-level permission until it is reduced to 0, and the user is forbidden to access.

If $\overline{\sigma}^2<\sigma(X)^2$, the user's access level will be increased, and the user's X_m trust value will change to a higher level of authority. The longer the operation cycle is, the more obvious the change of the trust value will be.

In the evaluation process of the convolutional neural network, the two classification authentication model is established to analyze the authentication results of users and vehicles. The results listed in Table 2 usually appear in the identification.

TP indicates the number of samples, in which all positive samples are classified correctly in the result. TN indicates the number of samples, in which all negative samples are classified correctly in the result. FN represents the number of samples, in which all positive samples are classified incorrectly in the result. FP indicates the number of samples, in which all negative samples are classified incorrectly in the result.

We used three indicators to assist in the analysis: precision, recall, and F1 score. We hope to obtain the comprehensive performance of the classifier in all categories. Therefore, it mainly focuses on the microaveraging F1 value as the measurement standard; that is, it calculates TP, FP, and FN for each group in the category set and completes the cumulative calculation of P, R, and F1. Their calculation formula is as follows:

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{(\text{TP}+\text{FP})}, \\ \text{Recall} &= \frac{\text{TP}}{(\text{TP}+\text{FN})}, \\ F_1 &= \frac{2\times \text{precision}\times \text{racall}}{\text{precision}+\text{recall}}. \end{aligned} \tag{5}$$

Under different thresholds, the accuracy rate and recall rate are often negatively correlated. In order to balance the overall evaluation effect of the two, the F1 score is calculated as the harmonic average of the two.

In the evaluation, we first need to find the weight convergence point of the CNN and then calculate the trust value of vehicle nodes with different trust levels. With the learning function of CNN, the weight convergence point is taken as an evaluation point. Once the network trust value to be classified is inputted at the input end, the weight coefficient in the network will be determined and finally reach the stable state after a process from initial to steady-state convergence. The trust level of vehicle nodes is the corresponding classification level. The specific CNN evaluation algorithm is as follows:

The trust value of the vehicle node to be rated is taken as the input value, and the predicted trust level of the vehicle node is taken as the output value. The CNN model will obtain the trust value of the sample vehicle node in the Internet of Vehicles.

Table 1: Vehicle rating level and authority.

Level	Estimation scale	Authorization
1	Complete trust	Vehicles and users can access, edit all data, and modify cloud facilities
2	General trust	Vehicles and users can access and edit all data
3	Neutrality	Vehicles and users can access all data
4	General distrust	Vehicles and users can access some data
5	Complete distrust	Refuse to provide service

Table 2: Classification results of binary classification model.

	Postive forecast	Negative forecast
True positives	TP (true positives)	FN (false negatives)
True negatives	FP (false positives)	TN (true negatives)

First, the model finds the convergence point through the given sample of node trust level. According to the set trust value rating rules, it matches the trust value levels of vehicle nodes treated separately and classified. Then, it creates a convolutional neural network (CNN) and sets the initial state value of the neural network. It takes the trust value of each node to be rated as the input value of CNN. Finally, the CNN obtains evaluation points through self-study convergence and predicts the obtained trust level of vehicle nodes.

4. Test

4.1. Experiment Environment. The computer platforms used for this test platform are Intel I7-8750H, 2.20 GHz CPU, 16 GB memory, 1 TB external memory, Ubuntu 20.04 operating system, and MySQL 8.0.22 database system. The blockchain system adopts hyperledger fabric architecture, the operating system is Ubuntu 20.04, the memory size is 8 GB, the development language is Java, and the blockchain type is federation chain. The vehicle model of the Internet of Vehicles system is BYD Tang 2021 automatic flagship model, the intelligent vehicle network system in the vehicle is DiLink intelligent system, CPU is 8-core 2.0 GHz processor, memory is 3 GB, external memory is 32 GB, the power system is 2.0 T 141 kW L4, maximum power is 141 kW, and maximum torque is 320 N·m.

The parameter settings are as follows: The input matrix is 80 by n. The number of convolutional filters is 80 and the number of convolution filter windows is 4. We choose a sigmoid function as a convolution layer activation function and set up 3 sampling layers. Then, we used a gradient descent method for parameter optimization and repeated training for 100 iterations.

The experimental environment of this study is listed in Table 3.

4.2. Computation and Evaluation. Through the above four steps, we have implemented the main experimental steps of zero trust. We will take the above four steps, that is, client authentication pass rate, fingerprint authentication pass rate, success rate of API certification, and SPA packet transmission receiving rate, as the four factors for fuzzy algorithm sets. According to the main influencing factors of each step, the weight of each step is reviewed by experts. From Tables 4–7, we describe the data distribution tables of the four evaluation indicators at each grade after rating evaluation of 100 tests.

First, the single factor evaluation is carried out, and the evaluation r_i of each factor is obtained through statistical data. We establish the fuzzy relationship between each evaluation index U_i and trust level D_i. Based on the actual test results of the three vehicles, we have obtained three evaluation matrices and then combined them with the weight set to calculate the trust value in an evaluation cycle:

$$R_1 = \begin{pmatrix} 0.82 & 0.13 & 0.03 & 0.02 & 0.00 \\ 0.7 & 0.1 & 0.15 & 0.04 & 0.01 \\ 0.6 & 0.23 & 0.1 & 0.04 & 0.03 \\ 0.48 & 0.21 & 0.13 & 0.10 & 0.08 \end{pmatrix},$$
$$R_2 = \begin{pmatrix} 0.85 & 0.05 & 0.05 & 0.03 & 0.02 \\ 0.75 & 0.08 & 0.08 & 0.05 & 0.04 \\ 0.63 & 0.19 & 0.3 & 0.08 & 0.07 \\ 0.54 & 0.05 & 0.12 & 0.15 & 0.14 \end{pmatrix}, \quad (6)$$
$$R_3 = \begin{pmatrix} 0.82 & 0.11 & 0.05 & 0.02 & 0.00 \\ 0.73 & 0.12 & 0.08 & 0.05 & 0.02 \\ 0.58 & 0.15 & 0.12 & 0.09 & 0.06 \\ 0.52 & 0.17 & 0.08 & 0.13 & 0.1 \end{pmatrix}.$$

By multiplying the evaluation matrix R_i of the three cars with the weight vector $W = \{W_1(0.4), W_2(0.2), W_3(0.2), W_4(0.2)\}$, the fuzzy evaluation T_i can be obtained:

$$\text{Vehicle 1 } T_1 = (0.684 \quad 0.160 \quad 0.088 \quad 0.044 \quad 0.024), \quad (7)$$

$$\text{Vehicle 2 } T_2 = (0.724 \quad 0.084 \quad 0.066 \quad 0.068 \quad 0.058), \quad (8)$$

$$\text{Vehicle 3 } \mathrm{T}_3 = (0.694 \quad 0.132 \quad 0.076 \quad 0.062 \quad 0.036). \quad (9)$$

Table 3: System configuration parameters.

Configuration items	Computer system	Configuration items	Blockchain system
CPU	Intel i7-8750H, 2.20 GHz	Name	Hyperledger fabric
MEM	16.00 GB	Operating system	Ubuntu 20.04 TLS
SSD	1 TB	MEM	8 GB
Operating system and version number	Ubuntu 20.04	Development language	Java
Database system and version number	Mysql 8.0.22	Types of chain	League chain

Table 4: Grade distribution of client authentication pass rate.

Vehicle	Complete trust	General trust	Neutrality	General distrust	Complete distrust
Vehicle 1	82	13	3	2	0
Vehicle 2	85	5	5	3	2
Vehicle 3	82	11	5	2	0

Table 5: Grade distribution of fingerprint certification pass rate.

Vehicle	Complete trust	General trust	Neutrality	General distrust	Complete distrust
Vehicle 1	70	10	6	4	0
Vehicle 2	75	8	8	4	4
Vehicle 3	73	12	8	5	2

Table 6: Grade distribution of API certification pass rate.

Vehicle	Complete trust	General trust	Neutrality	General distrust	Complete distrust
Vehicle 1	60	23	10	4	3
Vehicle 2	63	19	3	8	7
Vehicle 3	58	15	12	9	6

Table 7: Grade distribution of SPA packet delivery acceptance rate evaluation.

Vehicle	Complete trust	General trust	Neutrality	General distrust	Complete distrust
Vehicle 1	48	21	13	10	8
Vehicle 2	54	5	12	15	14
Vehicle 3	52	17	8	13	10

Finally, we will calculate the comprehensive evaluation matrix X.

$$X_1 = T_1 * R_1 = 3.936 \quad X_2 = T_2 * R_2 = 3.848 \quad X_3 = T_3 * R_3 = 3.886. \tag{10}$$

According to the scores, we can conclude that the scores of the three cars are between 3.5 and 4. According to the permission set, the three cars will obtain the general trust level, that is, the vehicle and the user can access and edit all the data. Through many experiments, we get the scores of the three cars, as shown in Figure 9.

In the first few cycles, as the number of tests increases, the trust value of the three vehicles also increases continuously. In the process of multiple visits, according to the dynamic engine detection, there is no malicious operation. Therefore, the trust value of vehicle 1 rises steadily, and the rise slows down gradually after reaching a relatively high level of trust. In terms of vehicle 2, when the initial access conforms to the access the specification, the trust value rises slowly. But after 3 times detection, due to the malicious operation, in the subsequent trust value tests, vehicle 2 keeps falling and the access right it has gained also lowers. If subsequent vehicle 2 does not change

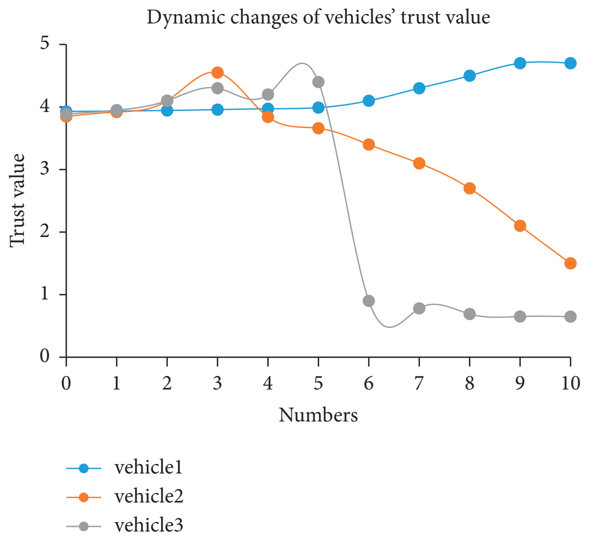


Figure 9: Dynamic changes of vehicles' trust value.

its behavior, it will eventually reduce to trust level 1 and the system will cease to serve it; when it comes to vehicle 3, to begin with, its trust value is rising and we can see at this time, there is no malicious operation in the process of access. However, after 5 times detection, it is identified as malicious nodes. Thanks to its great harmfulness, it is detected by the dynamic detection engine immediately and has directly reduced to the trust level 1and refused to continue to provide service for the vehicle. Thus, this model can accurately describe the behavior of vehicles and quickly identify malicious nodes.

In the test of dynamic evaluation engine, trust values of three vehicles will be updated once every cycle, and the size of trust values will affect the access rights of vehicles. Client authentication and device fingerprint authentication will determine whether the users can register normally and enter the Internet of Vehicles system. API gateway authentication and single-packet authentication access will identify whether malicious operations will occur during the user's access. For normal users, their trust value will finally reach a higher score range. At this moment, they can access and edit all data. For malicious nodes, the trust value will change rapidly. For example, the vehicles are suddenly controlled to attack the systems, resulting in the loss of system files or a system crash. In this case, the nodes will be forcibly disabled from access.

Finally, we use relevant data sets to conduct experiments on the convolutional neural network (CNN) model. The experiment consists of two levels: the first level is the input feature dimension analysis to analyze the accuracy of the classification of the trust level of vehicle nodes and the second level is to compare the accuracy of the algorithm under different noise ratios.

Let X_i be the internal state of a neuron node constituting the neural network, Y_i be the trust level of the input vehicle, and W be the weight of the connection from X_i to Y_i. If B_i is the external input signal (in some states, the neuron node Z_i can be controlled to keep it in a certain state), then its formal description is as follows:

The input Y_i of the trust node can be expressed as the measured value of the behavior attributes, and the influence of each behavior attribute of the users' trust value can be expressed by the weight W.

In order to deal with vehicle features of different dimensions, we define the maximum vehicle dimension as 80, and the dimensions less than the maximum length are filled with zero vectors. A stochastic gradient descent algorithm is used for model parameters. The outputs of each convolution layer and full connection layer in the model are connected to the ReLU activation function. The outputs of the last full connection layer are classified by SoftMax, and dropout is used to prevent overfitting. Comparative experiments are conducted on data sets in different fields to verify the accuracy of trust level assessment of the test model on data sets. Before and after adding the trust level tendency dimension, the accuracy of user trust level classification was analyzed. The experimental results are shown in Figure 10. It can be seen that when more feature attributes are used to participate in the convolution calculation, the classification accuracy of trust level will be significantly improved. However, with the increase of attributes, the classification accuracy does not necessarily increase together. The improper selection of feature attributes will reduce the accuracy of classification.

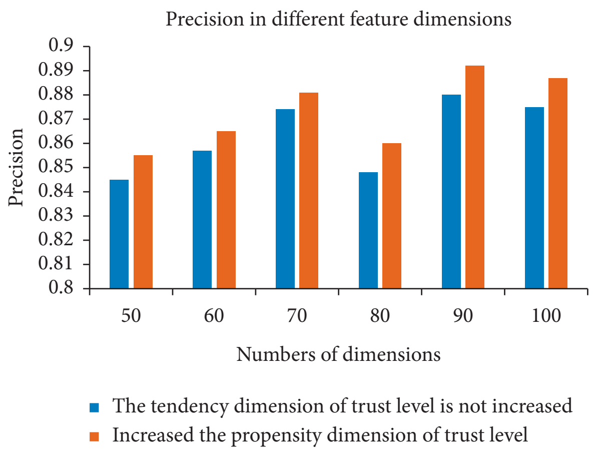


FIGURE 10: Precision in different feature dimensions.

Therefore, when we select the dimension of 80, the accuracy rate fluctuates to some extent. When the dimension of trust level tendency is added, the accuracy of the model calculation increases significantly compared with that without the dimension of trust level tendency, and the best effect of this experiment is achieved when the dimension is increased to 90. To solve the problem that the accuracy decreases when the dimension of feature attributes increases, we add noise value to trust classification to compare the performance of classification algorithms. As shown in Figure 11, the classified noise ratios of 20%, 30%, 40%, 50%, 60%, 70%, and 80% were respectively taken as four analysis environments. The results show that when the noise ratio is relatively small, the model has good accuracy, and when the noise ratio increases, the classification accuracy of the algorithm tends to decrease.

By using the CNN model proposed in this study, the predicted trust level by the CNN model is compared with the trust level obtained by zero-trust evaluation. We find that the result is basically consistent. We put the number of pass rates about client authentication, fingerprint identification authentication, API authentication, SPA packet acceptance, and other attributes of the vehicle into the CNN model. The corresponding trust level is obtained through self learning, which indicates that the model constructed in this study can extract key feature values and transform them into higher-level features through learning. The comparison results of trust level output are listed in Table 8.

Through the evaluation of the above CNN model, we find that the accuracy of CNN model is not only affected by the weights of different vehicle characteristics but also affected by the noise ratio. In order to improve the prediction accuracy of CNN model, the vehicle characteristic attributes should not be set too much. The horizontal dimension should be kept at 75, and the noise ratio should be set at 20%, so that the prediction accuracy can be higher.

5. Analysis

5.1. Security. After the CA certificate and identity fingerprint generated by PKI are authenticated, the data are stored in the blockchain through the consensus mechanism. The trust evaluation engine will evaluate the certificate level and

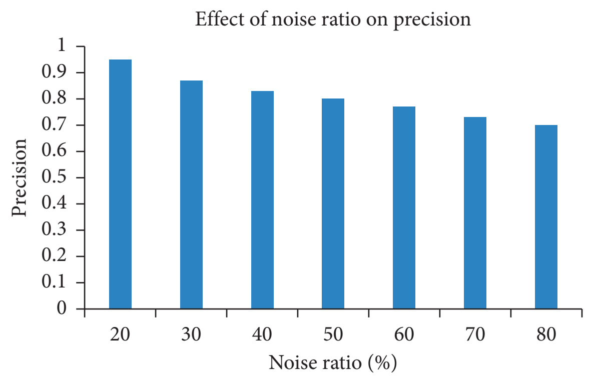


FIGURE 11: Effect of noise ratio on precision.

TABLE 8: The comparison results of trust level output.

Vehicle	Trust level output	Predicted output	Deviation rate (%)
Vehicle 1	3.936	3.91	0.66
Vehicle 2	3.848	3.81	0.99
Vehicle 3	3.886	3.85	0.93

historical behavior and convert them into relevant trust values. If the client wants to send data, it needs to be managed and verified through the API gateway. At this time, the client needs to be authorized by the controller to establish an MTLs connection and then requests the gateway to establish a connection. The gateway decrypts the data according to the key obtained from the control end. After the authentication succeeds, the client establishes a connection with the gateway to transmit data packets. The success rate of data transmission and the pass rate of certificate verification by the gateway and controller are converted into corresponding trust values for calculation. According to the size of the trust value, the trust level is determined and the corresponding service permissions are obtained. The access authority of the vehicle is modified by periodically updating the trust value to ensure the minimum access authority of the vehicle.

5.2. Confidentiality. In the Internet of Vehicles system, the session key is negotiated between the cloud server and the vehicle using the SM2 key negotiation algorithm to complete the secure transmission of sensitive data. In the Internet of Vehicles system, the ECS invokes the server cipher machine and uses SM4 and HMAC-SM3 algorithms to protect the confidentiality and integrity of user data and authentication data. At the same time, the vehicle calls the vehicle password module and uses SM4 and HMAC-SM3 algorithms to protect the confidentiality of key data such as identity authentication data, vehicle acquisition, and control data.

5.3. Attacks Defense

5.3.1. Switch Attack. In the process of evaluation of the Internet of Vehicles terminal nodes, if the attacker keeps performing well for a period to accumulate trust value, then it will reach a high trust level. Then, if it suddenly launches an attack, and then it will return to the state of good performance again. In this experiment, under the dual action of zero-trust historical trust record evaluation and incentive mechanism, the switch attack will be effectively suppressed. When the attacker carries out a switch attack, this malicious behavior will be recorded and uploaded to the blockchain network as a record of historical trust evaluation. Besides, users' trust will also decline rapidly. For those with malicious behaviors, if they want to increase their trust value again, then they will find that it grows at a very slow rate.

5.3.2. Novice Attack. Some attackers eliminate the historical interaction records badly and regain the trust of other Internet of Vehicles terminal nodes by registering new user identity information, thus launching attacks again in the network. Such attacks occur when malicious nodes can easily register as new vehicle terminal nodes. Aiming at the characteristics of novice attackers, we can use two-factor authentication, namely, fingerprint authentication for the vehicle and PKI authentication for the user to improve the difficulty of initial authentication. At the same time, the trust evaluation engine is used to render less trust value to the users who have passed the two-factor authentication. Only when they continue to perform trusted operations and other vehicle terminal nodes authenticate them with trust, the trust value can be gradually increased to become a trusted vehicle terminal node.

5.3.3. Replay Attack. Through the use of port knockout technology, the port for the communication of authentication information is closed, and the default discard firewall policy is dynamically reconfigured to allow access to services that would otherwise be blocked. Through this mechanism, remote users can be authenticated before granting access to services such as SSH daemon. On the SPA server system, fwknopd will sniff the replayed SPA packet and compare the SHA-256 summary of this packet with the SHA-256 summary of all previously seen and correctly decrypted SPA packets. If there is a match, fwknopd knows that a replay attack has been made. In this case, fwknopd generates a warning through syslog and does not grant access to the attacker to stop the replay attack.

5.3.4. Internal Attacks. In the Internet of Vehicles, there are also internal threats, which are more difficult to defend against than the external attacks mentioned above. The internal threats in car networking include three basic types of attacks such as networking system damage, zero-trust framework intellectual property theft, and electronic fraud. The focus and difficulty of the internal defense of the system lie in the internal staff, so as to ensure the efficient work and concerted efforts of the internal staff, which is what we need to do. On the one hand, we focus on systematic innovation; on the other hand, we strengthen internal cooperation and communication among employees. Not only there must be a positive working atmosphere but also the relevant management treaty. Insider threats can only be eradicated if everyone follows the rules.

5.4. Integrity. User's privacy information, device fingerprint, vehicle's trust value, punishment measures against malicious nodes, etc. will be uploaded to the blockchain in a specific data format. The participating organizations in the blockchain network can automatically update the distributed ledger through a unified consensus mechanism and algorithm, and there is no third-party centralized organization to participate. At the same time, the relevant principles of cryptography are used for data verification, and multiple private keys are used for access control. It ensures the security and nontamperability of users' data.

6. Conclusions

Aiming at the problems of easy disclosure of user data privacy in the Internet of Vehicles, we propose a data protection scheme for users in the Internet of Vehicles, which combines zero trust, blockchain, commercial cryptography, and other technologies in the Internet of Vehicles for the first time. It has achieved good results. The scheme relies on terminal environment analysis, trusted identity authentication, behavior model analysis, and other means to enhance the authentication ability and carries out minimum authorization and dynamic access control for the accessed vehicles, personnel, third-party enterprises, and institutions. It can effectively solve the security problems in the scenario of interconnection with people, vehicles, roads, and the cloud platform. It solves the risks encountered in data collection, data transmission, data use, data storage, data sharing, data destruction, and prevention of data leakage, data misuse, and abuse. The scheme protects the data security in the cloud platform of the Internet of Vehicles.

In view of the problems such as data loss and inadequate protection measures in data transmission of the Internet of Vehicles, we carry out PKI authentication and fingerprint identification as two-factor authentication for vehicles in the blockchain network. The SM series national secret algorithm is used for encryption and transmission, and the zero-trust evaluation engine and access control engine are used to realize identity authentication and authorization, improve the security and integrity of data transmission, and clarify the method of zero-trust system for vehicle networking data protection. Through the experiment of three vehicles, we tested the pass rate of identity authentication, the pass rate of fingerprint identification authentication, the success rate of API authentication, and the transmission and reception rate of SPA package and analyzed the change of trust value. The system can update the trust value of Internet of Vehicles devices in real time. By using the convolutional neural network to train the trust level of vehicle nodes, the set convolutional neural network can be trained to study the effectiveness of its application to the target, and the classification of user trust level in the convolutional neural network algorithm for specific applications can be determined quickly. Through the simulation of the influence of dimensions and weights of the four characteristic attributes of the vehicle on CNN identification performance, the specific dimension range of the characteristic attributes of vehicle nodes is determined.

This scheme has also some limitations. In the zero-trust framework, there are not enough vehicle nodes deployed to effectively simulate the huge Internet of Vehicles architecture in reality. Some representative nodes are selected in the scheme, which simplifies the workflow on the one hand and lacks sufficient data analysis on the other hand. In view of the limited number of selected nodes, we select some key nodes, increase the range of measurement data, and reduce the selection unit of experimental data.

Next, we will mainly conduct two aspects: one is to study the incentive measures of the trust evaluation model and the other is to conduct the system security and performance test targeted at the more complicated attack behaviors.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Hainan Provincial Natural Science Foundation of China (grant no. 621RC508), Henan Key Laboratory of Network Cryptography Technology (grant no. LNCT2021-A16), the Science Project of Hainan University (grant no. KYQD(ZR)-21075), National Natural Science Foundation of China (grant no. 62162020), the Hainan Provincial Natural Science Foundation of China (grant no. 620RC563), the Hainan Province Science and Technology Special Fund (grant no. ZDYF2021GXJS216), and the Science Project of Hainan University (grant no. KYQD(ZR)20021).

References

[1] B. H. Li, *Secure Communication for Internet of Vehicles Based on Blockchain*, Doctoral Dissertation, Chongqing University of Posts and Telecommunications, Chongqing, China, 2019.

[2] Y. H. Zhou, Z. Q. Lv, Y. G. Yang, and W. Shi, "Data deposit management system basedon blockchain technology," *Netinfo Security*, vol. 19, no. 8, pp. 8–14, 2019.

[3] J. W. Jiang, Z. Hong, and Z. J. Chen, "Application of SM4 encryption algorithm in Internet of vehicles," *Computer Networks*, vol. 46, no. 3, pp. 58–60, 2020.

[4] K. Feng, W. Li, and J. Gong, "Analysis on the application status of cryptographic algorithms in Internet of vehicles," *China Information Security*, vol. 2019, no. 9, pp. 97–99, 2019.

[5] X. K. Chen, *Zero-knowledge Identity Authentication Technology for Internet of Vehicles Based on Consortium Blockchain*, PhD Dissertation, Zhejiang University of Science and Technology, Hangzhou, China, 2020.

[6] L. J. Zhang, Y. F. Zou, W. Z. Wang, Z. Jin, Y. Su, and H. Chen, "Resource allocation and trust computing for blockchain-

enabled edge computing system," *Computers & Security*, vol. 105, Article ID 102249, 2021.

[7] S. A. Mostafa, A. Mustapha, S. S. Gunasekaran et al., "An agent architecture for autonomous UAV flight control in object classification and recognition missions," *Soft Computing*, vol. 150, pp. 1–14, 2021.

[8] M. Poongodi, M. Malviya, M. Hamdi, M. Mohammed, H. T. Rauf, and K. A. Al-Dhlan, "5G based Blockchain network for authentic and ethical keyword search engine," *IET Communications*, vol. 16, no. 5, pp. 442–448, 2021.

[9] S. Kumar, R. S. Raw, A. Bansal, M. A. Mohammed, P. Khuwuthyakorn, and O. Thinnukool, "3D location oriented routing in flying ad-hoc networks for information dissemination," *IEEE Access*, vol. 9, pp. 137083–137098, 2021.

[10] H. Xiong, C. J. Jin, M. Alazab et al., "On the design of blockchain-based ECDSA with fault-tolerant batch verication protocol for blockchain-enabled IoMT," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 5, pp. 1977–1986, 2022.

[11] T. Wang, Y. Quan, X. S. Shen, T. R. Gadekallu, W. Wang, and K. Dev, "A privacy-enhanced retrieval technology for the cloud-assisted Internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 7, pp. 4981–4989, 2022.

[12] S. A. Mostafa, M. S. Ahmad, A. Mustapha, and M. A. Mohammed, "Formulating layered adjustable autonomy for unmanned aerial vehicles," *International Journal of Intelligent Computing and Cybernetics*, vol. 10, no. 4, pp. 430–450, 2017.

[13] L. Zhang, Z. Zhang, W. Wang, Z. Jin, Y. Su, and H. Chen, "Research on a covert communication model realized by using smart contracts in blockchain environment," *IEEE Systems Journal*, vol. 99, pp. 1–12, 2021.

[14] W. Z. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, and C. Su, "Blockchain-based reliable and efficient certificateless signature for IIoT devices," *IEEE Transactions on Industrial Informatics*, vol. 11, pp. 1551–3203, 2021.

[15] Z. T. Lian, W. Z. Wang, and C. H. Su, "COFEL: Communication-Efficient and Optimized Federated Learning with Local Differential Privacy," in *Proceedings of the IEEE ICC*, Montreal, Canada, June 2021.

[16] L. J. Zhang, M. H. Peng, W. Z. Wang, Z. Jin, Y. Su, and H. Chen, "Secure and efficient data storage and sharing scheme for blockchain-based mobile-edge computing," *Transactions on Emerging Telecommunications Technologies*, vol. 5, 2021.

[17] J. C. Song, Z. Y. Han, W. Z. Wang, J. Chen, and Y. Liu, "A new secure arrangement for privacy-preserving data collection," *Computer Standards & Interfaces*, vol. 80, Article ID 103582, 2021.

[18] N. M. Balamurugan, S. Mohan, M. Adimoolam, A. John, T. R. Gadekallu, and W. Wang, "DOA Tracking for seamless connectivity in beamformed IoT-based drones," *Computer Standards & Interfaces*, vol. 79, Article ID 103564, 2021.

[19] B. B. Wang, *Research on Vehicular Ad-Hoc NetworksMessage Authentication Scheme Based on Lliptic Curve*, Doctoral Dissertation, Northwest Normal University, Lanzhou, China, 2020.

[20] Y. Xie, X. Li, S. S. Zhang, and L. B. Wu, "An improved provable secure certificateless aggregation signature scheme for vehicular ad hoc NETworks," *Journal of Electronics and Information Technology*, vol. 42, no. 5, pp. 1125–1131, 2020.

[21] X. W. Li, D. Q. Yang, X. Zeng, X. W. Zhu, B. H. Chen, and Y. Q. Zhang, "Cross—domain authentication and the key agreement protocol in VANETs," *Journal of Xidian University*, vol. 48, no. 1, pp. 141–148, 2021.

[22] Y. Xin, X. Feng, and T. T. Li, "Position related lightweight Sybil detection approach in VANET," *Journal on Communications*, vol. 38, no. 4, pp. 110–119, 2017.

[23] Y. L. Shi and L. M. Wang, "Spatio temporal analysis based resist conspiracy Sybil attack detection in VANETs," *China Information World*, vol. 41, no. 9, pp. 2148–2161, 2018.

[24] L. Si, *Research on Authentication Scheme for Vehicle-Mounted Ad Hoc Network under Cloud Service Environment*, Doctoral Dissertation, Tiangong University, Tianjin, China, 2019.

[25] X. Y. Zhang, *Research on New Vehicular Network SecurityArchitecture and Privacy Protection Authentication Method*, Doctoral Dissertation, Anhui University, Hefei, China, 2020.

[26] J. Y. Zhang, *Research on Security Authentication and Privacy protection Mechanism of Vehicular Cloud Computing*, PhD Dissertation, Beijing Jiaotong University, P. R. China, 2018.

[27] W. W. Chen, L. Cao, and C. H. Shao, "Blockchain based efficient anonymous authentication scheme for IOV," *Journal of Computer Applications*, vol. 40, no. 10, pp. 2992–2999, 2020.

[28] Z. Y. Ma, *Research and Implementation of Distributed Trustscheme for Vehicular Ad Hoc Network Based on Blockchain*, Doctoral Dissertation, Nanjing University of Posts and Telecommunications, Nanjing, 2020.

[29] B. Xu, *Vehicle Attribute Recognition Based on Convolutional Neural Network*, Doctoral Dissertation, Beijing Institute of Technology, P. R. China, 2015.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 8398591, 11 pages
https://doi.org/10.1155/2022/8398591

Research Article

FGL_Droid: An Efficient Android Malware Detection Method Based on Hybrid Analysis

Weiping Wang,[1] **Congmin Ren**,[1] **Hong Song**,[1] **Shigeng Zhang**,[1,2] **and Pengfei Liu**[1]

[1]*School of Computer Science and Engineering, Central South University, Changsha, Hunan, China*
[2]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*

Correspondence should be addressed to Hong Song; songhong@csu.edu.cn

Received 22 December 2021; Revised 3 March 2022; Accepted 14 March 2022; Published 28 April 2022

Academic Editor: Weizhi Meng

With the popularity of Android intelligent terminals, malicious applications targeting Android platform are growing rapidly. Therefore, efficient and accurate detection of Android malicious software becomes particularly important. Dynamic API call sequences are widely used in Android malware detection because they can reflect the behaviours of applications accurately. However, the raw dynamic API call sequences are very usually too long to be directly used, and most existing works just use a truncated segment of the sequence or statistical features of the sequence to perform malware detection, which loses the execution order information of applications and consequently results in high false alarm rate. In this work, we propose a method that transforms the dynamic API call sequence into a function call graph, which retains most of the application execution order information with significantly reduced sequence size. To compensate for the missed behaviour information during the transformation, the advanced features of permission requests extracted from the application are utilized. We then propose FGL_Droid, which fusions the transformed function call graph feature and the extracted permission request feature to perform accurate malware detection. Experiments on benchmark dataset show that FGL_Droid achieves a high detection accuracy of 0.975 and a high F-score of 0.978, which are better than the existing methods.

1. Introduction

To prevent the threats caused by Android malware including financial loss to users, information leakage, and system damage, an efficient and accurate detection scheme is urgently needed. In recent years, Android operating system has been growing at an alarming rate. As of 2020, approximately 3 billion Android-based devices were shipped, accounting for 80% of all mobile operating systems [1]. Therefore, Android devices have become a popular target for malware developers. In 2020, Kaspersky mobile products and technologies detected 5,683,694 malicious installation packages, increasing 2.1 million from last year, 156,710 new mobile banking Trojans, and 20,708 new mobile ransomware Trojans [2]. In order to protect the property and information security of Android device users, it is urgent to provide an efficient and accurate malware detection method.

A large number of companies and researchers have conducted research on malware detection. The mainstream detection scheme mainly includes two categories: static analysis [3–6] and dynamic analysis [7–13]. Static analysis does not need to execute the application. It extracts features from the APK file through reverse engineering, such as Permissions [3], API calls [14], bytecodes, and other static features [4], and then performs malicious detection based on these features. However, with the advancement of technology, many malwares begin to use code obfuscation or dynamic loading techniques to hide static features, which leads static analysis schemes to be completely ineffective. On the contrary, dynamic analysis needs to actually run the program and collect information about its behaviour at running time [15]. By analysing the application's behaviour, dynamic analysis can achieve higher accuracy and robustness than static analysis. In this paper, we focus on dynamic analysis.

Dynamic API call sequence can describe the behaviour information of the application, which contains all the operations during application execution (e.g., accessing network, sending SMS, etc.). It is an important data for application behaviour analysis and has been applied by many researchers in Android malicious judgment. However, in order to trigger malicious behaviour completely, a huge number of detective operations are required, which will produce distinctly long API call sequences, reaching millions of levels. It is a great challenge to process the huge amount of dynamic API call sequence. According to our statistics on dynamic API call sequences, the average length of collected dynamic API sequences is 1.698 million, and the number of different API types is 25834. There are two problems in using dynamic API call sequence as the feature for malicious determination: (1) The large amount of data makes it difficult to find the behaviour information of the application. (2) There are many kinds of features, which makes the model easy to overfit.

Several related works have used dynamic API call sequence to detect Android malware; the main challenge is how to process dynamic API call sequence to reduce the amount of data size. The current solutions are broadly categorized into two types: using statistical features of dynamic API call sequence and truncating a fixed-length subsequence of dynamic API call sequence. The methods of using statistical features of dynamic API call sequence have an advantage that the computational overheads are relatively low, but these methods cannot save the order information of the APIs. The methods of truncating a fixed-length subsequence of dynamic API call sequence can save the order information; however, these methods will lose most of the sequence information, while the computational overheads are extremely high.

In this paper, we propose a scheme to transform the API call sequence into a function call graph. It can convert a million levels of dynamic API call sequence into a directed and edge-weighted graph structure with only a few nodes, which can both preserve information about the order of the API calls and greatly reduce the scale of data. We first replace each API with its corresponding function class to get the function call sequence and then convert the function call sequence to function call graph. Coarse-grained substitution loses part of semantic information. Therefore, we use advance feature Permission to make up for this part of semantic information. We use GCN (graph convolutional network) model to extract application's behaviour features and concatenate these features with permission feature to construct the final feature vector. Then we send the concatenated feature into LR (logistic regression) model to obtain classification result.

Our method can effectively solve the problem of excessive data volume of dynamic behaviour information. At the same time, the detection accuracy is better compared to the existing methods. The main contributions of this paper include the three following points:

(a) We developed a dynamic behaviour capture tool that integrates APE [16], which can capture API calls during application running

(b) We propose a method to convert a dynamic API call sequence into a function call graph, which can save the behaviour information of the application while reducing the amount of data

(c) We design a fusion model FGL_Droid, which can achieve higher detection accuracy

The rest of the paper is organized as fallows: Section 2 discusses related work for Android malware detection using dynamic detection scheme. Section 3 introduces the overall structure of our framework. The method of processing dynamic API call sequence and the fusion model is explained in Section 4. Section 5 shows the experimental results to illustrate the performance of our framework. Section 6 concludes our research.

2. Related Work

Faced with such a huge amount of dynamic API call sequence, the existing research solutions can be classified into the two following types: using statistical features of dynamic API call sequence and truncating a fixed-length subsequence of a dynamic API call sequence:

(a) Using statistical features of dynamic API call sequence: use statistical algorithms to select the APIs that have a greater impact on malware detection from the dynamic API call sequence, such as the information gain algorithm [17] and the TF-IDF (term frequency-inverse document frequency) algorithm [18], and then use the selected API as feature for malware detection

(b) Truncating a fixed-length subsequence of dynamic API call sequence: intercept the first part of the entire dynamic API call sequence, and then use deep learning algorithms for malware detection [19]

2.1. Malware Detection Based on Statistical Features of API Call Sequence. Uppal. et al. [20] extracted 3-gram vectors from dynamic API call sequence and then used the odds ratio to select the most important vectors. Finally, they applied SVM model to complete malicious judgment. Mohammed K. Alzaylaee et al. [17] used the tool DynaLog to extract 178 behaviour features of applications execution process and then used the information gain algorithm to select the most important 120 from them. In order to increase the accuracy of the model, the authors extracted permission requested by the application and formed a 420-dimensional feature vector. Finally, the authors used DNN (deep neural network) to determine maliciousness. Fang et al. [21] constructed a TFIT (trace frequency information table) to save the number of API calls. After that, they calculated the weight of each API call according to TFIT and selected the most important API calls. Finally, they fed the selected API calls into XGBoost. Yong Qiao et al. [8] used a frequent itemset mining algorithm to find frequent API calls from the API call sequence and then used frequent item sets as features for malware clustering. Singh et al. [22] utilized Cuckoo Sandbox to capture the application's dynamic API call sequence and then checked whether a certain API appeared as feature. Kim J. et al. [23] counted the frequency of each API in dynamic API call sequence as feature. Afterward, the maliciousness of applications was judged by comparing the difference in API call counts.

Merely using statistical features will lose the order information of API call sequence. However, the order of API calls is very important; different orders can indicate different behaviours. For example, *getContentResolver().query()* $\longrightarrow$ *socket.getOutputStrean().write()* can complete the behaviour of stealing user information, but, if conversely, there is no such behaviour. Such statistical methods damage the accuracy of malicious detection. At the same time, due to the excessive number of APIs, there are too many feature dimensions, which can easily cause overfitting problem and reduce the generalization ability of malicious detection.

2.2. Malware Detection Based on a Fixed-Length Subsequence of Dynamic API Call Sequence.

Xi Xiao et al. [9] extracted the system call sequence during application execution and then applied the LSTM model to extract the information in the sequence. In order to adapt the length of the sequence to the LSTM (Long Short-Term Memory) model, the authors truncateed the sequence, limiting the length of the sequence to 500. Wenqi Xie et al. [10] proposed an algorithm to segment the system call sequence. They first cut the system call sequence into fixed-length subsequences and then labeled each subsequence. Finally, the LSTM model is used to determine the maliciousness of each subsequence. Zhaoqi Zhang et al. [11] proposed a method to map an API and its parameters to a fixed-length vector. The LSTM model is used to determine its maliciousness based on the vector. Kolosnjaji et al. [12] proposed a scheme to integrate CNN and LSTM models. The method first uses CNN (convolutional neural network) to learn a set of features and then uses LSTM to determine the maliciousness. Pascanu et al. [13] proposed a phased detection model, including feature extraction stage and malicious detection stage. In the feature extraction phase, they used the RNN (recurrent neural network) to predict the next API call based on the previous sequence of API calls. In the classification stage, RNN is frozen, and all API outputs are converted into feature vectors by maximum pooling for classification.

The LSTM model can learn the sequential relationship of API calls. However, the best effect length of the LSTM model is 200, so, in order to make API sequences suitable for the LSTM model, many researchers intercept a part of API call sequences. The intercepted API call sequence cannot fully represent the behaviour of the application at runtime, and the malicious behaviour may be cut off, which greatly reduces the detection accuracy.

Therefore, we propose a method to convert API call sequence into a function call graph, which not only retains most of the application execution order information but also significantly reduces sequence size. We will introduce our schema in the next part.

Table 1 compares the existing state-of-the-art schemas with the proposed work. The schemas which use statistical features of API call sequence will lose the order information of API call sequence. The schemas which truncate a fixed-length subsequence of dynamic API call sequence will cause huge computational overheads. Therefore, we propose a method to convert API call sequence into a function call graph, which not only preserves the complete sequence but also preserves the order information of the sequence. We will introduce our schema in the next part.

3. System Framework

The overview of the FGL_Droid system is shown in Figure 1, which is divided into four parts: AndroidGuard [24], DyAPICapture, Graph Construct, and FGL_Droid model.

3.1. Static Feature Extraction.

In the static feature extraction stage, we use AndroGuard [24] to extract the static features of each application and select the permission request list as our static features. The reasons mainly include the two following aspects:

(a) The distribution of Permissions in normal applications and malicious applications is significantly different, which can reflect the malicious behaviour of applications to a certain extent.

(b) For benchmark dataset, we collect a total of 135 kinds of Permissions. Using Permissions as static features will not suffer from high-level dimensions.

3.2. Dynamic Feature Extraction.

We have designed a dynamic behaviour capture system DyAPICapture to capture the API calls during the running of the application. As shown in Figure 2, DyAPICapture mainly includes two modules: dynamic test module and API capture module. In the dynamic test module, we adapt the model-based stateful dynamic triggering scheme APE [16], which can achieve higher function coverage with less detection actions. In the API capture module, we use the functions provided by JDWP (Java Debug Wire Protocol) [25] to obtain the API calls during the execution of applications. At the same time, in order to achieve higher functional coverage, the entire test process will last for 5 minutes, and the captured dynamic API sequence length has reached millions level, with an average length of 1.698 million. Moreover, some malicious applications can detect the operating environment; if they are found to be running in a virtual machine, they will hide their malicious behaviour and avoid detection. Therefore, our dynamic capture system can be deployed on real devices.

3.3. Function Call Graph Construction.

The collected API call sequence is too long. We propose algorithm 1, which converts API call sequence into a directed and edge-weighted function call graph. Function call graph contains

Table 1: Comparison of the proposed work with state-of-the-art android malware detection schemas.

Schema	Features	Classification algorithm	Computational overheads	Capacity of saving information
[1]		SVM		
[2]	Statistical features of API call sequence	DNN	Medium	Low
[3]		XGBoost		
[7]		LSTM		
[10]	A fixed-length subsequence of API call sequence	CNN + LSTM	High	Medium
[11]		RNN		
Ours	**Function call graph + permission**	**GCN + LR**	**Low**	**High**

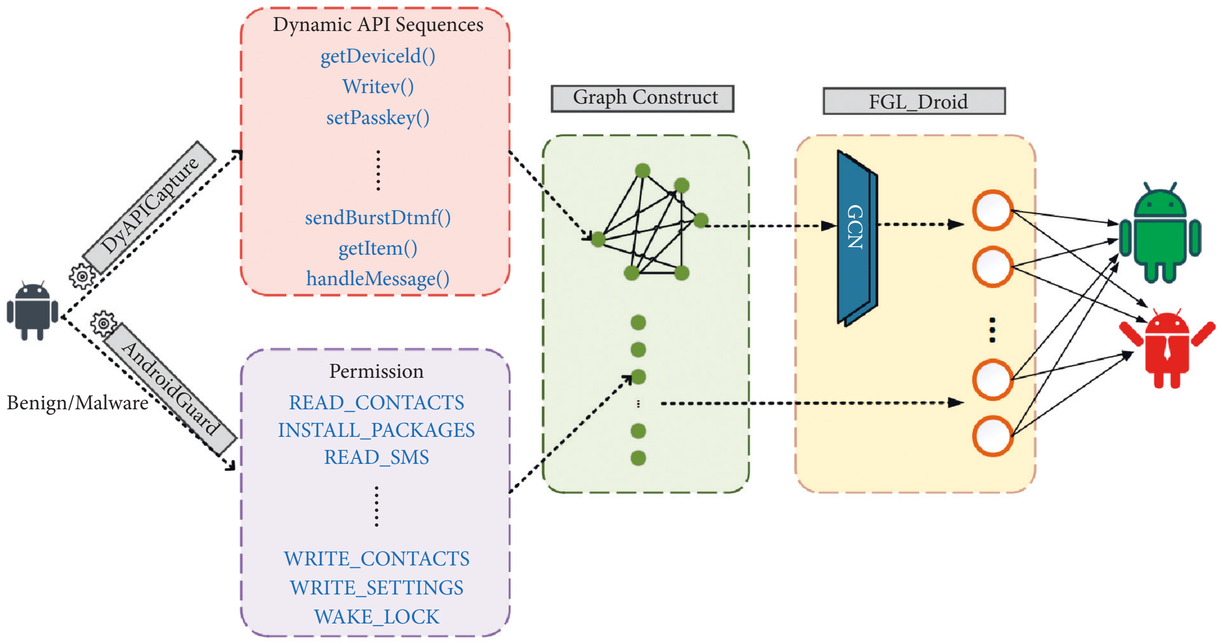


Figure 1: Framework of FGL_Droid. It includes four parts: DyAPICapture: extract the dynamic API call sequence; AndroidGuard: extract the Permission; Graph Construct: function call graph construction; and FGL_Droid: detection model.

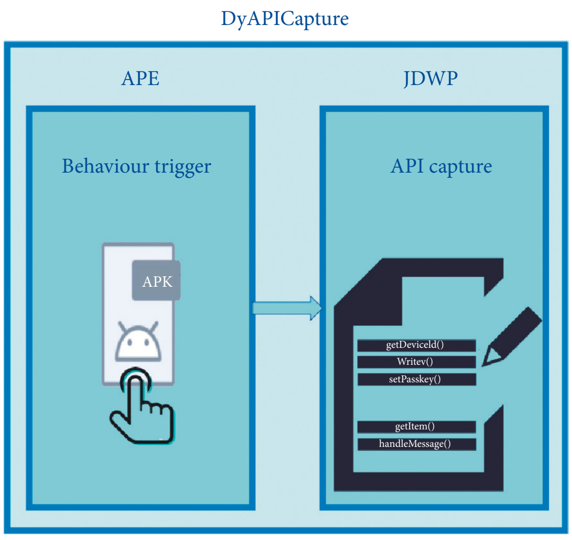


Figure 2: DyAPICapture consisting of two parts: behaviour trigger and API capturer.

26 vertices, each representing a function class in SUSI [26]. Each directed edge in function call graph represents a call relationship, and the corresponding weight means the number of call relationship. Therefore, the function call graph can keep order information of the API call sequence.

3.4. Model Training. We design a fusion model FGL_Droid, which can effectively mine applications behaviour pattern from the function call graph and merge it with the Permission feature. Once the function call graph is constructed, we train our FGL_Droid model on this graph for malware detection. The details of the entire model are explained in the Methodology chapter.

4. Methodology

4.1. Construction of the Function Graph. We propose a method to transform the sequence of API calls into a function call graph, which can save the order information of API calls while reducing data size of dynamic API call sequence. The method applies Algorithm 1 to construct a graph $G(V, E, W)$ by representing a unique function class at vertex V and a call relationship by a weighted edge E. As shown in Figure 3, Algorithm 1 mainly contains four steps: API substitution, deredundancy, graph construct, and normalization.

(1) API substitution: the algorithm first transfers the original dynamic API call sequence into function class sequence. Each API is replaced with the corresponding function class in SUSI [21]. As shown in Figure 3, the API call sequence [*'getDeviceId()', 'getPhoneNumber()', 'getCellLocation()', 'getSerialNumber()', 'query()', 'sendMessage()', 'sendSms()', . . ., 'getRawContactId()', 'getContactUri()', 'getAllContractName()', 'getViewAt()', 'getUserPassword()', 'getDriverName'*] will be converted into function call sequence [*'UNIQUE_IDENTIFIER', 'UNIQUE_IDENTIFIER', 'LOCATION_INFORMATION', 'NETWORK_INFORMATION', 'FILE_INFORMATION', 'ACCOUNT_INFORMATION', 'e-mail', 'SMS_MMS', . . ., 'CONTACT_INFORMATION', 'CONTACT_INFORMATION', 'CONTACT_INFORMATION', 'BROWSER_INFORMATION', 'SYSTEM_SETTINGS', 'NFC', 'NFC'*].

(2) Deredundancy: in order to reduce the length of function call sequence, we only retain one of the same function classes that appears continuously. The function classes call sequence will become [*'UNIQUE_IDENTIFIER', 'LOCATION_INFORMATION', 'NETWORK_INFORM-ATION', 'FILE_INFORMATION', 'ACCOUNT_INFORMATION', 'e-mail', 'SMS_MMS', . . ., 'CONTACT_INFORMATION', 'BROWSER_INFORMATION', 'SYSTEM_SETTINGS', 'NFC'*].

(3) Graph construction: Algorithm 1 converts the execution sequence of function classes to function call graph. For the two consecutive function classes [*'UNIQUE_IDENTIFIER', 'LOCATION_INFORMATION'*] in the sequence, we create a directed edge pointing from *UNIQUE_IDEN-TIFIER* to *LOCATION_INFORMATION.* If this edge does not exist in the graph, add this edge to the function call graph; if this edge exists in the graph, increase the weight of the edge by 1. We get the function call graph as shown in Figure 2 through this step.

(4) Edge normalization: in order to reduce the impact caused by different orders of magnitude of each edge, the min_max *normalization method* was adopted to map all the weights into a range between (0, 1). As shown in Equation 1, $W_{(i,j)}$ represents normalized weight of the edge, $w_{(i,j)}$ represents the number of calls, MAX_{weight} represents the maximum weight of the edge in the figure, and $\text{MIN}_{\text{weight}}$ represents the minimum weight of the edge in the figure. Finally, we obtain the directed weighted function call graph F_GRAPH (V, E, W) originating from API sequence. Each node of F_GRAPH represents a function class. Each directed edge in F_GRAPH represents a call relationship, and the corresponding weight means the frequency of call relationship. Therefore, the function call graph can keep sequence information of the API call sequence.

$$W_{(i,j)} = \frac{w_{(i,j)} - \text{MIN}_{\text{weight}}}{\text{MAX}_{\text{weight}} - \text{MIN}_{\text{weight}}}. \tag{1}$$

4.2. Fusion Model of GCN and LR. Our model consists of three steps: extracting dynamic behaviour feature, merging dynamic behaviour feature and Permission, and malicious judgment.

4.2.1. Extracting Dynamic Behaviour Feature. For function call graph F_GRAPH (V, E, W) obtained by Algorithm 1, we utilize graph neural network to extract application's behaviour feature from function call graph. For each vertex in the graph, the graph neural network can fuse the information in adjacent nodes and edges and keep the call information between nodes. The information fusion mode is

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}H^{l}W^{l}\right), \tag{2}$$

where A represents the sum of the adjacency matrix of the graph and the identity matrix, so that the information of its own nodes can be kept in the process of information fusion. D is the degree matrix of A, H is the characteristic of each layer, and σ is the activation function.

Through our observation, a malicious behaviour can be accomplished only by calling 2-3 function classes. For example, information leakage malware usually only needs to

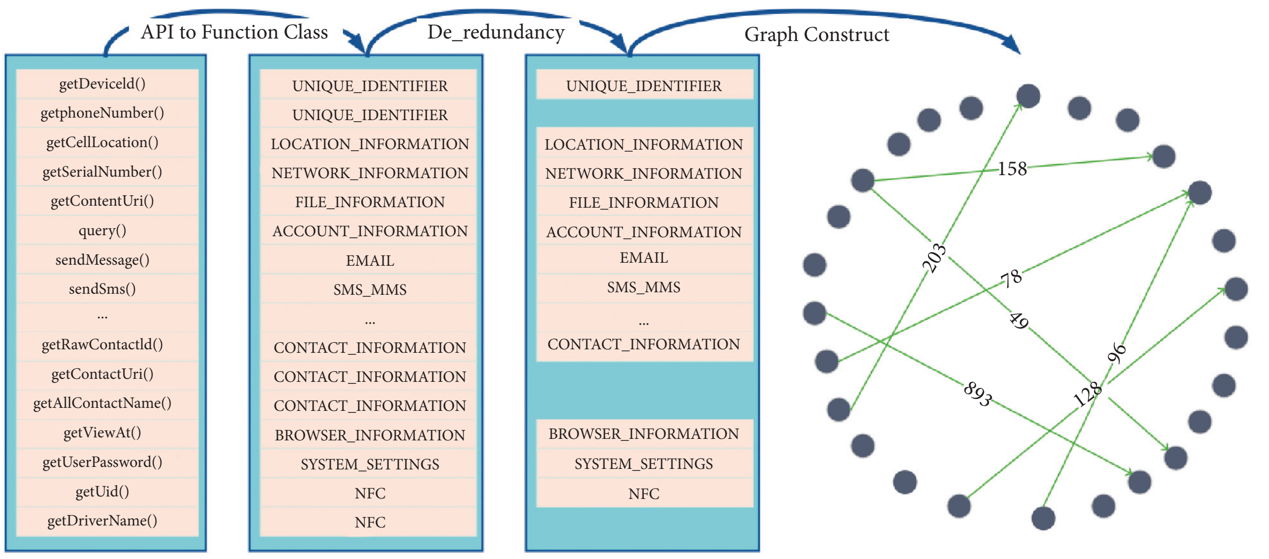


FIGURE 3: The process of converting dynamic API call sequence into function call graph. The figure only shows the first three steps.

```
(1)  Input: Dynamic API Call Sequence(S) and SUSI's API Category($D$)
(2)  Output: Function Call Graph F_Graph $(V, E, W)$
(3)  Initially: let $V = (F1, F2, F3, \ldots, F26)$ be all the function category in $D$ and $E \leftarrow \varnothing$
(i)  //Step 1: replace the API with corresponding function class
(4)  for API $\in S$ do
(5)          Change API to its class in $D$
(6)          let $F(f1, f2, f3, \ldots f_n)$ be the Function call Sequence
(7)  end for
(ii) //Step 2: delete adjacent duplicate function classes
(8)  while $i < \text{length}(F)$ do
(9)          if $F[i] = F[i-1]$ then
(10)                 delete $F[i]$
(11)     end if
(12) end while
(iii) //Step 3: convert the execution sequence of function classes to function call graph
(13) while $i < \text{length}(F)$ do
(14)         if $(F[i], F[i+1]) \notin E$ then
(15)                 Add an edge $(F[i], F[i+1])$ to $E$
(16)             $W(F[i], F[i+1]) = 1$
(17)     else
(18)             $W(F[i], F[i+1]) += 1$
(19)     end if
(20) end while
(iv) //Step 4: normalization
(21) maxWeight $\leftarrow$ max$(W)$, minWeight $\leftarrow$ min $(W)$
(22) for $w \in W$:
(23)         $w \leftarrow (w - \text{minWeight})/(\text{maxWeight} - \text{minWeight})$
(24) end for
(25) Return F_Graph $(V, E, W)$
```

ALGORITHM 1: Graph construction algorithm. Our algorithm includes four steps, which are (1) replace the API with corresponding function class, (2) delete adjacent duplicate function classes, (3) convert the execution sequence of function classes to function call graph, and (4) use max-min algorithm method for normalization.

obtain account information (*ACCOUNT_INFORMATION*) and then send it to the server through network (*NETWORK*); malicious download malware first needs to obtain the memory status of the phone (*UNIQUE_IDENTIFIER*) and then attach the information to the designated server (*NETWORK*) and finally download the malicious application to the designated memory through the network and decompress and install it (*FILE*). Therefore, in our GCN model, we use a double-layer GCN network. Each node merges the information of the nodes that are within two hops from the node, which can completely cover all the malicious behaviours.

4.2.2. Merging Dynamic Behaviour Feature. In order to reduce the amount of data and avoid the overfitting phenomenon caused by excessive feature dimension, we replaced API with the corresponding SUSI's function class to reduce the data dimension. Such coarse-grained method will inevitably lose some detailed information, resulting in a decrease in detection accuracy. We find Permission can be used to make up for the lost detail information. For example, sendTextMessage() and notifySendFailed() are assigned to the same functional class, losing the differences in their behaviours. However, sendTextMessage() requires Permission "SEND_SMS," but notifySendFailed() does not, so we can make up for lost details by introducing Permission. Due to the fact that the Permission information is an advanced feature, there is no need to apply complex deep neural network on this feature. In order to efficiently utilize both function call graph and Permission, a fusion model of GCN and LR is proposed, as shown in Figure 4. For complex graph data, we use double-layer GCN to extract the application behaviour information from the graph. Then apply Average_pooling and Max_pooling operations on each node and concatenate the two results together as the final graph representation Vector_Graph with length of 1150. We combine Vector_Graph with Permission features as Vector_mix.

4.2.3. Malicious Judgment. After getting the Vector_mix, we use a dense layer with 128 neural units to reduce the dimension of the intermediate vector to 128. A ReLU activation is adopted in this dense layer. Then we use a dropout layer with a rate of 0.6 to reduce overfitting. Finally, we use a full connection layer with an output dimension of 1 to obtain the malware probabilities. The binary cross-entropy function is adopted as our loss function, which can characterize the difference between the true sample label and the predicted probability. The loss function is

$$\text{Loss} = y \log\hat{y} + (1 - y)\log(1 - \hat{y}), \tag{3}$$

where y represents the true label and $\hat{y}$ represents the predicted label.

In addition, the optimization method we take is Adam, and the learning rate is 0.0005.

5. Results and Discussion

5.1. Datasets and Experimental Environment. The benchmark dataset used in this paper is shown in Table 2, including 4217 normal applications downloaded from the Google Play Store and 3950 malicious applications collected from Andro_dumpsys Project, which includes 13 malware families.

The configuration of the experimental equipment is shown in Table 3.

5.2. Effectiveness of Translating Dynamic APIs into Graph. To evaluate the effectiveness of translating dynamic APIs into function call graph, we measured the accuracies of the other two mainstream process methods: using the statistical features of dynamic API call sequences and intercepting a fix_length subsequence of the dynamic API call sequence:

(a) SF: using the statistical features of dynamic API call sequence. We count the number of occurrences of an API in dynamic API call sequence and then use DNN (deep neural network) to detect the maliciousness of the application. What we use is a three-layer DNN with 4500, 500, and 1 neural unit, respectively, and the activation function is ReLU. During the training process, we choose the learning rate of 0.003, the optimizer is the Adam optimizer, and the loss function is the cross-entropy loss function.

(b) FLS: intercepting a fix_length subsequence of the dynamic API call sequence. We use the first 1000 APIs of dynamic API call sequences and then use LSTM (Long Short-Term Memory) network to detect malware. What we use is a single-layer LSTM network, and the training process is consistent with the above parameters.

The results are shown in Table 4. Significantly, our process method of translating dynamic APIs into function call graph is superior to the other two processing methods in four evaluation indicators. As for the metric MTTD (mean time to detect), since we transformed the original sequence into a graph with only 26 vectors, our schema also outperforms the other two.

In the process of the experiment, we find that there is obvious overfitting phenomenon in schemes which use statistical features of dynamic API call sequences. On the contrary, our method does not have this problem. We record the change of loss value in the training process of using statistical features and Ours. As shown in Figure 5, we can clearly see that loss has been decreasing on the training set, but, on the test set, there has been a large increase after a small decrease. Obviously, there is an overfitting phenomenon when we use the statistical features of dynamic API call sequences to judge malicious.

Using the frequency of each API that appears in dynamic API call sequence as features will cause the data dimension to be too high. Google officially provides 25834 APIs, so the dimension of the feature is 25834. In some papers, they not only use API but also include Permissions and opcodes. The final feature dimension can reach tens of thousands of dimensions. The feature dimension is too high, so it is easy to suffer from overfitting. Our processing method converts the API call sequence into a graph with only 26 nodes, thus reducing the data dimension and avoiding the occurrence of overfitting.

5.3. Effect of the Layer of GCN. In this part, we verify the influence of different layers of GCN on the detection result. We conduct experiment under different layers of GCN from 1 to 5. The results are shown in Table 5. The best detection

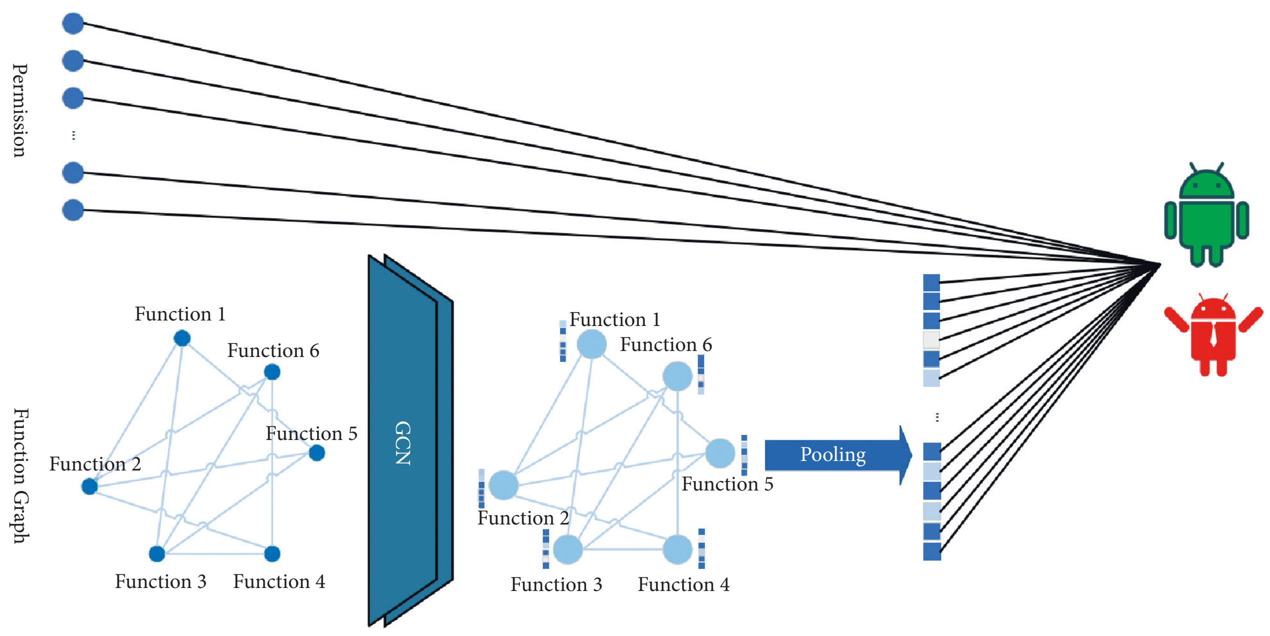


Figure 4: Fusion model of GCN and LR. Use GCN to extract the application behaviour pattern from the function call graph and then splice it with the Permission feature, and then use the LR model for malware detection.

Table 2: Summary of the dataset.

Dataset	Number	Source
Malware	3950	Andro_dumpsys
Benign	4219	Google Play Store

Table 3: Experiment environment.

Environment	Configuration
Operating system	Android 6.0.1
Phone model	LG Nexus 5

performance is achieved when there are 2 layers. For two-layer GCN, each node merges the information of the nodes that are within two hops from the node, which can completely cover all the malicious behaviours. For example, information leakage malware usually only needs to obtain account information (ACCOUNT_INFORMATION) and then send it to the server through network (NETWOR). When the layer of GCN is greater than 2, the representation of each node will tend to homogenize, which decreases the accuracy of detection.

5.4. Effectiveness of Normalizing the Edges of Function Call Graph. Android malware usually hides malicious code in normal code, and most of the dynamic API call sequences we extracted are normal call relationships. Therefore, the weight of the normal call relationship in the function call graph obtained from the dynamic API call sequence transformation is much larger than the malicious call relationship, which has a greater impact on the final judgment result. In order to reduce misjudgment caused by the large difference of edge weights, we normalized the edges.

To show the effect of normalizing the edges of the function call graph, we added experiments to verify the impact of edge normalization on the proposed scheme. The WW is a weighted graph without normalization, and WNW is a weighted graph with normalization. As shown in Table 6, the detection accuracy of WNW is significantly superior than the other method, reaching 0.975.

5.5. Effectiveness of the Integration Model. Due to the fact that the method of transforming dynamic API call sequences into function call graph loses some details, we propose a way to make up for the loss of details by merging the function call graph and Permission. As mentioned in the previous section, both *sendTextMessage()* and *notifySendFailed()* are assigned to the same functional class "*SMS_MMS*," losing the differences in their behaviours. However, *sendTextMessage()* requires Permission "*SEND_SMS*," but *notifySendFailed()* does not, so we can make up for lost details by introducing Permission. In order to evaluate the effectiveness of fusion features, we conduct the three following groups of experiments:

(a) Permission: use Permission only for maliciousness determination
(b) Function call graph: use only the function call graph for maliciousness determination
(c) Fusion feature: use fusion information to determine maliciousness

Table 7 shows the experimental results. It can be seen from the experimental results that our scheme is significantly better than only using Permission or function call graph, which indicates that our fusion scheme is indeed effective in the determination of maliciousness.

Table 4: Effectiveness of translating dynamic APIs into graph.

Process method	Approach	ACC	Precision	Recall	F1	MTTD (ms)
SF	DNN	0.925	0.933	0.933	0.933	0.208
FLS	LSTM	0.702	0.952	0.662	0.781	1.406
FCG + Permission	**GCN + LR**	**0.975**	**0.979**	**0.976**	**0.978**	**0.141**

SF: statistical features of dynamic API call sequence; FLS: fix_length subsequence of the dynamic API call sequence; FCG: function call graph.

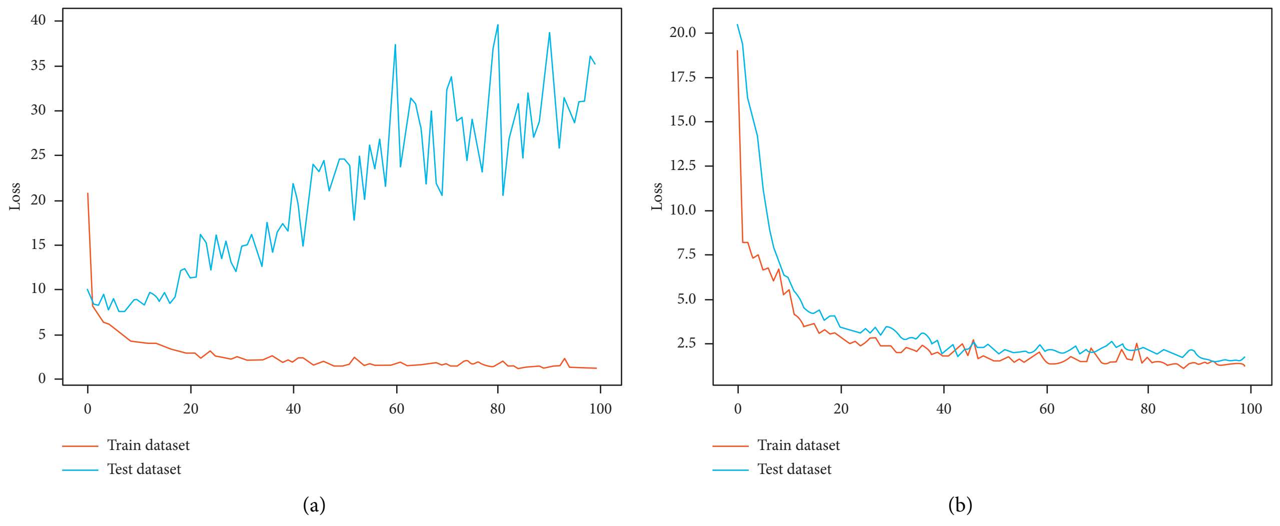


Figure 5: The change of loss value during model training and testing using different API call sequence processing methods. (a) Using statistical features of API call sequence. (b) Ours: using function call graph.

Table 5: Results under different numbers of GCN layers.

Layer	ACC	Precision	Recall	F1
1	0.957	0.963	0.940	0.951
2	**0.975**	**0.979**	**0.976**	**0.978**
3	0.968	0.977	0.966	0.972
4	0.954	0.969	0.954	0.961
5	0.953	0.962	0.934	0.948

Table 6: The performance comparison with other proposed systems.

Method	ACC	Precision	Recall	F1
FCG_WW	0.713	0.653	0.799	0.719
FCG_WNW	**0.975**	**0.979**	**0.976**	**0.978**

WW: function call graph with weights; WNW: function call graph with normalization weights.

Table 7: Evaluating the effectiveness of the integration model.

Feature	ACC	Precision	Recall	F1
Permission	0.748	0.665	0.999	0.799
Function call graph	0.920	0.904	0.906	0.930
Fusion feature	**0.975**	**0.979**	**0.976**	**0.978**

5.6. Model Evaluation. In order to study the performance improvement of FGL_Droid model, we compare the state-of-the-art scheme of malicious detection using dynamic API call sequences, which mainly includes two ways: using statistical features of API call sequences or intercepting the API call sequence to a certain length.

Table 8 shows the results of the comparative experiment. Our scheme is significantly better than other methods. The precision of FGL_Droid in the test set is 0.975, and the F-Measure is 0.978.

For the method of using the statistical features of the dynamic API call sequence [5, 17, 22], they lose the sequence

Table 8: The performance comparison with other proposed systems.

Approach	ACC	Precision	Recall	F1
Mohammed K. Alzaylaee [17]	0.950	0.941	0.978	0.959
Jagsir Singh [22]	0.855	0.769	0.949	0.849
Zhenlong Yuan [5]	0.966	0.966	0.966	0.966
Xi Xiao [9]	0.923	0.939	0.905	0.922
Bojan Kolosnjaji [12]	0.894	0.856	0.894	—
Rakshit Agrawal [6]	0.954	0.963	0.951	—
Ours	**0.975**	**0.979**	**0.976**	**0.978**

information of the API call sequence. For the method of intercepting a part of dynamic API call sequence [6, 9, 12] may lose malicious behaviour, our method not only preserves the complete sequence but also preserves the order of the sequence. Therefore, our method obtains better detection accuracy compared to other methods.

6. Conclusion

In this paper, we propose a new framework for detecting Android malicious applications. First, we transform the multimillion-length dynamic API call sequence into a directed and edge-weighted function call graph with only 26 nodes, which can greatly reduce the amount of data, while preserving the order information of dynamic API call sequence. Then we use a double-layer GCN network to extract the behaviour information of the application from the heterogeneous function call graph and concatenate the results with the advanced features Permission. Finally, we feed the results into an LR model for malicious detection. The experimental results show that the proposed method of transforming API call sequence into a graph can significantly improve the time efficiency of detection, while the loss of precision is very small. We combined the two types of information through the fusion model, which was significantly better than other baseline models.

Data Availability

The data used to support the findings of this study are included within the article. The data presented in this study will be available upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant nos. 61672543 and 61772559.

References

[1] C. Secutiry, "Cyber secutiry report," 2021, https://www.ntsc.org/assets/pdfs/cyber-security-report-2020.pdf%20.

[2] Kaspersky, "Mobile malware evolution 2020," 2021, https://securelist.com/mobile-malware-evolution-2020/101029/.

[3] A. Arora, S. K. Peddoju, and M. Conti, "Permpair: android malware detection using permission pairs," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1968–1982, 2019.

[4] T. G. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, pp. 773–788, 2018.

[5] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, 2016.

[6] R. Agrawal, J. W. Stokes, M. Marinescu, and K. Selvaraj, "Neural sequential malware detection with parameters," in *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Calgary, AB, Canada, April 2018.

[7] M. Schofield, G. Alicioglu, R. Binaco et al., "Convolutional neural network for malware classification based on API call sequence," in *Proceedings of the 2021 the 14th International Conference on Network Security & Applications*, Computer Science & Information Technology (CS & IT), Zurich, Switzerland, January 2021.

[8] Y. Qiao, Y. Yang, L. Ji, and J. He, "Analyzing malware by abstracting the frequent itemsets in API call sequences," in *Proceedings of the 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, IEEE, Melbourne, VIC, Australia, July 2013.

[9] X. Xiao, S. Zhang, F. Mercaldo, G. Hu, and A. K. Sangaiah, "Android malware detection based on system call sequences and LSTM," *Multimedia Tools and Applications*, vol. 78, no. 4, pp. 3979–3999, 2019.

[10] W. Xie, S. Xu, S. Zou, and J. Xi, "A system-call behavior language system for malware detection using a sensitivity-based LSTM model," in *Proceedings of the 2020 3rd International Conference on Computer Science and Software Engineering*, Beijing, China, May 2020.

[11] Z. Zhang, P. Qi, and W. Wang, "Dynamic malware analysis with feature engineering and feature learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, NY, USA, February 2020.

[12] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*, Springer, Hobart, TAS, Australia, December 2016.

[13] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, South Brisbane, QLD, Australia, April 2015.

[14] W. Wang, J. Wei, S. Zhang, and X. Luo, "LSCDroid: malware detection based on local sensitive API invocation sequences," *IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 174–187, 2019.

[15] Y. Qin, W. Wang, S. Zhang, and K. Chen, "An exploit kits detection approach based on HTTP message graph," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3387–3400, 2021.

[16] T. Gu, C. Sun, X. Ma et al., "Practical GUI testing of Android applications via model abstraction and refinement," in *Proceedings of the IEEE/ACM 41st International Conference on*

Software Engineering (ICSE), IEEE, Montreal, QC, Canada, May 2019.

[17] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, Article ID 101663, 2020.

[18] M. Ali, S. Shiaeles, G. Bendiab, and B. Ghita, "MALGRA: machine learning and N-gram malware feature extraction and detection system," *Electronics*, vol. 9, no. 11, p. 1777, 2020.

[19] S. Jha, D. Prashar, H. V. Long, and D. Taniar, "Recurrent neural network for detecting malware," *Computers & Security*, vol. 99, Article ID 102037, 2020.

[20] D. Uppal, R. Sinha, V. Mehra, and V. Jain, "Malware detection and classification based on extraction of API sequences," in *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, Delhi, India, September 2014.

[21] Y. Fang, B. Yu, Y. Tang et al., "A new malware classification approach based on malware dynamic analysis," *Australasian Conference on Information Security and Privacy*, Springer, Cham, 2017.

[22] J. Singh and J. Singh, "Assessment of supervised machine learning algorithms using dynamic API calls for malware detection," *International Journal of Computers and Applications*, vol. 44, no. 3, pp. 270–277, 2022.

[23] J. Kim, S. Lee, J. M. Youn, and H. Choi, "A study of simple classification of malware based on the dynamic api call counts," in *Proceedings of the Advances in Computer Science and Ubiquitous Computing*, pp. 944–949, Bangkok, Thailand, December 2016.

[24] androguard, "reverse engineering, malware and goodware analysis of android applications," 2020, https://github.com/androguard/androguard/.

[25] oracle. Java, "Debug wire protocol," 2020, https://docs.oracle.com/javase/8/docs/technotes/guides/jpda/jdwpspec.html.

[26] S. Arzt, S. Rasthofer, and E. Bodden, "Susi: A Tool for the Fully Automated Classification and Categorization of Android Sources and Sinks," Rep. TUDCS-2013-0114, University of Darmstadt, Darmstadt, Germany, 2013.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 4748528, 21 pages
https://doi.org/10.1155/2022/4748528

Research Article

An Enhanced Intrusion Detection System for IoT Networks Based on Deep Learning and Knowledge Graph

Xiuzhang Yang,[1,2] Guojun Peng,[1,2] Dongni Zhang,[1,2] and Yangqi Lv[1,2]

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

Correspondence should be addressed to Guojun Peng; guojpeng@whu.edu.cn

Received 24 December 2021; Revised 17 February 2022; Accepted 18 March 2022; Published 26 April 2022

Academic Editor: Weizhi Meng

Nowadays, the intrusion detection system (IDS) plays a crucial role in the Internet of Things (IoT) networks, which could effectively protect sensitive data from various attacks. However, the existing works have not considered multiview features fusion and failed to capture the semantic relationships among the anomalous requests. They are not robust and cannot detect the attack types in real-time. This paper proposes a lightweight intrusion detection system based on deep learning and knowledge graph. First, our system extracts semantic relationships and key features by knowledge graph and statistical analysis. Then, IoT network requests are converted into word vectors through multiview feature fusion and feature alignment. Finally, an attention-based CNN-BiLSTM model is designed to identify malicious request attacks, which can capture long-distance dependence and contextual semantic information. Experiment results show that the proposed model significantly outperforms the existing solution in the robustness of the model. Moreover, it can select more critical features for IDS to achieve better accuracy and lower the false alarm rate. Compared with the state-of-the-art systems, the proposed IDS achieves a higher detection accuracy of 90.01%. In addition, our system can detect various stealthy attack types (including DoS, Probe, R2L, and U2L) and extract semantic relationships among features.

1. Introduction

With various terminal devices and applications becoming interoperable through networks, the Internet of Things (IoT) systems improve the quality of life and enhance real-life intelligent devices [1]. The prominent devices in IoT networks include outdoor surveillance cameras, smart home devices, mobile user-worn devices, and industrial control services [2]. As a result, IoT devices are increasingly deployed in critical infrastructures. However, due to the intuitive design, IoT also faces many security threats and challenges, making the infrastructures suffer from serious attacks and undermining the integrity of sensitive data. Some incidents (e.g., Stuxnet [3], the widespread blackout in the Ukrainian power grid [4], SolarWinds supply chain attack [5], Colonial Pipeline suffering the ransomware attack [6]) have shown that network attacks on IoT devices can result in catastrophic consequences to the society, enterprises, and human life [7, 8]. Thus, effectively detecting security threats and protecting the IoT environment is crucial.

As an active defense technology, intrusion detection system (IDS) has gradually become a key technology to ensure network security. IDS can identify abnormal requests in the communication network, detect potential network threats, and generate alarms, thereby protecting the Internet and infrastructures in runtime. In this paper, we focus on constructing IDS for IoT networks. Meanwhile, machine learning (ML) has been widely used in security research recently, such as APT attack detection [9], personal privacy protection [10], malicious code analysis [11], cyberphysical system defense [8], and malicious traffic detection [12]. In order to detect malicious behaviors in large-scale network traffic of IoT, machine learning-based IDS has attracted widespread attention. The goal of ML-based IDS is to learn a decision boundary that discriminates malicious network traffic from normal network traffic. In this process, it learns to distinguish different behaviors from the network traffic and host audit records. Traditional ML-based intrusion detection algorithms include support vector machine (SVM) [13], K-Nearest Neighbors (KNN) [14], Decision Tree (DT) [15], Random Forest (RF) [16], AdaBoost [17], K-Means [18], and Artificial Neural Networks (ANNs) [19].

However, these systems usually require a large amount of expert knowledge and labeled data to construct rules or models to detect abnormal requests [8]. Hence, they have poor applicability in large-scale intrusion detection systems. Also, traditional ML-based systems have not considered multiview features fusion and fail to capture the semantic relationships among the anomalous requests. The existing IDSs focus on constructing machine learning models and lack detailed feature extraction analysis [20], resulting in a poor ability to capture crucial features of IoT network traffic and detect unknown attack types (e.g., the zero-day attack). In addition, many machine learning algorithms are shallow learning algorithms, which have low detection throughput and suffer from evasion attacks that can easily bypass IDSs. Thus, they cannot efficiently defend IoT environments against various network attacks, especially when facing large-scale data [21]. They cannot detect the attack types in real time. Therefore, improving the design of current IDSs to provide suitable intrusion detection for IoT networks is urgent. Also, some features of different types of attacks often appear in network traffic at the same time, and traditional IDS ignores the semantic knowledge between features. Therefore, effectively capturing semantic-semantic relationships will help improve the accuracy and robustness of our IDS.

To address the above problems, this paper proposes an enhanced intrusion detection system for IoT networks based on deep learning and knowledge graph. The main research problem in this paper is how to build a model to better learn the semantic relationship between network traffic features, so as to improve the robustness of IDS to traffic recognition in IoT networks. We mainly address this problem from the perspective of feature extraction and feature fusion. Therefore, we particularly devise two feature extraction methods: knowledge graph-based feature extraction and statistical analysis-based feature extraction. By this means, we can effectively extract the contextual semantic relationship of IoT network traffic and key features of different attack types, thereby increasing the similarity distance between benign and malicious samples. Furthermore, to solve feature ambiguity and the single perspective of traditional models, we design a multiview feature fusion and feature alignment algorithm to build more robust word vectors and construct a deep learning model to detect IoT requests. The contributions of our paper are summarized as follows:

(i) To our best knowledge, we are the first to design a multiview fusion model for intrusion detection of IoT networks. This model can extract the contextual semantic relationship through a knowledge graph, as well as crucial features through statistical analysis. Then, we make the IDS more robust and interpretable by feature fusion and alignment of the two feature extraction algorithms.

(ii) We present an attention-based Convolutional Neural Network and Bidirectional Long Short-term Memory (CNN-BiLSTM) model, which can capture the long-distance dependence and contextual semantic information. Moreover, our IDS relies on less prior knowledge and detects more attack types. The deep learning-based classifier can better detect attack types in runtime by adaptively updating the weights of different features, especially the features extracted by the knowledge graph.

(iii) We conduct a systematic comparison experiment with state-of-the-art systems. The results show that our system is effective and robust in detecting malicious requests of IoT and can accurately detect various stealthy attack types (including unknown attacks).

The rest of the paper is organized as follows. Section 2 summarizes the related works. Section 3 presents the proposed intrusion detection model for IoT networks in detail. Section 4 experimentally evaluates the performances of our model and Section 5 concludes the paper. Also, Table 1 summarizes all acronyms used in this paper for completeness and readability.

2. Related Work

In this section, we introduce the related work of intrusion detection systems for IoT networks, including rule-based systems, machine learning-based systems, and deep learning-based systems. Note that we introduce from the perspective of methodology, rather than discuss the following host-based IDS and network-based IDS.

2.1. Rule-based System. With the increasing complexity and stealthy of network attacks, intrusion detection system plays a significant role in detecting malicious requests. Traditional intrusion detection system mainly relies on expert knowledge and manual experience to extract traffic features by defining rules and feature databases. Then, such IDS analyzes network requests with statistical correlation algorithms to identify external attacks [22]. In 1987, Denning [23] first proposed a universal intrusion detection system and introduced it into computer security defense systems. Later, Lunt and Jaganna [24] designed an intrusion detection expert system based on domain knowledge. This system adaptively learned the normal behavior of each user through audit records and detected abnormal users in real time. After that, more rule-based intrusion detection systems appear, especially in the IoT ecosystem. Chimera [25] is a declarative query language for network traffic processing that combines the advantages of intrusion detection systems and SQL syntax with rules. The vNIDS [26] is an innovative architecture for network intrusion detection systems (NIDSs),

Table 1: List of the acronyms used in the manuscript.

Acronym	Definition	Acronym	Definition
ANN	Artificial Neural Network	ML	Machine Learning
BiLSTM	Bidirectional Long Short-Term Memory	M2-DAE	MultiModal Deep AutoEncoder
CNN	Convolutional Neural Network	NIDS	Network Intrusion Detection Systems
DL	Deep Learning	NIN	Network-in-Network
DoS	Denial of Service	RDP	Remote Desktop Protocol
DT	Decision Tree	ReLU	Rectified Linear Unit
FAR	False Alarm Rate	RF	Random Forest
FN	False Negative	RNN	Recurrent Neural Network
FP	False Positive	R2L	Remote to Local
IDS	Intrusion Detection System	SA	Statistical Analysis
IoT	Internet of Things	SVM	Support Vector Machine
KG	Knowledge Graph	TN	True Negative
KNN	K-Nearest Neighbors	TP	True Positive
LR	Logistic Regression	U2R	User-to-Root
LSTM	Long Short-Term Memory		

which achieves elastic security by configuring virtual NIDS as microservices and employing program slicing to partition the detection logic programs. Haugerud et al. [27] designed and implemented a lightweight elastic architecture NIDS, which constructed a flexible IDS in a docker container through network function virtualization and intelligent rule ordering.

However, rule-based systems rely heavily on various rules and expert experience, which will constantly update rules. Since massive network attacks with stealth and anti-detection, malicious requests often cannot be detected in time. In addition, the system ignores the semantic relationship between traffic features, making it easy for specific constructed and confused attack requests to bypass online firewalls or IDSs. Therefore, it cannot detect unknown attacks (e.g., zero-day attacks).

2.2. Machine Learning-Based System. With the rapid advancement in statistical machine learning, machine learning-based intrusion detection system is widely deployed to protect IoT networks from various attacks [28]. This system uses the established and trained machine learning model to predict the malicious behavior of unknown network requests. Common methods include SVM, random forest, logistic regression, and K-Means. Whisper [21] is a real-time ML-based malicious traffic detection system utilizing frequency domain features to achieve high accuracy and throughput. Also, Whisper has good robustness because attackers cannot easily interfere with frequency domain features. Bitton and Shabtai [29] propose an intrusion detection algorithm based on clustering. This algorithm can protect the cyberphysical system from Remote Desktop Protocols (RDP) vulnerability attacks. Li et al. [30] design a model based on sustainable ensemble learning, which considers the accumulation of historical knowledge to realize incremental learning and improve the robustness of intrusion detection systems.

Compared with rule-based intrusion detection systems, machine learning-based systems can improve the model's accuracy and robustness, which is no longer limited to specific rules. However, because machine learning belongs to shallow learning, it lacks deep semantic knowledge and context analysis. Therefore, traditional IDS based on machine learning has some limitations in detecting complex, confusing, and hidden massive malicious requests and cannot detect the attack the first time.

2.3. Deep Learning-Based System. With the application of deep neural networks in various security fields [31], intrusion detection systems based on deep learning appear gradually. Representative models are mainly convolutional neural networks and recurrent neural networks (including LSTM). On the one hand, they can mine the deep information of network requests and improve detection accuracy through continuous learning of the model. On the other hand, they can avoid the limitations caused by artificial experience. LIO-IDS [32] is an intrusion detection system based on the LSTM classification model and promotion one-to-one technology, which is used to deal with infrequent network intrusions. Imrana et al. [33] developed a centralized intrusion detection system based on BiLSTM to identify attack types with fewer samples. DL-IDS [34] mixed CNN and LSTM models for intrusion detection. Li et al. [35] designed an IoT feature extraction and intrusion detection system for smart cities based on the deep migration learning model. Balakrishnan et al. [36] leveraged a deep belief network to enhance intrusion detection systems and scrutinize malicious activity in the IoT network. Moreover, BDL-IDS [37] was a big data-aware deep learning system, which can reduce the number of false alarms in the NSL-KDD dataset. Kasongo and Sun [38] proposed a feed-forward deep neural network wireless IDS system using a wrapper-based feature extraction unit. Ferrag et al. [39] compared the cybersecurity intrusion detection systems and datasets based on deep learning in detail and conducted a comparative study.

Furthermore, to solve the problem that existing neural networks need supervised training and label network traffic with the help of expert knowledge, Mirsky et al. [40] proposed an unsupervised online network intrusion detection system called Kitsune. This system can distinguish normal and abnormal traffic patterns by building an ensemble of autoencoders. Bovenzi et al. [41] proposed a two-stage hierarchical network intrusion detection system, namely H2ID. The system performed anomaly detection via a novel lightweight solution based on MultiModal Deep AutoEncoder (M2-DAE) and achieved attack classification by soft-output classifiers.

However, the above systems lack the deep semantic relationship among traffic features and fail to analyze the correlation and difference between attack types through feature extraction and feature fusion. In addition, the lack of knowledge mapping to capture semantic information makes it impossible to sense hidden, unknown attacks (with small samples) and spoofing malicious traffic. Table 2 compares the characteristics of the three intrusion detection models in detail. It can be seen that the system in this paper can effectively extract semantic features through a knowledge graph, has better robustness, and can detect unknown types of attacks in real time.

3. System Design

In this section, we present the architecture of our proposed intrusion detection system for IoT networks and introduce how to extract features by knowledge graph and statistical analysis. Then, we describe the design and implementation of this IDS in detail.

3.1. Overview. The overall architecture of our IDS is depicted in Figure 1. This IDS consists of five phases: data preprocessing, feature extraction, feature fusion and alignment, model construction, and intrusion detection.

The proposed intrusion detection system is designed to increase the difference between normal and abnormal requests. As shown in Figure 1, our detection framework is comprised of five main phases. (1) The data preprocessing component can improve the quality of the dataset by processing the IoT network requests, which includes data cleaning, numerical conversion, log processing, normalized processing, and one-hot encoding. (2) The feature extraction component consists of two parts. First, the semantic relationship of different attack types is extracted through semantic feature analysis, cooccurrence calculation, and knowledge graph construction. Then, the key features of normal and malicious requests are mined through statistical analysis and power-law distribution. (3) The feature fusion and alignment component judges whether the feature should be weighted, aiming to improve the robustness and interpretability of the IDS by multiview fusion. In addition, this step uses the word embedding module to convert traffic features into word vectors. (4) The model construction component presents an attention-based CNN-BiLSTM model to capture the long-distance and contextual semantic features. (5) The intrusion detection component constructs a classifier through the softmax function, thereby detecting malicious requests (i.e., binary classification) and identifying attack types (i.e., multiclass classification).

3.2. Data Preprocessing. To obtain a high-quality dataset and solve the constraint problem of nonnumerical features, this paper carries out detailed data preprocessing to ensure the IoT intrusion detection experiment. This part mainly includes five processing, namely, data cleaning, numerical conversion, log processing, normalized processing, and one-hot encoding. The following part explains each phase in detail.

Note that we apply the NSL-KDD dataset in our experiment, which is an improvement of the KDD Cup 99 dataset [42]. This dataset is the traffic request generated by MIT Lincoln Laboratory using IoT devices to simulate attacks in the real-time environment. Thus, it is a relatively authoritative dataset. The entire NSL-KDD dataset can be described in Figure 2. It includes 41 traffic attribute features and one label feature. Among them, the attribute feature contains four categories: intrinsic feature, content feature, time-based feature, and host-based feature, which can be formulated as (1), where x_i corresponds to the feature i. Moreover, the label feature is divided into normal network traffic and abnormal network traffic. The abnormal network traffic includes four types (i.e., DoS, Probe, R2L, and U2R), covering a total of 39 different attack types (e.g., Nmap, Smurf, Sqlattack, and Spy). The detailed feature description and data distribution will be introduced in the following experimental section.

$$X = (x_1, x_2, \ldots, x_{40}, x_{41}). \tag{1}$$

Data cleaning can standardize the dataset and remove redundant data. Typical operations include filling in gaps and deleting duplicate values. In this paper, data cleaning is performed on both the training set and the testing set. Since the deep learning model requires a vector of real numbers as input embedding, it is necessary to convert symbolic features into numerical features. Thus, we perform numeric conversion processing, encoding some strings from 0 to convert them to the corresponding numeric value. For example, the types of protocol (i.e., TCP, UDP, and ICMP) will be converted into corresponding numeric values (i.e., 0, 1, and 2). In the case of the NSL-KDD dataset, the nonnumeric features that require numerical transformation are protocol type, service type, flag, and category type.

To enhance the performance of the proposed IDS, we will execute log processing and normalized processing. The former can effectively reduce the difference among traffic features in the dataset. For example, the NSL-KDD dataset counts the size of requests sent from the source host to the target host, denoted as src_bytes, and the maximum value is 1,379,963,888. Obviously, this value is an outlier, which will seriously affect our experimental results. Therefore, we need to execute the log function to reduce dimensionality. By this means, it can reduce feature values to the same or similar granularity, and the processed result is 9.1399. In this paper, five features (i.e., duration, src_bytes, dst_bytes, num_compromised, and num_root) of the NSL-KDD dataset will be processed by the log function, whose calculation is shown in equation (2). The results are shown in Table 3.

Table 2: Comparing the existing intrusion detection systems.

Category	Related work	Techniques	Robust detection	Knowledge graph	Real-time detection	Unknown attack detection	Semantic feature extraction
Rule-based System	Chimera [25]	Rules SQL syntax	✗	✗	✓	✗	✗
	vNIDS [26]	virtual NIDS	✗	✗	✓	✗	✗
	Haugerud et al. [27]	rule ordering virtualization	✗	✗	✓	✗	✗
ML-based System	Whisper [21]	Clustering frequency domain	✓	✗	✓	✓	✗
	Bitton et al. [29]	Clustering T-SNE	✓	✗	✓	✗	✗
	Li et al. [30]	ensemble learning	✓	✗	✓	✗	✗
DL-based System	LIO-IDS [32]	LSTM	✓	✗	✓	✗	✗
	Imrana et al. [33]	BiLSTM	✓	✗	✓	✗	✗
	DL-IDS [34]	CNN-LSTM	✓	✗	✓	✓	✗
	Our IDS	KG + SA CNN BiLSTM	✓	✓	✓	✓	✓

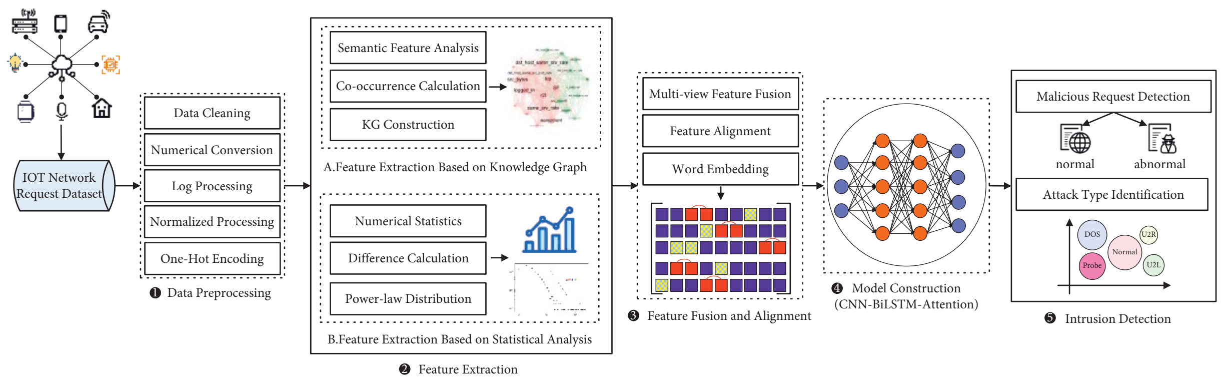
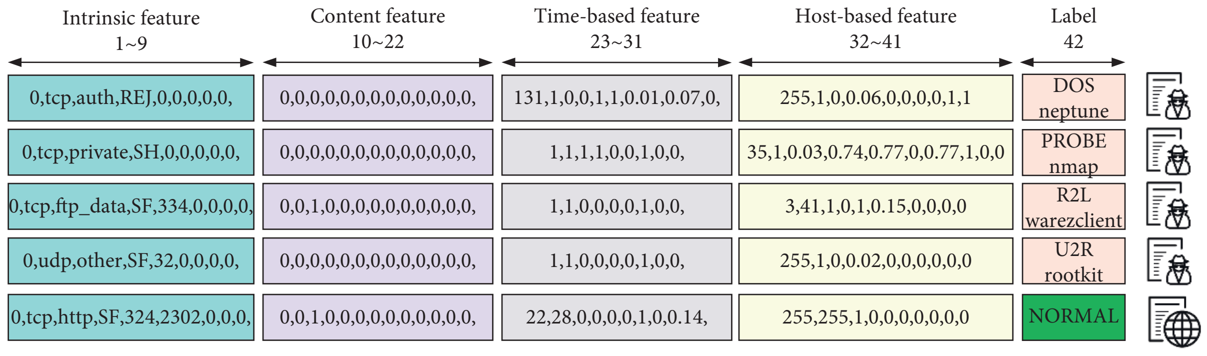

IOT Network Request Dataset
Data Cleaning
Numerical Conversion
Log Processing
Normalized Processing
One-Hot Encoding
❶ Data Preprocessing
Semantic Feature Analysis
Co-occurrence Calculation
KG Construction
A.Feature Extraction Based on Knowledge Graph
Numerical Statistics
Difference Calculation
Power-law Distribution
B.Feature Extraction Based on Statistical Analysis
❷ Feature Extraction
Multi-view Feature Fusion
Feature Alignment
Word Embedding
❸ Feature Fusion and Alignment
❹ Model Construction (CNN-BiLSTM-Attention)
Malicious Request Detection
normal
abnormal
Attack Type Identification
DOS
Normal
U2R
Probe
U2L
❺ Intrusion Detection

Figure 1: The architecture of our proposed intrusion detection system for IoT.



Intrinsic feature 1~9	Content feature 10~22	Time-based feature 23~31	Host-based feature 32~41	Label 42
0,tcp,auth,REJ,0,0,0,0,0,	0,0,0,0,0,0,0,0,0,0,0,0,0,	131,1,0,0,1,1,0.01,0.07,0,	255,1,0,0.06,0,0,0,0,1,1	DOS neptune
0,tcp,private,SH,0,0,0,0,0,	0,0,0,0,0,0,0,0,0,0,0,0,0,	1,1,1,1,0,0,1,0,0,	35,1,0.03,0.74,0.77,0,0.77,1,0,0	PROBE nmap
0,tcp,ftp_data,SF,334,0,0,0,0,	0,0,1,0,0,0,0,0,0,0,0,0,0,	1,1,0,0,0,0,1,0,0,	3,41,1,0,1,0.15,0,0,0,0	R2L warezclient
0,udp,other,SF,32,0,0,0,0,	0,0,0,0,0,0,0,0,0,0,0,0,0,	1,1,0,0,0,0,1,0,0,	255,1,0,0.02,0,0,0,0,0,0	U2R rootkit
0,tcp,http,SF,324,2302,0,0,0,	0,0,1,0,0,0,0,0,0,0,0,0,0,	22,28,0,0,0,0,1,0,0.14,	255,255,1,0,0,0,0,0,0,0	NORMAL

Figure 2: The description of the features in the NSL-KDD dataset.

Table 3: The result of log processing in the NSL-KDD dataset.

No.	Feature	Scope before processing	Scope after processing
1	duration	[0, 58329]	[0, 4.7659]
5	src_bytes	[0, 1379963888]	[0, 9.1399]
6	dst_bytes	[0, 1309937401]	[0, 9.1173]
13	num_compromised	[0, 7479]	[0, 3.8738]
14	num_root	[0, 7468]	[0, 3.8732]

$$x_i' = \log(x_i). \tag{2}$$

Moreover, data normalization converts the feature value into a suitable range, which can effectively eliminate the problem of data imbalance and preference for larger values. This paper mainly uses the Min-Max scaling to normalize the values of different features between 0 and 1, and the Min-Max scaling is described as

$$x_{ij}' = \frac{x_{ij} - x_i^{\min}}{x_i^{\max} - x_i^{\min}}, \tag{3}$$

where i is a feature in the dataset, j is a record of the dataset, and $x_i^{\max}$ and $x_i^{\min}$ are the maximum and minimum values of the feature i. Hence, we can map every continuous feature's values within the range of (0, 1) and effectively characterize the importance and distribution of the corresponding features. However, to preserve the authenticity and relationship of the original network requests, this paper does not call the Z-score function for standardized processing.

Finally, we perform the one-hot encoding to convert the category label of the NSL-KDD dataset to a unique category, which assigns the current category bit to 1 and other low bits to 0. For example, the DoS label in Figure 2 will be converted to the form [1, 0, 0, 0, 0]. By this, our deep learning model can receive better input vectors for training. In short, we show how the proposed IDS implements data preprocessing in five steps, and then, we will introduce the feature extraction part in detail (the second step in Figure 1).

3.3. Feature Extraction Based on Knowledge Graph. Without prior knowledge, it is difficult to obtain the correlation between features, especially facing unknown network attacks. Before training, it is important for our IDS model to reduce the noise of redundant features and to mine the semantic relationships. Therefore, this paper proposes a feature extraction method based on the knowledge graph. The method can construct a knowledge graph through semantic feature analysis and cooccurrence calculation. By this, it can extract the critical feature pairs of normal and abnormal requests (including four types of attacks) of the NSL-KDD dataset. Algorithm 1 shows the feature extraction method based on the knowledge graph, which is designed to select crucial feature pairs of different types.

In Algorithm 1, the output is the set M, which covers the four categories (i.e., intrinsic feature, content feature, time-based feature, and host-based feature) of the NSL-KDD dataset, and completes the intrusion detection classification tasks from a global perspective. Through this processing, we can effectively count out the semantic relations of the representative features of different attack types, thereby providing better support for subsequent classification. For example, the <scr_bytes, dst_host_same_srv_rate> and <tcp, same_src_rate> features often exist in R2L attacks at the same time.

3.4. Feature Extraction Based on Statistical Analysis. The knowledge graph highlights the semantic relationship of features and improves the robustness of intrusion detection by calculating features that often appear in pairs in the same attack type. However, some individual features also make an important contribution to intrusion detection systems. For instance, in the Probe attack of the NSL-KDD dataset, the Nmap command is repeatedly called to perform port or IP scanning. Thence, the corresponding number of network requests or certain features exceeds other types of attacks, even reaching a threshold. To this end, we propose a feature extraction method based on statistical analysis. This method uses mathematical statistics to calculate the average, median, and mode of different features in the training set. Further, this paper selects the average as the threshold of statistical features, and the difference between normal requests and abnormal requests is calculated as shown in

$$d_i = \sum_{i=1}^{N} \sum_{j=1}^{N} \left| \overline{x_i} - \overline{x_j} \right|. \tag{4}$$

In (4), $\overline{x_i}$ and $\overline{x_j}$ represent the average value of feature x's two classification types (i.e., i and j). The entire equation calculates a sum of the average differences of different features in network requests. The number N represents five types (normal, DoS, Probe, R2L, and U2R) in the NSL-KDD dataset. Through the above processing, we can maximize the differences between various categories. Then, we use the sorting algorithm to process the four feature categories (i.e., intrinsic feature, content feature, time-based feature, and host-based feature) and identify the two largest values of each category. Finally, eight key features based on statistical analysis are formed, which can further enhance the performance of our IDS. In addition, in the later deep learning classification model, when these features exceed a defined threshold (shown in the previous calculation), they are weighted to increase the gradient descent speed of the neural network.

Moreover, we use the power-law distribution to analyze the NSL-KDD dataset statistically, as shown in (5). As a result, the whole data presented a long-tail distribution, which often appears in some specific cyberattacks. Also, this stage further verifies that the intrusion detection system conforms to the social law and can map the crucial features from the statistical view.

$$P(k) \sim Cx^{-\alpha}. \tag{5}$$

3.5. Feature Fusion and Alignment. In this module, we design and implement a multiview feature fusion and alignment method. In recent years, multimodel deep learning methods have been gradually applied to intrusion detection. Aceto et al. [43] proposed a novel multimodal deep learning framework for encrypted traffic classification, named MIMETIC. The method can improve the performance of mobile traffic classification by learning intramodality and intermodality dependencies, thereby utilizing complementary views to identify traffic. Bu et al. [44] designed a neural network model with deep and parallel network-in-network (NIN) structures for classifying encrypted network traffic.

```
     Input: Feature vector set $D = \{x_{ij}\}$ of the dataset, where $i$ is the $i^{th}$ network request, $j$ is the $j^{th}$ feature. Feature set
     $F = \{f_i, i = 1, 2, \ldots, n\}$.
     Output: The selected feature subset $M = \{\langle f_i, f_j \rangle\}$.
(1)  Initialization: set $M = \varnothing$.
(2)  for each $x_{ij} \in D$ do
(3)      Feature conversion by $v_{ij} = \begin{cases} 0, \text{ if } x_{ij} \le 0 \\ 1, \text{ if } x_{ij} > 0 \end{cases}$
(4)  end for
(5)  for each $f_i \in F$ do
(6)      Calculate pairwise feature pairs and judge whether the features cooccur.
(7)      $\text{NumPairs}(\langle f_i, f_j \rangle) = \begin{cases} n_{ij} + 0, \text{ if there is no cooccurrence relationship} \\ n_{ij} + 1, \text{ if features } f_i \text{ and } f_j \text{ are cooccurring} \end{cases}$
(8)      end for
(9)  for each $\langle f_i, f_j \rangle$ from NumPairs do
(10)     Use feature semantic cooccurrence relationship to construct a knowledge graph.
(11)     (a) Build an entity feature set $E = \{e_1, e_2, \ldots, e_m\}$.
(12)     (b) Build a relational feature set $R = \{r_{ij}, i, j = 1, 2, \ldots, m\}$, where $r_{ij}$ is the weight of entity $e_i$ and $e_j$.
(13)     (c) Calculate the cooccurrence threshold set $T$ of different types of features.
(14)     (d) Build a knowledge graph of the IDS dataset.
(15) end for
(16) for each feature category do
(17)     for each classification type do
(18)         (a) Calculate the critical feature pairs of four features (i.e., intrinsic feature, content feature, time-based feature, and host-
     based feature) by sorting algorithm, which covers all classification types (e.g., DoS, Probe, R2L, U2R).
(19)         (b) $M = \text{TopFeaturePairs}(\langle f_i, f_j \rangle)$
(20)     Output key feature pairs based on the knowledge graph.
     return $M$
```

ALGORITHM 1: Feature extraction based on the knowledge graph.

NIN can adopt a micronetwork after each convolutional layer to enhance local modeling and improve classification accuracy.

However, the above methods only learn features by fusing or combining different deep learning models, ignoring the feature extraction and feature fusion of network traffic in intrusion detection. Also, their time overhead and model complexity are high. Compared with these methods, this paper implements feature fusion and feature alignment in the feature extraction stage. This multiview feature fusion method belongs to a lightweight intrusion detection feature preprocessing, which effectively combines the advantage of two feature extraction methods. On the one hand, a knowledge graph-based feature extraction algorithm can mine the semantic relevance between features and extract four feature pairs. On the other hand, a statistical analysis-based feature extraction algorithm can select a single feature, which makes an essential contribution to our IDS and extracts eight key features.

Actually, the feature extraction from the two views proposed in this paper is better than the traditional IDS. The latter tends to ignore the correlation between features and select features only from a single view, such as using statistical features or weak correlation algorithms (e.g., principal component analysis or statistical frequency).

Next, we need to introduce multiview feature fusion and feature alignment processing for these features so that the following deep learning model can be better transformed into input word vectors. The whole calculation process is shown in Algorithm 2. The algorithm steps are as follows. (1) Convert the feature pairs identified by the knowledge graph into sequences, and perform alignment and deduplication processing. (2) Take the union operation on the features extracted from the knowledge graph and statistical analysis, and perform feature combination. (3) Assign different weights to the features of the two types of fusion, which will be used as the initial weights of the neural network model.

Algorithm 2 generates the weight set W through feature fusion and feature alignment. Then, we use word embedding to convert the network traffic into word vectors, which is the input of neural networks. Next, the set W will initialize the weight parameters of the corresponding feature vectors. In other words, if feature f_i belongs to the critical feature selected by our feature extraction algorithm (i.e., from the set U), the weight corresponding to the feature will be multiplied by for addition. Otherwise, the other weight remains unchanged. Through the above processing, we can improve the robustness of IDS and increase the gradient descent speed of neural networks. Thus, this stage will significantly contribute to the following intrusion detection and malicious request classification tasks.

3.6. Model Construction. Now, we construct an attention-based CNN-BiLSTM algorithm to learn the traffic features obtained from the proposed feature extraction and feature fusion module. In this model, CNN is similar to a feature extractor, which can reduce computing resources and is suitable for IoT devices (with limited resources). Moreover, LSTM performs classification operations based on the serialized feature information given by CNN, similar to a classifier. For example, if only one TCP handshake packet is

```
     Input: Feature set $F = \{f_i, i = 1, 2, \ldots, n\}$. The feature subset $M = \{\langle f_i, f_j \rangle\}$ is selected by a knowledge graph. The feature subset
     $F_{SA} = \{f_k, k = 1, 2, \ldots, m\}$ is selected by statistical analysis.
     Output: $W = \{w_i, i = 1, 2, \ldots, n\}$ is the weight parameter to be multiplied.
 (1) Initialization: set $W = \varnothing$.
 (2) Perform feature alignment on the feature pair $M$, and convert it into a sequence.
 (3) for each $\langle f_i, f_j \rangle \in M$ do
 (4)     $F_{KG}$ = FeaturesOfKG.append$(f_i, f_j)$
 (5)     $F_{KG}$ = AlignmentBySort$(F_{KG})$
 (6) end for
 (7) Fuse the features extracted from the two views of KG and SA.
 (8) $U = F_{KG} \cup F_{SA}$
 (9) for each $f_i \in F$ do
(10)     Weight calculation by $w_i = \begin{cases} w_i & \text{if } f_i \notin U \\ \alpha \bullet w_i & \text{if } f_i \in U \end{cases}$
(11) end for
(12) Output the weight set $W$.
     return $W$
```

Algorithm 2: A multiview feature fusion and feature alignment approach.

analyzed in intrusion detection, it is difficult to judge whether it is a port scan. However, when multiple data packets are serialized and the LSTM network is used for judgment, it can be more accurately judged that these data packets are from port scanning attacks, with the reason that the LSTM network can learn context information and serialized features. Thus, this paper designs an attention-based CNN-BiLSTM model to reduce the computing resources and improve the result of IDS in IoT networks. Figure 3 summarizes the architecture of the attention-based CNN-BiLSTM model, which consists of six phases: (1) word embedding layer, (2) convolutional layer, (3) pooling layer, (4) BiLSTM layer, (5) attention mechanism layer, and (6) fully connected layer.

3.6.1. Word Embedding Layer. We perform word embedding processing on the extracted features and weight information. In this paper, Word2Vec is used to implement word embedding processing. The feature information of network traffic is converted into a vector, which will be used as the input of the subsequent deep learning model. Therefore, the attention-based CNN-BiLSTM model can be trained more quickly, making our model learn the feature distribution efficiently.

3.6.2. Convolutional Layer. Convolutional Neural Network (CNN) is designed to extract local features by scrolling the convolution kernel. In IoT intrusion detection, a CNN model can effectively highlight the key features of network requests. In this paper, we construct a convolutional layer to extract import elements (e.g., some features that significantly impact DoS attacks). The convolution kernel convolves the input word vector matrix, and its filter will select different features (step 2 in Figure 3). The convolutional layer calculation is as shown in

$$h_i^d = f\left(w_d \cdot V_i + b_d\right), \quad (6)$$

where V_i is a word vector of network request feature in (6), and $V_i \in R^{n\times k}$, n is the number of features, k is the dimension of the word vector, w_d is a convolution kernel of size d, and b_n is the bias vector. Here, f is an activation function (e.g., ReLU), and the obtained feature is denoted as h_i^d. After the convolution processing, the local feature set H is obtained by mapping, and we can write the following equation:

$$H_d = \left\{h_1^d, h_2^d, \ldots, h_{n-d+1}^d\right\}. \quad (7)$$

3.6.3. Pooling Layer. In this stage, we sample the output vector of the convolution process and calculate the optimal solution of local features, thereby reducing the dimension of our features and maintaining the core features of the network traffic (step 3 in Figure 3). This paper uses Max Pooling technology to pool features as (8), which can calculate the most critical feature of malicious or normal requests.

$$M_i = \max\{H_d\}. \quad (8)$$

After the pooling layer extracts essential features, the obtained features are added to the subsequent network model. Finally, the output vector S formed by the combination of this step is defined by

$$S = \{M_1, M_2, \ldots, M_n\}. \quad (9)$$

3.6.4. BiLSTM Layer. Long Short-Term Memory (LSTM) is a variant of Recurrent Neural Network (RNN), a sequence processing model. The model retains the key information and forgets the secondary information through the memory unit and gate structure. A Bidirectional LSTM (BiLSTM) model comprises a forward LSTM and a backward LSTM, which encodes features from the front and back directions. This paper uses BiLSTM to extract contextual semantic information and capture the long-distance dependence of network traffic (as shown in Figure 3). Taking DoS attacks as an example, this model can effectively identify the

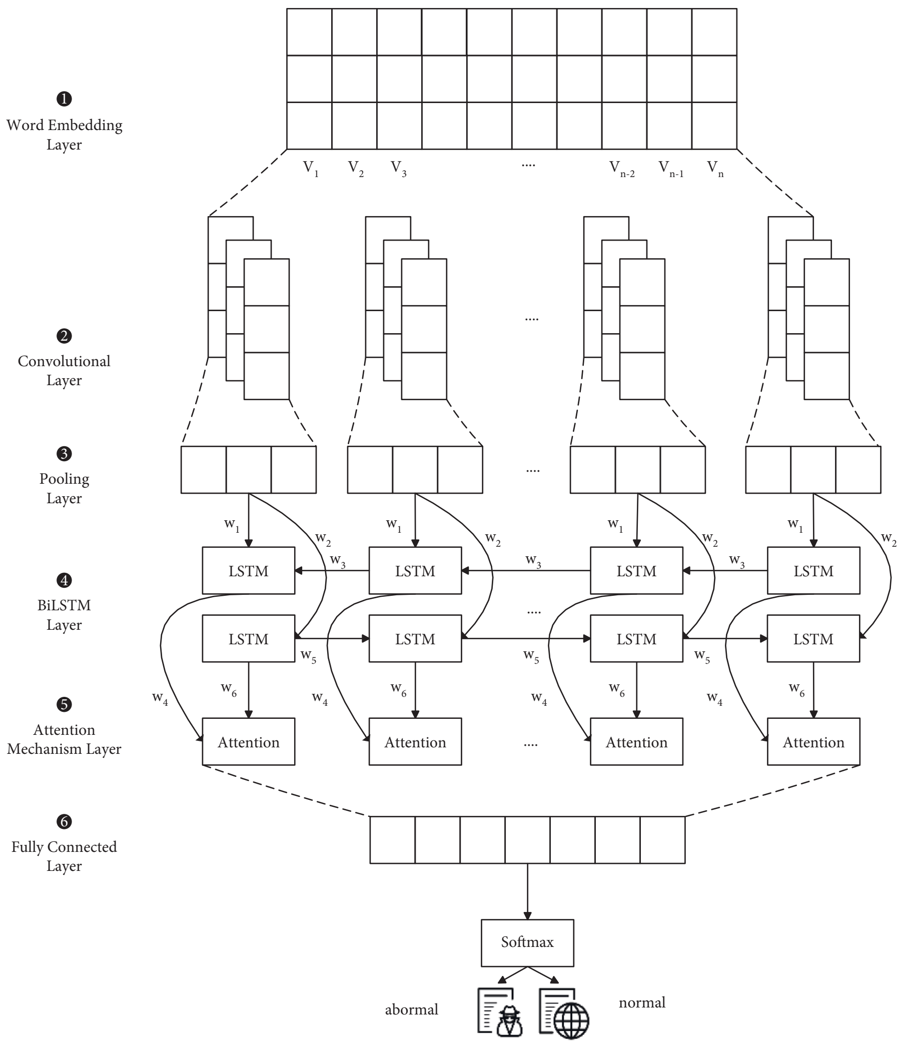


Figure 3: The architecture of an attention-based CNN-BiLSTM model.

relationship between the number of host connections and a specific flag feature of the connection rejection error (e.g., REJ). Obviously, this processing can maintain contextual semantic information, and the BiLSTM is a coarse-grained intrusion detection model. In this paper, the S vectors processed by the CNN model are used as the input of the BiLSTM model. They will be connected to form the CNN-BiLSTM model and complete the IoT intrusion detection task. The structure of BiLSTM includes the forget gate, input gate, output gate, and memory unit, which can be defined as follows:

$$\begin{cases} \overrightarrow{h_t} = f\left(w_1 \cdot s_t + w_2 \cdot \overrightarrow{h_{t-1}}\right), \\ \overleftarrow{h_t} = f\left(w_3 \cdot s_t + w_5 \cdot \overleftarrow{h_{t+1}}\right), \\ y_t = g\left(w_4 \cdot \overrightarrow{h_t} + w_6 \cdot \overleftarrow{h_t}\right). \end{cases} \quad (10)$$

In (10), $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, respectively, represent the state of the forward LSTM layer and the backward LSTM layer at time t, corresponding to the context feature of IoT network traffic. s_t is the word vector input at time t, and w_1 to w_6 represent the layer's weight parameters. Also, f and g are activation functions, including sigmoid and tanh, y_t is the final output result of the BiLSTM model. By this, we can effectively extract long-distance dependent features and solve the problem of local feature loss.

3.6.5. Attention Mechanism Layer. The attention mechanism aims to allocate limited attention resources to crucial information, thereby enhancing the relevance of feature vectors and output results, which is widely used in natural language processing. In this paper, the attention mechanism is introduced to strengthen the attention features of the neural network. These features are extracted from the previous knowledge graph and statistical analysis. Thus, the attention layer can increase the contribution of key features to the corpus and add weights to better distinguish between normal requests and abnormal requests. This paper mainly uses the basic form of attention mechanism to pay attention to the weight distribution of IoT network traffic. The attention mechanism will focus on the semantic features and key features of different attack types. For example, in DoS attacks, the feature same_srv_rate and dst_host_same_srv_rate often appear. Our model will highlight their importance and ignore lesser features. The calculation process of this step is shown in equations (11) to (13).

$$v_t = \tanh\left(w_c \cdot y_t + b_c\right), \tag{11}$$

$$a_t = \mathrm{soft}\max\left(w^T, v_t\right), \tag{12}$$

$$u = \sum_{t=1}^{n} a_t \cdot y_t. \tag{13}$$

Here, (11) generates the target attention weight, namely, v_t, which is a result of the nonlinear transformation of the activation function tanh, y_t is a vector output of the CNN-BiLSTM network, w_c is a parameter of training weight, and b_c is bias item. (12) uses the softmax function to calculate the importance of each component v_t, the probability vector of the weight is a_t, and w^T is the transposed matrix. Finally, the generated attention weight is matched to the corresponding output vector y_t in our model, and u represents the sentence vector of the weighted sum of the importance of y_t, as shown in (13).

3.6.6. Fully Connected Layer. The fully connected layer plays the role of classifier in the neural network model. It maps the learned distributed features representation to the sample label space. Finally, this paper designs a softmax classifier for obtaining the classification results of intrusion detection and complete IoT networks' intrusion detection. In a word, we have successfully constructed the attention-based CNN-BiLSTM model to distinguish different classes through the above six steps shown in Figure 3.

4. Evaluation

In this section, we evaluate the performance of the proposed system in intrusion detection and attack classification. Also, we analyze the results of feature extraction methods based on the knowledge graph and statistical analysis. Finally, we compare the proposed system with state-of-the-art systems to verify the effectiveness and robustness of intrusion detection.

4.1. Dataset. In this paper, we evaluate the proposed approach on the NSL-KDD dataset [42], which is a new revised version of the KDD Cup 99 that has been generated by IoT devices simulating real-time attacks, bench-marked datasets for IDS. We can get the NSL-KDD dataset through the link [45]. The NSL-KDD dataset is the traffic request generated by MIT Lincoln Laboratory using IoT devices to simulate attacks in the real-time environment. Thus, it is a relatively authoritative dataset. Table 4 gives details of the NSL-KDD dataset. There are 125973 training traffic samples and 22544 testing traffic samples in the dataset, involving four categories of attacks. DoS is a denial of service attack, Probe is a port listening or scanning attack, Remote to Local (R2L) is a remote to local attack, and User-to-Root (U2R) is an unauthorized and trying to gain superuser or root. In the NSL-KDD dataset, there are 39 subcategories of attack scattered in four categories (i.e., DoS, Probe, R2L, and U2R), among which 22 subcategories appear in the training set, and 17 more different attack subcategories (i.e., unknown attacks) exist in the testing set. By this, we can evaluate the performance of our model for unknown attack types.

Next, Table 5 is a detailed description of 41 features, which includes four categories: intrinsic features of TCP connections, content features of TCP connections, time-based network traffic statistics features, and host-based network traffic statistics features. In addition, the attribute features include three nonnumerical features (i.e., protocol_type, service, and flag) and 38 numeric features. In the previous data preprocessing section, we systematically explain how to clean and preprocess our dataset. This section mainly focuses on the experimental evaluation of the proposed model.

4.2. Evaluation Metrics. To evaluate the performance of the intrusion detection model for IoT networks, this paper calculates the precision, recall, F_1-score, accuracy, and false alarm rate (FAR). Table 6 shows the detailed confusion matrix. By comparing the actual network request label and the predicted network request label (including normal requests and attack requests), the results of the classification algorithm are evaluated.

As shown in Table 6, True Positive (TP) means that both the predicted results of the network request and the actual label are positive (i.e., attack label). True Negative (TN) implies that the predicted results of the network request and the actual label are negative (i.e., normal label). False Positive (FP) indicates that the predicted result is negative, but the actual label is positive. False Negative (FN) indicates that the predicted result is positive, but the actual label is negative.

The five metrics are calculated by the following equation:

$$\begin{aligned}
\text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}},\\
\text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}},\\
\text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}},\\
F_1 - \text{score} &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},\\
\text{FAR} &= \frac{\text{FP}}{\text{FP} + \text{TN}}.
\end{aligned} \tag{14}$$

4.3. Experimental Setup. We implement the intrusion detection model of IoT networks using TensorFlow and Keras, with the programming language Python (version 3.7). Also, all experiments are run in Windows 10 64-bit operating

Table 4: Distribution of the NSL-KDD dataset.

Category	Normal	Abnormal				Total
		DoS	Probe	R2L	U2R	
Train	67343	45927	11656	995	52	125973
Test	9711	7458	2421	2754	200	22544

Table 5: Feature list of the NSL-KDD dataset.

No.	Feature Category	Feature Name
1	Intrinsic feature	duration
2		protocol_type
3		Service
4		Flag
5		src_bytes
6		dst_bytes
7		land
8		wrong_fragment
9		Urgent
10	Content feature	hot
11		num_failed_logins
12		logged_in
13		num_compromised
14		root_shell
15		su_attempted
16		num_root
17		num_file_creations
18		num_shells
19		num_access_files
20		num_outbound_cmds
21		is_hot_login
22		is_guest_login
23	Time-based feature	count
24		srv_count
25		serror_rate
26		srv_serror_rate
27		rerror_rate
28		srv_rerror_rate
29		same_srv_rate
30		diff_srv_rate
31		srv_diff_host_rate
32	Host-based feature	dst_host_count
33		dst_host_srv_count
34		dst_host_same_srv_rate
35		dst_host_diff_srv_rate
36		dst_host_same_src_port_rate
37		dst_host_srv_diff_host_rate
38		dst_host_serror_rate
39		dst_host_srv_serror_rate
40		dst_host_rerror_rate
41		dst_host_srv_rerror_rate

system environment with Inter Core i7-8700K CPU, 64 GB memory, and GTX 1080Ti GPU.

To detect malicious requests in IoT networks, we construct an attention-based CNN-BiLSTM model in this paper. In terms of model parameter setting, the number of convolution kernels of the CNN model is 128. The activation function rectified linear unit (ReLU) is adopted for the convolutional and fully connected layers. At the same time, the number of neurons in both the forward and backward directions of the BiLSTM model is 128, and the optimization algorithm selects the Adam optimizer. Then, the initial learning rate is set to 0.001, which is a standard starting point for traditional deep learning. During the experiment, the loss functions are the binary cross-entropy for binary classification and the categorical cross-entropy for multiclass classification. The dropout mechanism is introduced to randomly sample data for training to prevent overfitting, and its keep_prob parameter is set to 0.4. For completeness, we leverage the sci-kit-learn library to construct the other machine learning classification models as baselines (e.g., logistic regression, support vector machine, random forest, etc.). Further, all experiments use the same environment and the NSL-KDD dataset for a fair comparison. We can not only complete the detection of malicious requests (for binary classification) but also identify the types of attacks (for multiclass classification).

Table 6: Confusion matrix.

Predicted label	Actual label	
	Normal	Attack
Normal	True Negative (TN)	False Negative (FN)
Attack	False Positive (FP)	True Positive (TP)

Moreover, we implement two feature extraction algorithms based on knowledge graph and statistical analysis in Python, as shown in the previous section. Then, these semantic feature pairs and key features will be visually displayed by the Gephi tool and some Python libraries (e.g., matplotlib, seaborn, and pyecharts). In particular, these key features will add to the weight parameters of the IDS neural network model proposed in this paper. Finally, to better verify the effectiveness and robustness of the model, the final experimental result is the average of 10 network traffic classification results. By this, we can reduce the noise influence of a possible abnormal result through multiple experiments.

4.4. Analysis of Proposed Feature Extraction. We implement a feature extraction algorithm based on knowledge graphs in this section and conduct detailed experimental evaluations. We use the previous Algorithm 1 to perform entity extraction and relationship extraction on the training set of the NSL-KDD dataset and generate the corresponding feature knowledge graphs of four attack types as shown in Figure 4. The knowledge graph is scattered from the center to the surroundings, and the feature relationship pairs are represented by two tuples. Note that the thicker the lines between the features that are more closely related, and the more features that appear, the greater their size, and the features are clustered in different colors. For example, Figure 4(a) is the knowledge graph of the DoS attack type. It can be found that the feature two tuples with close semantic relationship are <same_srv_rate, dst_host_same_srv_rate>, <dst_host_diff_srv_rate, same_srv_rate>, <tcp, same_srv_rate>, <dst_host_count255, same_srv_rate>, etc.

In addition, by comparing each subgraph, we can find that the semantic relationship of different attack types is

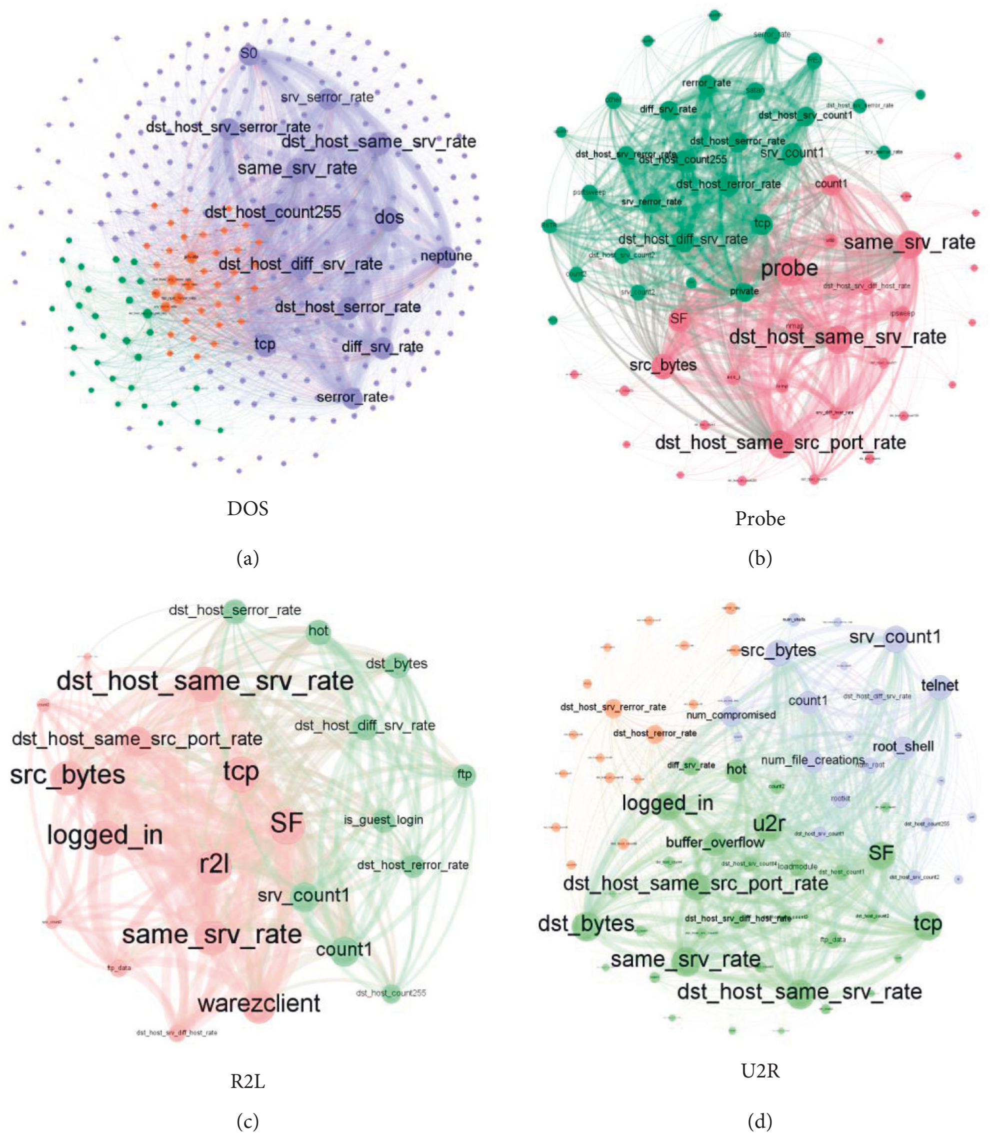


Figure 4: The semantic feature relationships of four attack types are extracted by a knowledge graph. (a) DoS. (b) Probe. (c) R2L. (d) U2R.

different. Among them, the closely related feature pairs in the DoS attack are concentrated in the host-based feature; the closely related feature pairs in the Probe attack are concentrated in the two major categories of time-based features and host-based features; the closely related feature pairs in the R2L attack cover four categories, namely, intrinsic features (e.g., SF), content features (e.g., logged_in), time-based features (e.g., same_srv_rate), and host-based features (e.g., dst_host_same_srv_rate); U2R attacks also cover four categories of features.

At the same time, we construct a knowledge graph for the training set of normal network requests, as shown in Figure 5 in the appendix. Table 7 shows the critical feature pairs of normal and abnormal requests (covering five labels). Each classification label displays the top 8 unique results represented by triples <src_feature, dst_feature, weight>, corresponding to the source position of the feature, the destination position of the feature, and semantic relation weight. Each weight adopts Min-Max normalization processing.

This section implements the feature extraction algorithm based on statistical analysis. We select features on the different classification labels of the four feature categories (i.e., normal, DoS, Probe, R2L, and U2L). Each category extracts two essential features. The results are shown in Table 8 in the appendix, represented by the two-tuple < feature, diff>, corresponding to the key features and the average difference of this feature's category. The calculation process is shown in the previous equation (4).

To further evaluate the effect of the feature extraction algorithm based on statistical analysis, we use the power-law distribution function to analyze the NSL-KDD dataset and extract six representative features (i.e., duration, num_compromised, num_root, src_bytes, dst_bytes, and srv_count). The experimental results are shown in Figure 6. The experimental results show that the power-law

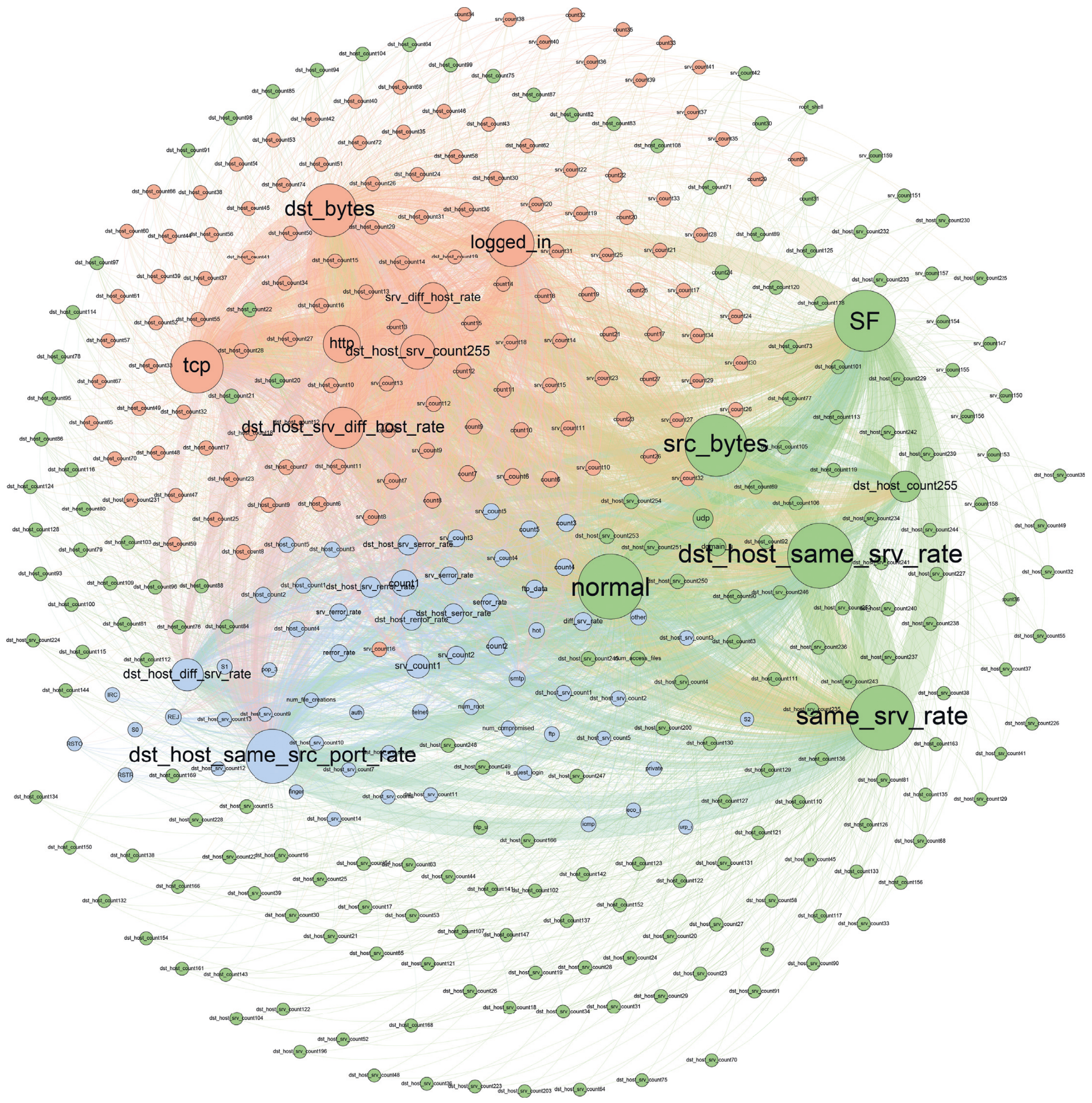


FIGURE 5: The semantic feature relationships of normal requests are extracted by a knowledge graph.

distribution of features is generally consistent with the features extracted in this section, and some features are affected by unbalanced data distribution. However, the traffic features of the whole NSL-KDD dataset basically conform to the law of human network attack activities, and the attacks are mainly caused by human factors. So far, we effectively verified that the feature extraction algorithm proposed in this paper is very important to the identification of abnormal requests and the detection of attack types.

Finally, we implement feature fusion and feature alignment algorithms according to Algorithm 2. The fusion features are shown in Table 9. Among them, $f^{(KG)}$ represents the experimental results of the knowledge graph-based feature extraction algorithm, $f^{(SA)}$ represents the experimental results of the statistical analysis-based feature extraction algorithm, and $f^{(KG,SA)}$ represents the features recognized by two algorithms. Also, f_i represents the i-th feature, with a total of 41 features. Through Algorithm 2, we perform feature fusion and alignment for normal, DoS, Probe, R2L, and U2L, and the number of key features extracted for each type is 12, 14, 15, 13, and 14, respectively. Then, these features will be added to the initial weight of the subsequent IDS system based on deep learning. Therefore, since the semantic relationship between features and the statistical distribution of features are considered in our system, IDS combining the two views can better detect the IoT malicious traffic and identify different types of network attack requests.

Table 7: Key semantic features extracted by the knowledge graph.

Label	Feature category
Normal	①<same_srv_rate, dst_host_same_srv_rate,0.980> ②<src_bytes, same_srv_rate,0.950> ③<SF,src_bytes,0.940> ④<src_bytes, dst_bytes,0.830> ⑤<tcp,same_srv_rate,0.800> ⑥<logged_in,same_srv_rate,0.710> ⑦<same_srv_rate, dst_host_same_src_port_rate,0.670> ⑧<http, same_srv_rate,0.570>
DoS	①<same_srv_rate, dst_host_same_srv_rate,0.970> ②<dst_host_diff_srv_rate, same_srv_rate,0.950> ③<tcp,same_srv_rate,0.920> ④<dst_host_count255,same_srv_rate,0.920> ⑤<tcp,diff_srv_rate,0.910> ⑥<dst_host_serror_rate, dst_host_diff_srv_rate,0.780> ⑦<tcp,dst_host_srv_serror_rate,0.770> ⑧<serror_rate, dst_host_serror_rate,0.770>
Probe	①<same_srv_rate, dst_host_same_src_port_rate,0.950> ②<same_srv_rate, dst_host_same_srv_rate,0.810> ③<src_bytes, same_srv_rate,0.700> ④<dst_host_diff_srv_rate, dst_host_rerror_rate,0.640> ⑤<count, dst_host_same_src_port_rate,0.610> ⑥<SF,src_bytes,0.600> ⑦<dst_host_count255,dst_host_diff_srv_rate,0.580> ⑧<rerror_rate, dst_host_rerror_rate,0.550>
R2L	①<tcp,same_srv_rate,1.000> ②<same_srv_rate, dst_host_same_srv_rate,0.990> ③<src_bytes,tcp,0.970> ④<SF,same_srv_rate,0.940> ⑤<logged_in,SF,0.910> ⑥<dst_host_same_src_port_rate, dst_host_same_srv_rate,0.740> ⑦<count, same_srv_rate,0.710> ⑧<srv_count, count,0.700>
U2L	①<SF,same_srv_rate,0.980> ②<dst_bytes, same_srv_rate,0.961> ③<tcp,dst_bytes,0.941> ④<dst_host_same_srv_rate, same_srv_rate,0.922> ⑤<logged_in,dst_bytes,0.882> ⑥<srv_count, same_srv_rate,0.824> ⑦<dst_host_same_src_port_rate, dst_host_same_srv_rate,0.824> ⑧<src_bytes, dst_bytes,0.725>

4.5. Comparison of Different Intrusion Detection System Models. This paper proposes an enhanced intrusion detection system model based on deep learning and a knowledge graph. To verify the effectiveness of our system, the experimental evaluation is compared with the traditional machine learning models and the existing deep learning models. Detailed comparative experimental results are shown in Table 10. The precision, recall, and F_1-score of the proposed system are 0.9035, 0.9107, and 0.9071, respectively, superior to state-of-the-art systems. Moreover, the F_1-score of our system is 10.29% higher than that of the best machine learning algorithm (random forest) and 5.46% higher than that of the compared deep learning algorithm (CNN-BiLSTM-attention). Therefore, our system improves the semantic relationship between traffic features through knowledge graphs, improves the importance of crucial features through statistical analysis, and adds weight to the deep learning model to improve the accuracy of IDS for IoT networks. Meanwhile, the CNN-BiLSTM-attention model constructed in this paper can effectively capture long-distance dependent information, and the attention mechanism can highlight some features. These factors can highlight the contribution of our model to intrusion detection.

In addition, the detection time of IoT network traffic is also an important indicator for evaluating IDS, which can effectively measure the time cost and algorithm complexity of a model. Column 6 of Table 10 gives the detection times for different systems in seconds. The detection time of the system in this paper for the network traffic of the test set is 22.17 seconds, and it can effectively detect 1016 network requests per second. The entire detection time efficiency is in the upper-middle range. Compared with the improvement of the F1-score, the time cost is within an acceptable range, only slightly higher than other models, and does not show an exponential increase.

Table 8: Key features extracted by the statistical analysis.

Label	Intrinsic feature	Content feature
Normal	<dst_bytes,0.1187> <src_bytes,0.0971>	<logged_in,0.3149> <is_guest_login,0.0036>
DoS	<dst_bytes,0.1379> <duration,0.0284>	<logged_in,0.3749> <is_guest_login,0.0094>
Probe	<dst_bytes,0.1435> <src_bytes,0.1082>	<logged_in,0.3886> <is_guest_login,0.0093>
R2L	<src_bytes,0.1625> <duration,0.0736>	<logged_in,0.5179> <is_guest_login,0.3052>
U2L	<duration,0.2563> <dst_bytes,0.2138>	<logged_in,0.4889> <num_shells,0.0671>
Label	Time-based Feature	Host-based Feature
Normal	<same_srv_rate,0.3085> <serror_rate,0.2711>	<dst_host_srv_count,0.2927> <dst_host_same_srv_rate,0.2907>
DoS	<serror_rate,0.4690> <srv_serror_rate,0.4642>	<dst_host_srv_serror_rate,0.4659> <dst_host_serror_rate,0.4634>
Probe	<srv_rerror_rate,0.3233> <rerror_rate,0.3173>	<dst_host_same_src_port_rate,0.5034> <dst_host_srv_rerror_rate,0.3208>
R2L	<same_srv_rate,0.3358> <serror_rate,0.2726>	<dst_host_same_src_port_rate,0.4485> <dst_host_count,0.3651>
U2L	<srv_serror_rate,0.2825> <same_srv_rate,0.2706>	<dst_host_count,0.5270> <dst_host_srv_count,0.4147>

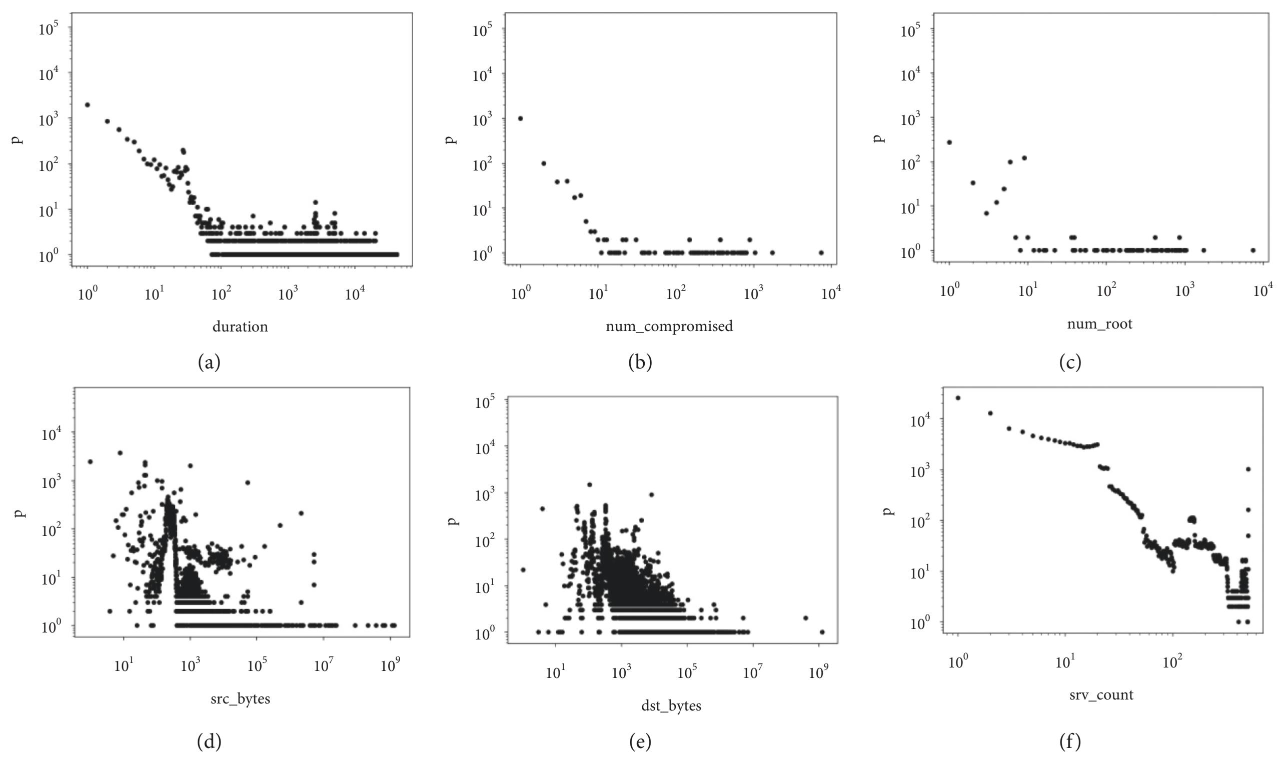


Figure 6: Experimental results of power-law distribution with different features. (a) duration. (b) num_compromised. (c) num_root. (d) src_bytes. (e) dst_byte.s (f) srv_count.

Note that the comparison IDS in this paper is due to differences in datasets, and some of the code is not open source. Therefore, the comparison systems combine the methods and ideas of the references and the typical intrusion detection methods to reconstruct the code, such as SVM [13], KNN [14], DT [15], RF [16], LSTM [32], and BiLSTM [33] in Table 10. Finally, we reproduce these baseline methods and conduct a detailed comparative analysis of our dataset.

In order to further measure the detection effect of the proposed IDS model and evaluate its identification results of malicious network requests and normal network requests,

Table 9: Feature fusion results for the five types of requests on the NSL-KDD dataset.

Label	Number	Feature fusion and feature alignment
Normal	12	$f_2^{(KG)}, f_3^{(KG)}, f_4^{(KG)}, f_5^{(KG,SA)}, f_6^{(KG,SA)}, f_{12}^{(KG,SA)}, f_{22}^{(SA)}, f_{25}^{(SA)}, f_{29}^{(KG,SA)}, f_{33}^{(SA)}, f_{34}^{(KG,SA)}, f_{36}^{(KG)}$
DoS	14	$f_1^{(SA)}, f_2^{(KG)}, f_6^{(SA)}, f_{12}^{(SA)}, f_{22}^{(SA)}, f_{25}^{(KG)}, f_{26}^{(SA)}, f_{29}^{(KG,SA)}, f_{30}^{(KG)}, f_{32}^{(KG)}, f_{34}^{(KG)}, f_{35}^{(KG)}, f_{38}^{(KG,SA)}, f_{39}^{(KG,SA)}$
Probe	15	$f_4^{(KG)}, f_5^{(KG,SA)}, f_6^{(SA)}, f_{12}^{(SA)}, f_{22}^{(SA)}, f_{23}^{(KG)}, f_{27}^{(KG,SA)}, f_{28}^{(SA)}, f_{29}^{(KG)}, f_{32}^{(KG)}, f_{34}^{(KG)}, f_{35}^{(KG)}, f_{36}^{(KG,SA)}, f_{40}^{(KG)}, f_{41}^{(SA)}$
R2L	13	$f_1^{(SA)}, f_2^{(KG)}, f_4^{(KG)}, f_5^{(KG,SA)}, f_{12}^{(KG,SA)}, f_{22}^{(SA)}, f_{23}^{(KG)}, f_{24}^{(KG)}, f_{25}^{(SA)}, f_{29}^{(KG,SA)}, f_{32}^{(SA)}, f_{34}^{(KG)}, f_{36}^{(KG,SA)}$
U2L	14	$f_1^{(SA)}, f_2^{(KG)}, f_4^{(KG)}, f_5^{(KG)}, f_6^{(KG,SA)}, f_{12}^{(KG,SA)}, f_{18}^{(SA)}, f_{24}^{(KG)}, f_{26}^{(SA)}, f_{29}^{(KG,SA)}, f_{32}^{(SA)}, f_{33}^{(SA)}, f_{34}^{(KG)}, f_{36}^{(KG)}$

Table 10: Performance comparison of various IoT intrusion detection models.

Type	Model	Precision	Recall	F_1-score	Detection Time
ML-based IDS	KNN	0.7997	0.7940	0.7968	23.40
	LR	0.7739	0.7578	0.7658	5.97
	DT	0.7726	0.7613	0.7669	5.44
	SVM	0.7741	0.7600	0.7670	13.26
	RF	0.8131	0.7956	0.8043	10.99
	AdaBoost	0.8065	0.7874	0.7969	18.40
DL-based IDS	CNN	0.8294	0.8228	0.8261	7.82
	TextCNN	0.8417	0.8391	0.8404	19.40
	LSTM	0.8265	0.8179	0.8222	13.93
	BiLSTM	0.8500	0.8504	0.8502	14.15
	BiGRU	0.8271	0.8183	0.8227	11.46
	CNN-BiLSTM-Att	0.8520	0.8530	0.8525	19.34
	Our System	0.9035	0.9107	0.9071	22.17

we compared the accuracy and FAR of different IDS models. The results are shown in Figure 7, and the accuracy of our system is 0.9001, and the FAR is 0.0120, both of which are better than the experimental results of other models. The accuracy is 14.71% higher than the average of the six machine learning models and 8.50% higher than the average of the other six deep learning models. On the other hand, the FAR is 4.61% lower than the average of the six machine learning models and 2.45% lower than the average of the other six deep learning models. The experiment result indicates that our system can identify more malicious requests and discover semantic feature relationships between normal and abnormal requests, that is, the feature relationship pairs that often appear at the same time, including < same_srv_rate, dst_host_same_srv_rate>, <dst_host_diff_srv_rate, same_srv_rate>, <same_srv_rate, and dst_host_same_src_port_rate>.

This paper implements various malicious request attack identification experiments on the proposed model. The experimental results of the five labels are shown in Table 11. Among them, the recognition performance of normal request, DoS attack, and Probe attack is better, and their F_1-score is 0.9107, 0.9273, and 0.8849, respectively, indicating that our system can effectively extract semantic relations and key features. Although the experimental results of the U2R type are poor due to small samples, our system yet detects 12 attack requests, reflecting that our system has better robustness and accuracy in identifying unknown attacks.

To better evaluate the performance of the attention-based CNN-BiLSTM intrusion detection system and compare the ability of IDSs to identify malicious requests with different thresholds, this paper selects the random forest algorithm (a better machine learning model) and the CNN-BiLSTM-Att algorithm (a better deep learning model) to compare the ROC curve with the IDS system. Figure 8 shows the final renderings. It can be seen that the AUC area corresponding to the ROC curve of our system is the largest, which proves that our IDS has the best comprehensive performance and can obtain higher TPR and lower FPR.

Moreover, this paper compares different intrusion detection systems to identify malicious attack types (i.e., multiclassification tasks), and the experimental results are shown in Figure 9. The abscissa is various systems, and the ordinate is the F_1-score corresponding to the detection results of different systems. Note that we only select the random forest model, the best machine learning algorithm in our experiment. In Figure 9, whether the system in this paper detects normal network requests or recognizes malicious request types, its F_1-score is better than other systems, and it can effectively identify unknown attack types. Among them, the F_1-score of the normal type is 13.37% higher than the average value of other comparison systems; the DoS attack type is 7.25% higher than the average value of the other comparison systems; the Probe attack type is 15.68% higher than the average value of the other comparison systems; the R2L attack type is higher than the average value. The average value of other comparison systems is 43.12% higher; the U2R attack type is 10.67% higher than the average value of other comparison systems. In short, this experiment further proves the contribution of the IDS in this paper. The two feature extraction algorithms proposed in this paper (i.e., knowledge graph-based feature extraction algorithm and statistical analysis-based feature extraction algorithm) can effectively mine the semantic relationship between features and improve the performance of the IDS for IoT networks.

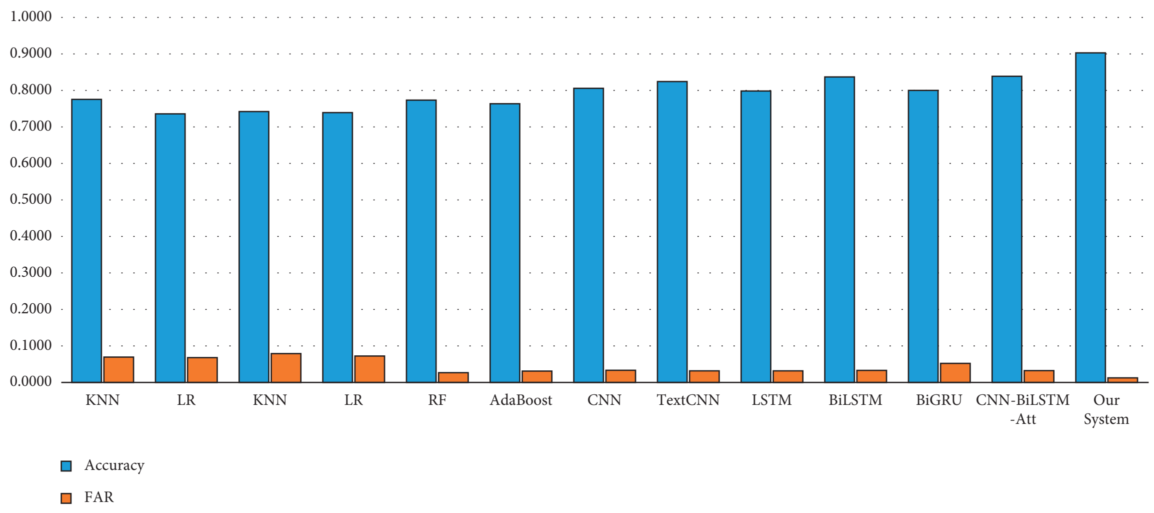


Figure 7: Comparison of accuracy and FAR of different IoT intrusion detection models.

Table 11: The results of five labels by our intrusion detection system.

Label	Precision	Recall	F_1-score
Normal	0.8541	0.9754	0.9107
DoS	0.9021	0.9540	0.9273
Probe	0.8742	0.8959	0.8849
R2L	0.9060	0.3500	0.5050
U2R	0.5455	0.0600	0.1081

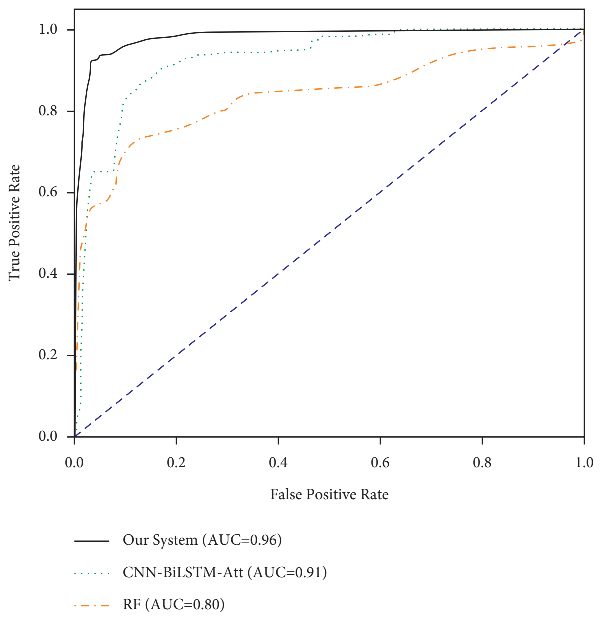


Figure 8: Comparison of ROC curves of different systems.

Finally, to prove that our system has a good detection effect on unknown attacks, this paper compares the results of different models against unknown types of attacks. In the testing set of the NSL-KDD dataset, 17 subcategories belong to unknown attacks. Among them, DoS includes 4 types of unknown attacks (i.e., apache2, mailbomb, processtable, and udpstorm) and 1717 traffic samples; Probe includes 2 types of unknown attacks (i.e., mscan and saint), a total of 1315 traffic samples; R2L includes 7 types of unknown attacks (i.e., named, sendmail, snmpgetattack, snmpguess, worm,

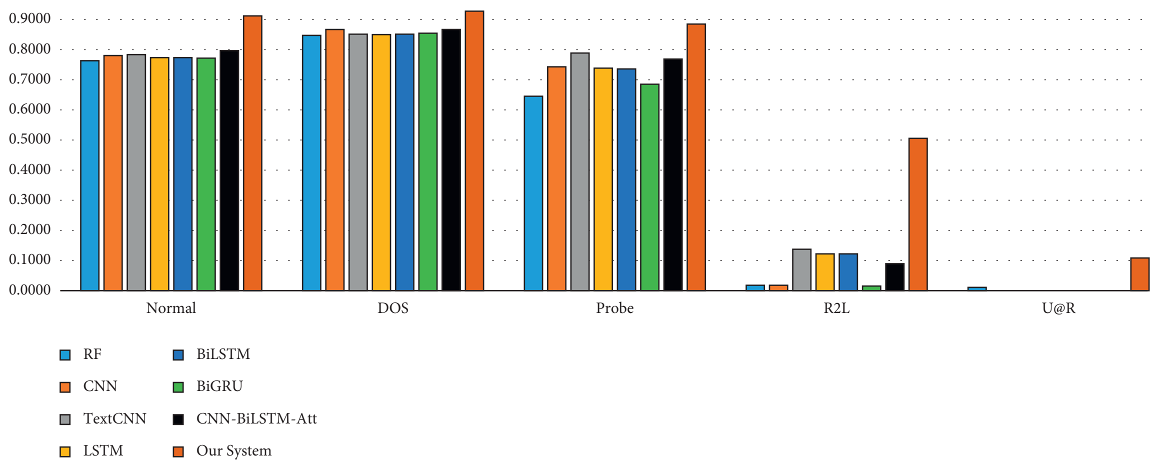
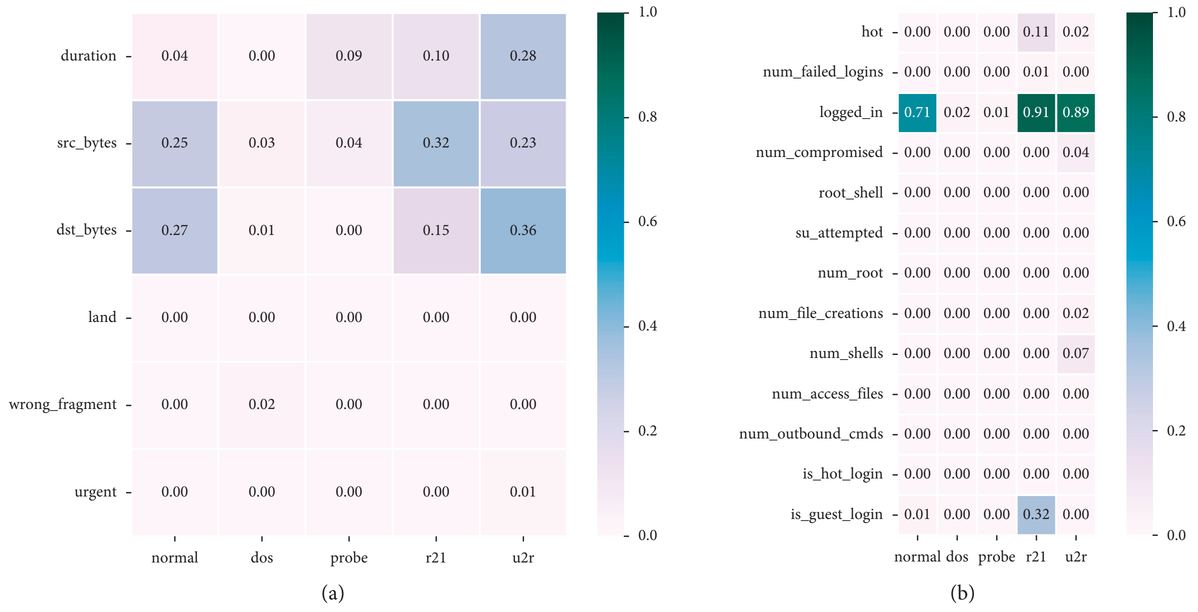


Figure 9: Experimental results of different systems to identify malicious attack types.

Table 12: Performance comparison of different systems for unknown types.

Type	Model	F_1-score	Ranking of various labels of detection performance
ML-based IDS	KNN	0.6201	Probe > DoS > U2R > R2L
	LR	0.5701	Probe > DoS > U2R > R2L
	DT	0.5804	DoS > Probe > U2R > R2L
	SVM	0.5865	Probe > DoS > U2R > R2L
	RF	0.6329	Probe > DoS > U2R > R2L
	AdaBoost	0.5676	Probe > DoS > U2R > R2L
DL-based IDS	CNN	0.7096	Probe > DoS > U2R > R2L
	TextCNN	0.7566	Probe > DoS > U2R > R2L
	LSTM	0.7169	Probe > DoS > U2R > R2L
	BiLSTM	0.8033	Probe > DoS > U2R > R2L
	BiGRU	0.7579	Probe > DoS > U2R > R2L
	CNN-BiLSTM-Att	0.8146	Probe > DoS > U2R > R2L
	Our System	0.8783	Probe > DoS > U2R > R2L



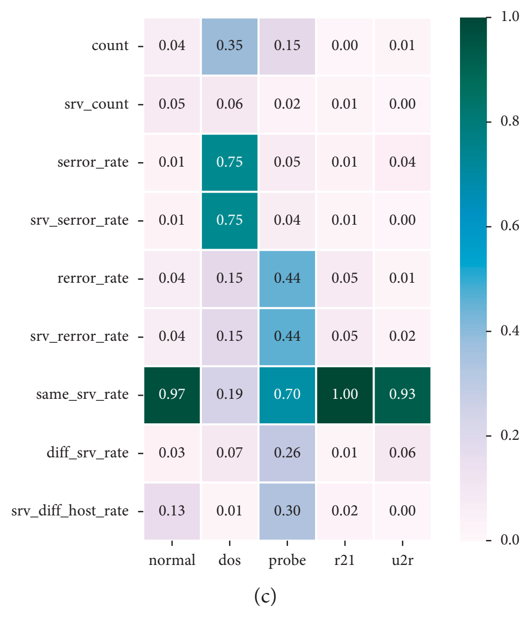
(a)

	normal	dos	probe	r21	u2r
duration	0.04	0.00	0.09	0.10	0.28
src_bytes	0.25	0.03	0.04	0.32	0.23
dst_bytes	0.27	0.01	0.00	0.15	0.36
land	0.00	0.00	0.00	0.00	0.00
wrong_fragment	0.00	0.02	0.00	0.00	0.00
urgent	0.00	0.00	0.00	0.00	0.01

(b)

	normal	dos	probe	r21	u2r
hot	0.00	0.00	0.00	0.11	0.02
num_failed_logins	0.00	0.00	0.00	0.01	0.00
logged_in	0.71	0.02	0.01	0.91	0.89
num_compromised	0.00	0.00	0.00	0.00	0.04
root_shell	0.00	0.00	0.00	0.00	0.00
su_attempted	0.00	0.00	0.00	0.00	0.00
num_root	0.00	0.00	0.00	0.00	0.00
num_file_creations	0.00	0.00	0.00	0.00	0.02
num_shells	0.00	0.00	0.00	0.00	0.07
num_access_files	0.00	0.00	0.00	0.00	0.00
num_outbound_cmds	0.00	0.00	0.00	0.00	0.00
is_hot_login	0.00	0.00	0.00	0.00	0.00
is_guest_login	0.01	0.00	0.00	0.32	0.00

Figure 10: Continued.



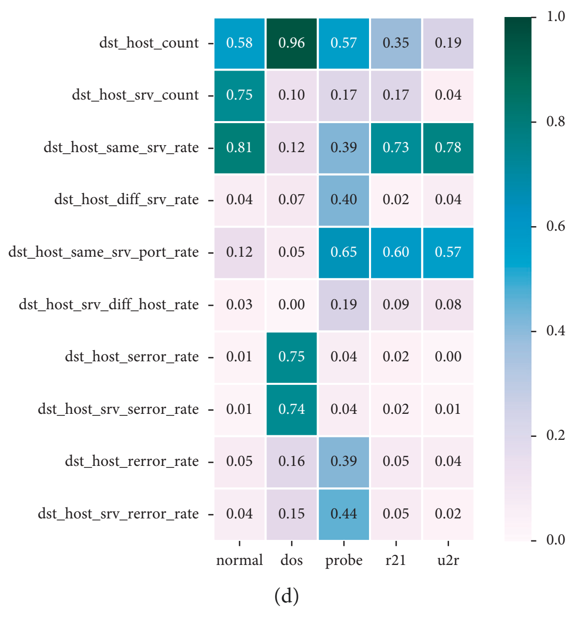


Figure 10: Heat map distribution of different traffic features. (a) Intrinsic feature. (b) Content feature. (c) Time-based feature. (d) Host-based feature.

xlock, and xsnoop) and 555 traffic samples; U2R includes 4 types of unknown attacks (i.e., httptunnel, ps, sqlattack, and xterm) and 163 traffic samples. The results of the whole experiment are shown in Table 12.

Among them, the F_1-score of the proposed IDS in this paper is 0.8783, which is higher than other IDSs based on machine learning and deep learning. This F_1-score of our IDS is 25.82%, 30.82%, 29.79%, 29.18%, 24.54%, and 31.07% higher than KNN, LR, DT, SVM, RF, and AdaBoost. Also, this value is higher than CNN, TextCNN, LSTM, BiLSTM, BiGRU, and CNN-BiLSTM-Att out 16.87%, 12.17%, 16.14%, 7.50%, 12.04%, and 6.37%. This experiment shows that the system in this paper can better detect unknown attacks, and its improvement is better than the previous Table 10. In addition, this paper compares the F_1-score of different categories and puts the obtained ranking results in the fourth column of Table 12. The experimental results show that, except for the DT algorithm, other systems' detection and ranking results are Probe, DoS, U2R, and R2L. In short, our IDS has a good performance in both semantic feature extraction and malicious detection of unknown attacks.

5. Conclusions

This paper proposed and implemented an enhanced intrusion detection system based on a knowledge graph and CNN-BiLSTM-attention. Our IDS combines knowledge graph-based feature extraction and statistical analysis-based feature extraction, which can effectively extract the contextual semantic relationship and crucial features of IoT network traffic. The model has better accuracy and robustness. In particular, the proposed system extracts the key features of normal and abnormal requests (including DoS, Probe, R2L, and U2R attack types) in detail, ensuring robust detection and identifying the attack types (including unknown attacks) of network requests in real-time. We demonstrated that the feature extraction algorithm based on knowledge graph and multiviews fusion could accurately extract key traffic features and have certain interpretability. Extensive experiments showed that our IDS could effectively detect various attacks on IoT networks. It achieved a precision of 90.35%, a recall of 91.07%, and an F_1-score of 90.71%, which outperformed state-of-the-art systems. Moreover, the F_1-score of our system is 10.29% higher than that of the best machine learning algorithm (random forest) and 5.46% higher than that of the compared deep learning algorithm (CNN-BiLSTM-attention). The accuracy of our system is 0.9001, which is 14.71% higher than the average of the six machine learning models and 8.50% higher than the average of the other six deep learning models. In particular, our IDS can identify unknown attack types with small samples, the recognition performance of DoS attack and Probe attack is better than other systems, and their F_1-score is 0.9273 and 0.8849. In short, our system can detect various stealthy attack types (including DoS, Probe, R2L, and U2L) and extract semantic relationships among features.

In the future, on the one hand, we will construct a network intrusion detection system based on knowledge graphs and deep learning for larger-scale network traffic, which can realize real-time monitoring of malicious traffic on enterprise IoT networks. On the other hand, we will combine the advantages of graph neural network and provenance graph to optimize our neural network model in this paper, thereby building a more robust intrusion detection system to identify encrypted or obfuscated malicious network traffic.

Appendix

Figure 5 is a knowledge graph, which is constructed by the training set of normal network requests. Among them, the closely related feature relationship pairs mainly cover features such as src_bytes, same_srv_rate, dst_host_same_srv_rate, and SF.

The key features extracted based on statistical analysis are shown in Table 8. Each category extracts two essential features, and each feature is represented by a two-tuple < feature, diff>, corresponding to the key features and the average difference of this feature's category. The calculation process is shown in the previous equation (4). For example, the time-based feature of the type of DoS attack is error_rate, and the average difference is 0.4690.

In particular, the authors calculate the average value of different (numerical) features. The heat map distribution is shown in Figure 10. The authors can see the key features of different attack types. The statistical analysis results of the whole distribution are basically consistent with those in Table 8. For example, src_bytes, logged_in, same_srv_rate, and dst_host_same_srv_rate features play an important role in detecting different types of network requests.

Data Availability

The dataset used in this study is free and publicly available on the Internet. We can get the NSL-KDD dataset through the following link: https://www.unb.ca/cic/datasets/nsl.html.

Conflicts of Interest

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

Acknowledgments

This research was in part supported by the National Natural Science Foundation of China under Grant Nos. 62172308, U1626107, 61972297, and 62172144.

References

[1] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[2] A. J. Siddiqui and A. Boukerche, "TempoCode-IoT: temporal codebook-based encoding of flow features for intrusion detection in Internet of Things," *Cluster Computing*, vol. 24, no. 1, pp. 17–35, 2021.

[3] Wikipedia. 2021. Stuxnet, 2021, https://en.wikipedia.org/wiki/Stuxnet.

[4] Wikipedia, *Ukraine Power Grid Hack*, 2021, https://en.wikipedia.org/wiki/Ukraine_power_grid_hack.

[5] Wikipedia, *2020 United States Federal Government Data Breach*, 2021, https://en.wikipedia.org/wiki/2020_United_States_federal_government_data_breach.

[6] Wikipedia, *Colonial Pipeline Ransomware Attack*, 2021, https://en.wikipedia.org/wiki/Colonial_Pipeline_ransomware_attack.

[7] T. Shekari, C. Bayens, M. Cohen, L. Graber, and amd R. Beyah, "RFDIDS: radio frequency-based distributed intrusion detection system for the power grid," *Proceedings of the 26th Network and Distributed System Security Symposium*, San Diego CA, USA, 2019.

[8] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. Yao, "Deep learning-based anomaly detection in cyber-physical systems: progress and opportunities," *ACM Computing Surveys*, vol. 54, no. 5, pp. 1–36, 2021.

[9] A. Alsaheel, Y. Nan, S. Ma et al., "ATLAS: a sequence-based learning approach for attack investigation," *Proceedings of the 30th USENIX Security Symposium*, pp. 3005–3022, August 2021.

[10] Y. Tang, Y. Wang, H. Li, and X. Li, "To cloud or not to cloud: an on-line scheduler for dynamic privacy-protection of deep learning workload on edge devices," *CCF Transactions on High Performance Computing*, vol. 3, no. 1, pp. 85–100, 2021.

[11] Y. Duan, X. Li, J. Wang, and H. Yin, "DeepBinDiff: learning program-wide code representations for binary diffing," *Proceedings of the 27th Network and Distributed System Security Symposium*, San Diego CA, USA, 2020.

[12] W. Meng, W. Li, L. Jiang, K.-K. R. Choo, and C. Su, "Practical bayesian poisoning attacks on challenge-based collaborative intrusion detection networks," *Lecture Notes in Computer Science*, pp. 493–511, Luxembourg, 2019.

[13] P. Hadem, D. K. Saikia, and S. Moulik, "An SDN-based intrusion detection system using SVM with selective logging for IP traceback," *Computer Networks*, vol. 191, Article ID 108015, 2021.

[14] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: an intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Systems*, vol. 78, pp. 13–21, 2015.

[15] A. J. Malik and F. A. Khan, "A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection," *Cluster Computing*, vol. 21, no. 1, pp. 667–680, 2018.

[16] S. Masarat, S. Sharifian, and H. Taheri, "Modified parallel random forest for intrusion detection systems," *The Journal of Supercomputing*, vol. 72, no. 6, pp. 2235–2258, 2016.

[17] M. Mazini, B. Shirazi, and I. Mahdavi, "Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms," *Journal of King Saud University - Computer and Information Sciences*, vol. 31, no. 4, pp. 541–553, 2019.

[18] C. Liu, Z. Gu, and J. Wang, "A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning," *IEEE Access*, vol. 9, pp. 75729–75740, 2021.

[19] S. Ahn, H. Yi, Y. Lee, W. R. Ha, G. Kim, and Y. Paek, "Hawkware: network intrusion detection based on behavior analysis with ANNs on an IoT device," *Proceedings of the 57th Design Automation Conference*, pp. 1–6, San Francisco, CA, USA, 2020.

[20] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.

[21] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *Proceedings of the 27th ACM SIGSAC Conference on Computer and Communications Security*, pp. 3431–3446, Virtual Event, Korea, 2021.

[22] J. B. D. Caberera, B. Ravichandran, and R. K. Mehra, "Statistical traffic modeling for network intrusion detection,"

Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 466–473, San Francisco, California, USA, 2000.

[23] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.

[24] T. F. Lunt and R. Jaganna, "A prototype real-time intrusion-detection expert system," *Proceedings of the 9th IEEE Symposium on Security and Privacy*, pp. 59–66, Oakland, California, USA, 1988.

[25] K. Borders, J. Springer, and M. Burnside, "Chimera: a declarative language for streaming network traffic analysis," in *Proceedings of the 21st USENIX Security Symposium*, pp. 365–379, Bellevue, WA, USA, 2012.

[26] H. Li, H. Hu, G. Gu, G. Ahn, and F. Zhang, "vNIDS: towards elastic security with safe and efficient virtualization of network intrusion detection systems," *Proceedings of the 25th ACM SIGSAC Conference on Computer and Communications Security*, pp. 17–34, Toronto, ON, Canada, 2018.

[27] H. Haugerud, H. N. Tran, N. Aitsaadi, and A. Yazidi, "A dynamic and scalable parallel Network Intrusion Detection System using intelligent rule ordering and Network Function Virtualization," *Future Generation Computer Systems*, vol. 124, pp. 254–267, 2021.

[28] W. Li, W. Meng, and M. H. Au, "Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in IoT environments," *Journal of Network and Computer Applications*, vol. 161, Article ID 102631, 2020.

[29] R. Bitton and A. Shabtai, "A machine learning-based intrusion detection system for securing remote desktop connections to electronic flight bag servers," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1164–1181, 2021.

[30] X. Li, M. Zhu, L. T. Yang et al., "Sustainable ensemble learning driving intrusion detection model," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 4, pp. 1591–1604, 2021.

[31] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[32] N. Gupta, V. Jindal, and P. Bedi, "LIO-IDS: handling class imbalance using LSTM and improved one-vs-one technique in intrusion detection system," *Computer Networks*, vol. 192, Article ID 108076, 2021.

[33] Y. Imrana, Y. Xiang, L. Ali, and Z. Abdul-Rauf, "A bidirectional LSTM deep learning approach for intrusion detection," *Expert Systems with Applications*, vol. 185, pp. 1–12, 2021.

[34] P. Sun, P. Liu, Q. Li et al., "DL-IDS: extracting features using CNN-LSTM hybrid network for intrusion detection system," *Security and Communication Networks*, vol. 2020, pp. 1–11, Article ID 8890306, 2020.

[35] D. Li, L. Deng, M. Lee, and H. Wang, "IoT data feature extraction and intrusion detection system for smart cities based on deep migration learning," *International Journal of Information Management*, vol. 49, pp. 533–545, 2019.

[36] N. Balakrishnan, A. Rajendran, D. Pelusi, and V. Ponnusamy, "Deep Belief Network enhanced intrusion detection system to prevent security breach in the Internet of Things," *Internet of Things*, vol. 14, Article ID 100112, 2021.

[37] M. Mahdavisharif, S. Jamali, and R. Fotohi, "Big data-aware intrusion detection system in communication networks: a deep learning approach," *Journal of Grid Computing*, vol. 19, no. 4, pp. 1–28, 2021.

[38] S. M. Kasongo and Y. X. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Computers & Security*, vol. 92, Article ID 101752, 2020.

[39] M. A. Ferrag, L. A. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, Article ID 102419, 2020.

[40] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *Proceedings of the 25th Network and Distributed System Security Symposium*, San Diego CA, USA, 2020.

[41] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "A hierarchical hybrid intrusion detection approach in IoT scenarios," *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–7, 2020.

[42] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 53–58, 2009.

[43] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, vol. 165, Article ID 106944, 2019.

[44] Z. Bu, B. Zhou, P. Cheng, K. Zhang, and Z.-H. Ling, "Encrypted network traffic classification using deep and parallel network-in-network models," *IEEE Access*, vol. 8, pp. 132950–132959, 2020.

[45] UNB NSL-KDD Datasets: 2020, https://www.unb.ca/cic/datasets/nsl.html.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 4196945, 10 pages
https://doi.org/10.1155/2022/4196945

Research Article

Automating Group Management of Large-Scale IoT Botnets for Antitracking

Pengyu Pan,[1,2] **Xiaobo Ma**,[1,2] **Yingjie Fu**,[1,2] **and Feitong Chen**[1,2]

[1]*MOE Key Lab for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, China*
[2]*Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China*

Correspondence should be addressed to Xiaobo Ma; xma.cs@xjtu.edu.cn

Received 23 December 2021; Revised 13 March 2022; Accepted 31 March 2022; Published 14 April 2022

Academic Editor: Weizhi Meng

With the popularity of Internet of Things (IoT) devices, IoT botnets like Mirai have been infecting as many devices as possible such as IP cameras and home routers. Because of the sheer volume and continual operation of many vulnerabilities (many users do not pay much attention to IoT update alerts and leave the configurations by default) of IoT devices, the population of an IoT botnet becomes increasingly tremendous. The growing population, though making a botnet powerful, results in an increased risk of exposure. Specifically, once a bot is captured, the command and control (C&C) channel may be cracked and then tracked, potentially rendering more bots being discovered. To solve this problem, this paper proposes an automated approach to group management of large-scale IoT bots. The basic idea of the proposed approach is to establish a reliable and unsuspicious social network-based C&C channel capable of automatically grouping bots, wherein a group of bots have a unique ID that is against cross-group tracking. The Diffie–Hellman key exchange method is leveraged for efficiently generating the unique group ID, thereby scaling up automatic bot grouping. We refer to the botnet proposed in this paper as a multichannel automatic grouping botnet (MCG botnet) and conduct verification experiments using social networks and more than 2,000 docker nodes. The experimental results show that the MCG botnet has the ability of automatic grouping and antitracking.

1. Introduction

A botnet consisting of a large number of computers is controlled by the botmaster. The botmaster can remotely control bots to initiate DDoS attacks, send spam, collect user privacy, and conduct other malicious activities [1]. With the popularity of Internet of Things (IoT) devices, IoT botnets like Mirai have been infecting as many devices as possible such as IP cameras and home routers. Because of the sheer volume and continual operation of many vulnerabilities (many users do not pay much attention to IoT update alerts and leave the configurations by default) of IoT devices, the population of an IoT botnet becomes increasingly tremendous. The growing population, though making a botnet powerful, results in an increased risk of exposure.

Although there has been quite a lot of research on the security of IoT devices [2–4], Mirai and its variant botnets still pose a huge threat to increasingly more IoT devices [5]. On the other side, when the botmaster faces a large-scale IoT botnet, how to effectively manage the group is a big challenge. Specifically, once a bot is captured, the command and control (C&C) channel may be cracked and then tracked, potentially rendering more bots being discovered. The disclosure of the bot program will lead to the cracking of key information such as communication patterns.

When designing a botnet, how the botmaster communicates to its bots is a problem of top priority [6]. Centralized communication is the simplest and most common method. The salient feature of this type of communication is that it has a command and control (C&C) server. The botmaster and its bots communicate through the server using protocols like IRC and HTTP. Centralized communication has the advantages of low delay, scalability, and large communication capacity. However, its disadvantages are also obvious. Once the C&C server is shut down or taken over, the entire botnet will most likely be destroyed. To alleviate this problem, the DGA algorithm is applied to the design of botnets. Through such an algorithm, the botmaster and bots can generate the same domain names hosting C&C servers.

However, if one obtains the bot program, it is feasible to reverse-engineer the DGA algorithm. Therefore, the DGA-based botnet could be blocked by predicting and registering domain names in advance. Moreover, the use of the DGA algorithm will cause bots to generate a large number of DNS (failure) queries, which is a distinguishable detection feature. With the aid of machine learning, defenders can easily identify this type of botnet through traffic characteristics [7].

To improve the stealthiness and survivability of botnet communication, more and more botnets use P2P to communicate. Compared with centralized botnets, P2P botnets effectively avoid the problem of single point of failure. However, P2P botnets are not absolutely reliable. Firstly, if a P2P botnet uses a proprietary P2P communication protocol, it is easy to be detected and shut down. At the same time, Sybil attacks and index poisoning will also affect the reliability of P2P botnets. In addition, the delay of the P2P network leads the bot to be not able to receive the commands from the botmaster in time.

To implement real-time, stealthy, and robust botnet communication, the new generation of botnets began to gradually use social networks and other public services as their C&C channels, such as Twitter, Facebook, and Cloud storage. However, no matter what communication channels are used, once the botnet is captured and reverse-engineered, the botnet faces two problems.

P1. How to ensure the survivability of botnets.

P2. How to manage the large number of bots without being tracked.

To solve these problems, this paper proposes an automated approach to group management of large-scale IoT bots. The basic idea of the proposed approach is to establish a reliable and unsuspicious social network-based C&C channel capable of automatically grouping bots, wherein a group of bots have a unique ID that is against cross-group tracking. The Diffie–Hellman key exchange method is leveraged for efficiently generating such a unique group ID, thereby scaling up automatic bot grouping.

The main contributions of this paper are as follows:

(i) We establish a reliable C&C channel through social networks for our MCG botnet. To verify the timeliness and reliability of our established C&C channel, we deploy a real-world testbed consisting of Twitter and GitHub.

(ii) We dissect the C&C channel in consideration of its fine-grained functionalities, namely, control channel, command channel, return channel, and registration channel. We analyze the functions and characteristics of these four channels in detail in comparison to the characteristics of existing social networks and public network services.

(iii) We leverage the Diffie–Hellman key exchange method for automatic bot grouping of the MCG botnet. To evaluate the grouping performance, we use more than 2,000 docker nodes to conduct experiments. The results show that as the number of bots increases, the bots could be effectively grouped in real time without the botmaster's intervention, enabling the botmaster to control each group in an isolated fashion with built-in antitracking capability.

The structure of this paper is as follows: Section 2 discusses related work. Section 3 elaborates the design and workflow of the MCG botnet. Section 4 conducts performance evaluation. Section 5 discusses countermeasures against the MCG botnet. We finally conclude the paper in Section 6.

2. Related Work

Botnets have been a hot research area. With the wide application of machine learning-based detection techniques, traditional centralized botnets (e.g., based on IRC and HTTP protocols) and P2P botnets could be detected under certain conditions [8–12].

To evade detection, more sophisticated C&C structures like hybrid P2P and more dynamic C&C resource management like URL Flux are used for improving botnet survivability [13]. Meanwhile, increasingly more botnets adopt social networks, such as Twitter, as C&C channels [14]. For example, Pantic et al. designed a botnet that exploits Twitter accounts as C&C channels, wherein the account names are automatically generated based on Markov chains. The botnet is of strong stealthiness due to its social network-based and automatically generated channels [15]. Yin et al. proposed a social network-based botnet with antidisruption capability [16]. The proposed botnet introduces a divide-and-conquer and automatic reconstruction mechanism to improve survivability. In addition to directly leveraging social networks as C&C channels, botmasters are incorporating steganography techniques, such as image steganography and Unicode steganography [17, 18], to further improve the stealthiness of botnets. Besides social networks, many Internet services are also used by botnets. For example, Google's C2DM service provides C&C channels for mobile botnets [19]; SLBot is a serverless botnet based on Service Flux [20]; the URL shortening services are used to evade network-level detection and blacklisting [21].

Social network-based botnets have been paid much attention to in the literature. Chew et al. proposed a social network-based hybrid botnet (RSHB) [22]. When RSHB uses social networks for communication, it also adds a reputation mechanism and a resurrection mechanism, which further improves the robustness of the botnet. In addition to social networks, blockchain has also become a focus for botnet designers. The op_return field of Bitcoin provides a certain storage space, which can transmit information simply and efficiently [23]. The ECDSA algorithm used by Bitcoin has also been used to build subliminal channels and has been proven feasible [24]. To improve the efficiency of ciphertext transmission and reduce the number of interactions, Luo et al. proposed a covert communication method based on Bitcoin transactions [25]. The method of communication using Bitcoin inevitably comes at the cost of transaction fees. To address this limitation, the Whisper protocol used in Ethereum allows botmasters to control bots at almost zero cost [26].

Existing studies provide covert communication between the botmaster and the bot. However, they do not take into account the situation that a bot using their covert communication is hacked or captured by the honeynet. The proposed MCG botnet focuses on how to effectively and automatically group large-scale botnets so to improve the antitracking capability. That is, capturing and containing some bots do not threaten the stealthiness of other bots and the survivability of the entire botnet.

3. Design

In this section, we will detail the MCG botnet architecture and its workflow.

We refer to the proposed botnet as a multichannel automatic grouping botnet (MCG botnet). The multichannel property means that the C&C channels are categorized into different types according to their functionalities in terms of botnet communication. Different types of channels are separate yet cooperative, beneficial to the robustness of the botnet. The MCG botnet has four types of channels, namely, registration, command, control, and return channels. The registration channel is used to collect the information of newly infected bots. Having a number of registered bots, the botmaster stores the command in the command channel, and then a bot could find the address of the command channel in the control channel so as to retrieve the command. After executing the retrieved command, the bot transmits the results to the botmaster through the return channel. The automatic grouping property enables bot group management hierarchically. That is, the parent group is in charge of the management of the child group. The taking down of one group does not affect its parent or child group (if the child group has already been generated).

3.1. Architecture.

Designing a botnet architecture needs to consider the communication requirements of different parties like the botmaster, the bot, and the C&C channels. However, the communication requirements of different parties may differ in various scenarios. For example, when launching DDoS attacks, the botmaster propagates commands to its bots in a timely manner due to the need of all bots simultaneously participating in the attack. In contrast, when bots transmit information like user privacy, the timeliness is not a top priority. According to the communication requirements of the MCG botnet (see Table 1), the MCG botnet is architected, comprising four channels, namely, command channel, control channel, return channel, and registration channel.

3.1.1. Command Channel and Control Channel.

Command and control (C&C) channel is a general term involving almost all aspects concerning how a botmaster controls his/her bots (e.g., transmitting commands, sending botnet configuration information). Existing studies do not consider command and control channels separately. Since the command channel and the control channel have distinct communication requirements, we design the two channels separately to fulfill their distinct communication requirements.

Command Channel. We use the command channel to store the commands released by the botmaster. When selecting a command channel, one should focus on the storage capacity as well as security.

Control Channel. We use the control channel to store the address of command channel. The bots obtain the address of the command channel through the control channel and finally obtain the commands stored by the botmaster in the command channel. Control channel should be considered from a reliability perspective.

The whole process is shown in Figure 1, which is divided into four stages.

Phase I (**1–3**): The botmaster stores ciphertext and signature in command channel.

Phase II (**4**): Convert the address of the command channel into a short URL and publish it to the control channel.

Phase III (**5**): The bots regularly obtain the short URL from the control channel.

Phase IV (**6–8**): The bots find the command channel through the short URL to extract information.

In Phase II, we are facing an issue with synchronizing the control channel address between the botmaster and its bots. Based on the previous experience of designing botnets [6], a botmaster generally uses DGA algorithm or similar algorithms, which brings new problems while completing synchronization. If the bot program is obtained by defenders, the seeds used will be obtained. It is fatal to botnets because defenders can predict subsequent addresses in the same way and block addresses that may be C&C channels in advance.

The fundamental problem is that no matter how the botmaster generates new addresses, they will be considered malicious. Therefore, this paper adopts social network as the C&C channel to avoid this problem. The botmaster and its bots have agreed on the designated social network user in advance. The botmaster will convert the address of the command channel into a short URL and send it to the comments section of the designated user's blog post. The bots only need to periodically read the comments section of the designated social network user to obtain the command channel address. Then the bots obtain the commands stored in advance by the botmaster from the command channel. Neither social network administrators nor defenders can block a normal user with just a malicious comment. Therefore, the MCG botnet does not need to consider the synchronization between botmaster and bot. For the MCG botnet, it is necessary to find a social network that can retain malicious comments from users for a long time as a control channel.

Table 1: Channel features.

Channel	Timeliness	Security	Anonymity	Reliability	Capacity
Control	*Y*	—	—	*Y*	—
Command	—	—	—	—	*Y*
Return	—	—	*Y*	*Y*	*Y*
Registration	—	*Y*	—	—	—

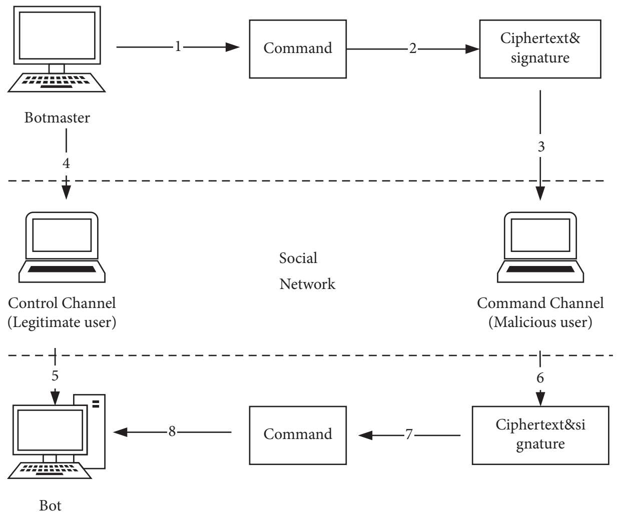


Figure 1: Command and control channel design.

3.1.2. Return Channel. Return channel is used by bots to transmit information to the botmaster. When the botmaster uses its bots to steal user privacy, the bots need to send information to the botmaster frequently, and the amount of information may be relatively large. Therefore, the return channel should be easy to register and use. As the receiver, the botmaster should focus on the issue of anonymity to avoid being traced by defenders.

3.1.3. Registration Channel. The bot that infects normal devices is called the parent bot, and the new infected bot is called the child bot. In the MCG botnet, the child bot needs to rely on the parent bot to complete the registration so as to formally join the botnet. The botmaster obtains the scale of the entire botnet through the registration channel.

In recent years, defenders have generally adopted technologies such as honeypots and honeynets to monitor, detect, and analyze the entire botnet. Therefore, the registration channel should ensure real time and reliability and avoid the leakage of some key information as much as possible. This is important for grouping of botnets.

So far, we detailed the four channels of the MCG botnet. As shown in Table 1, we summarize the four channels from the aspects of timeliness, security, anonymity, reliability, and large capacity requirements (Y: there is a clear demand for this indicator; —: there is no clear demand for this indicator).

3.2. Bot Grouping. The ever-expanding botnet also brings two new problems to the botmaster:

P1. How the botmaster can control part of the botnet more accurately when the botnet scale is large.

P2. Once a bot is captured, the information such as the C&C channel used by the botnet will be leaked, potentially making the entire botnet taken down/over.

To address the above problems, we propose the following techniques:

(1) Bot grouping: Child bots will be divided into new groups by parent bots without intervention by the botmaster. The botmaster can also adjust the scale of each group of bots in real time according to the size of the botnet.

(2) Botnet group management: To avoid the harm caused by the bot being captured, this paper generates exclusive configuration information for each group of the MCG botnet. Once a group of bots are mixed with honeypots, the configuration information leakage will not affect other groups.

The Diffie–Hellman key exchange method is applied for automatic bot grouping, as is shown in Figure 2. First, the botmaster and its bots generate random numbers a and b, respectively. P and G are two prime numbers shared between the botmaster and its bots. They are used to generate A and B. Finally, the botmaster and each bot exchange A and B to generate K, the group ID of the bot. Bots in the same group generate the same random number b.

If any one of a, b, P, and G changes, the shared key K will change accordingly. In the MCG botnet, as long as different groups use different random numbers b, the botmaster and its bots can obtain different K.

It has been explained in Section 3.1 that the child bots can only rely on the parent bots to register. The parent bots will generate exclusive configuration information for the child bots, including encryption and decryption keys and random number b. Subsequently, the parent bots send this information to the botmaster to complete the registration of the child bots. Parent bots and child bots belong to different groups. Parent bots update the random number b regularly to achieve the effect of automatic grouping. The botmaster can also actively request the parent bots to generate a new random number b for the child bots according to the scale of the botnet to complete the scale control of different groups.

Regardless of whether the botmaster intervenes, the random number b is only generated by the parent bots. Then it is sent to botmaster to avoid security risks caused by synchronization. Assuming that a child bot is a honeypot since the bots of different groups use different encryption and decryption keys, defenders can only crack the communication content between the bots of the current group and the

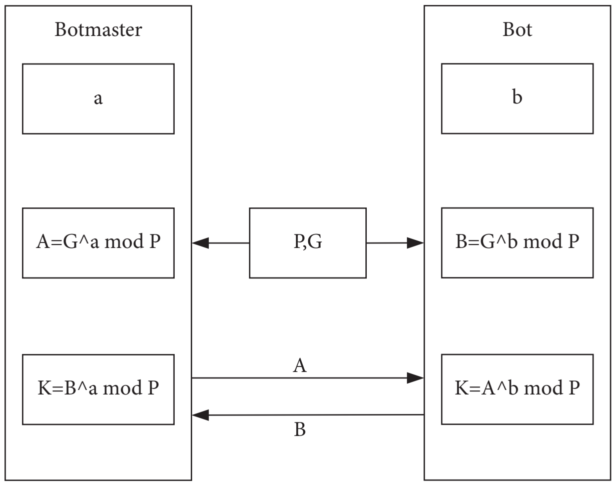


Figure 2: Group ID generation.

botmaster through this honeypot. For a large-scale botnet, the impact is very limited. A large-scale botnet refers to a botnet that consists of a large number of bots, and the bots usually are geographically dispersed in various networks.

3.3. Workflow. When the MCG botnet is initialized, the botmaster will first generate two prime numbers P and G, random number a, and public key A required by the Diffie–Hellman key exchange method. Then the botmaster generates the private key Priv$K$ for signing the ciphertext and the corresponding public key Pub$k$. To reduce the harm caused by key leakage, each group uses a different key for encryption and decryption. Both symmetric and asymmetric encryption algorithms could be used, for example, AES algorithms and RSA algorithms. When the amount of transmitted information is small, asymmetric encryption algorithms could be used. Otherwise, symmetric encryption algorithms could be used.

P, G, A, and Pub$k$ are hardcoded into the bot program. Pub$k$ is used to verify the botmaster's signature. It is different from the RSA key used for encryption and decryption in the bot group. When the MCG botnet is first deployed, the botmaster generates different random numbers b and RSA keys for them according to different groups. The same group of bots uses the same random number and key. The working flow chart of the MCG botnet is shown in Figure 3.

(1) The botmaster uses the RSA public key corresponding to the bot group to encrypt the information. Then Pri$K$ is used to sign and store to command channel.

(2) The botmaster converts the address of the command channel into a short URL and sends it to the control channel, which is the comments section of the designated social network user.

(3) The bots regularly querie the control channel to obtain the address of the command channel.

(4) The bots find the command channel according to the address obtained in the previous step and obtain the information transmitted by the botmaster.

(5) The execution result is transmitted to the botmaster through the return channel.

(6) After a bot (parent bot) infects a new bot (child bot), it copies its own program to the new bot and passes it to the random number b and RSA key (not the random number b and RSA key used by the current bot group).

(7) After step (6), the parent bot will complete the registration of the child bot through the registration channel.

Different bots may originate from different groups. Consequently, when communicating with a bot group, the botmaster should use the public key of the corresponding bot group to encrypt the commands.

For example, Group-A, Group-B, and Group-C are three bot groups that use different public/private key pairs. When the botmaster commands bots in Group-A to perform tasks, it encrypts commands using Group-A's public key. The botmaster sends the encrypted commands to bots in Group-A through C&C channels (e.g., the channel proposed in Sec. 3.1.1). Bots in Group-A, Group-B, and Group-C will receive the encrypted commands, but only bots in Group-A can decrypt them. After bots in Group-A execute the commands, the execution result is transmitted to the botmaster through the return channel.

The group size during automatic grouping of our proposed MCG botnet can be accomplished in a predefined manner. Consider that bots in Group-A infect new bots to constitute Group-B. Given the size of Group-A S_a, the botmaster just needs to define the maximum number of bots that each bot in Group-A could infect (e.g., n) to control the upper bound of the size of Group-B (i.e., $S_a n$). We control the upper bound because in most cases a large group size would lead to the taking down of many bots. However, since bots in Group-A may infect overlapping sets of bots, the actual size of Group-B may be much smaller than the upper bound. In such a case, the botmaster could assign a larger value to n to further increase the upper bound. Note that if one keeps n constantly exceeding 1, the child group size may grow larger than the parent group size. To stop the group size growing unlimitedly, the botmaster normally needs to make n decrease from generation to generation.

Consider that a bot in Group-A infects a set of new bots. Such a set of new bots, along with those infected by all the remaining bots in Group-A, constitute a new group named Group-B. Group A is the parent group, while Group-B is its child.

When the bots in Group-A infect a new bot, they generate a grouping random number, which is the same across all the bots in Group-A due to the same seeding mechanism, as well as a symmetric key (or a private/public key pair), for Group-B. The grouping random number and the key will be sent to all the bots in Group-B directly, and to the botmaster through the registration server, by bots in Group-A. The random number is used to generate the ID of Group-B, and the key is for encrypting the communication message between the botmaster and the bots in Group-B.

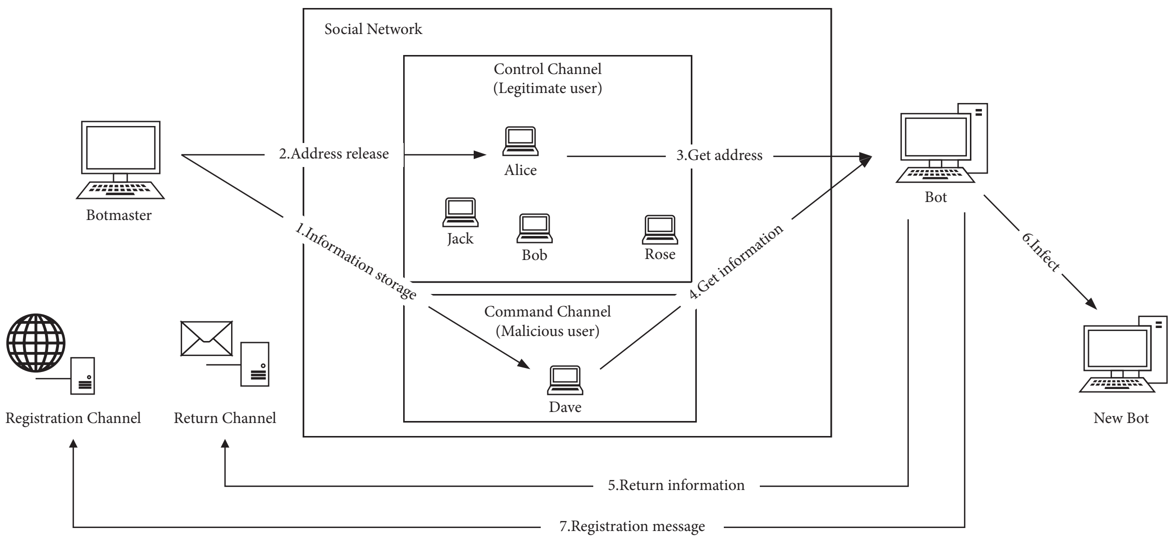


Figure 3: The overall workflow of the MCG botnet.

The exposure of the key would affect the confidentiality of the communication between the botmaster and the bots in Group-B. However, the affection is limited because the communication between the botmaster and the bots in other groups remains confidential.

4. Experiment

To evaluate the MCG botnet, we first selected and verified the four channels and finally conducted a simulation experiment on the botnet automatic grouping.

4.1. Channel Selection. In Section 3.1, we have described the functions and requirements of the four channels in detail. As shown in Table 2, to select the appropriate channel, we investigated popular social networks and e-mail services. In Table 2, the "Register" column represents whether using the service is conditioned on registration. The "High Capacity" means whether uploading programs, pictures, and other large-sized objects is allowed.

The control channel uses the comments section of legitimate users in the social network to transmit the address of the command channel. The two following issues should be considered: (1) The social network allows the botmaster to comment freely in the comments section of other people's accounts. (2) Social networks do not strictly control comments.

The bots obtain the address of the command channel from the control channel and obtain the commands and so forth. Therefore, the command channel should have a larger capacity. As shown in Figure 4, we use GitHub as the command channel and Twitter as the control channel.

Among all channels, the command channel and the control channel are crucial to the robustness of the MCG botnet. To verify the robustness, we measure the survival times of the command channel over GitHub and the control channel over Twitter.

As Figure 5 shows, we register one GitHub account to store the ciphertext of the command "DDoS www.test.com" along with the signature of the bomaster and one Twitter account to post the address of the command at the comments sections of those Twitter accounts that our registered account follows. The address of the command is a short URL (compressed by TinyURL [27]) locating "DDoS https://www.test.com" at GitHub. Note that there are 20 Twitter accounts that our registered account follows, and these followed accounts are all news accounts due to their long time stability that is beneficial to short URL propagation. We post one short URL at the comments section of each of the 20 news Twitter accounts and measure the survival times of the 20 posted short URLs (i.e., 20 control channels). Also, we measure the survival time of the control channel, the time that the posted short URLs are accessible. Figure 5 shows the results. We see that all the 20 control channels and one command channel survive for the entire 72 hours, indicating the robustness of the MCG botnet.

As shown in Table 3, we checked the command channel and the control channel every 24 hours and found that the survival rate was 100%, and comments on Twitter can be accessed immediately. This experiment demonstrates the real time and robustness of using the Twitter user comments section as the control channel and GitHub as the command channel.

The return channel is used by the bots to actively send information to the botmaster so the botmaster should be more careful to avoid leaking too much personal information. The botmaster can use e-mail as the receiver and configure different sending mailboxes for different bot groups through the command channel. The registration channel is used for the registration of new bots, and the communication volume is small. Due to the needs of botnet grouping, it is necessary to ensure high security and real-time performance. Third-party services are usually difficult to achieve. Therefore, we use a private server as a registration channel.

Table 2: Network service properties.

Type	Name	Register	High Capacity
Social network	Twitter	*Y*	*N*
	Facebook	*Y*	*N*
	Instagram	*Y*	*N*
	Weibo	*Y*	*N*
	GitHub	*Y*	*Y*
E-mail service	Gmail	*Y*	*Y*
	Outlook	*Y*	*Y*
	YOPmail	*N*	*N*
Cloud storage	OneDrive	*Y*	*Y*
	Dropbox	*Y*	*Y*
	Pan.baidu	*Y*	*Y*

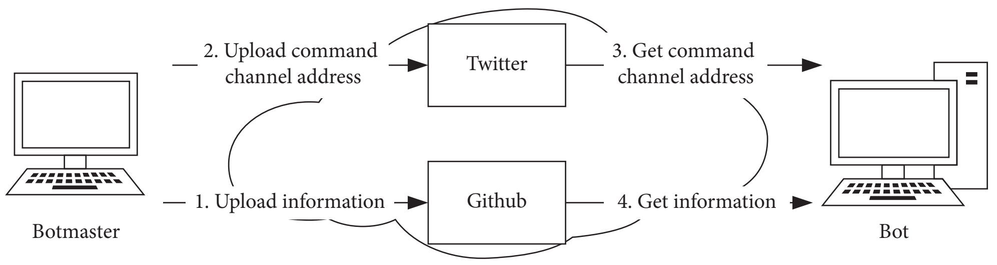
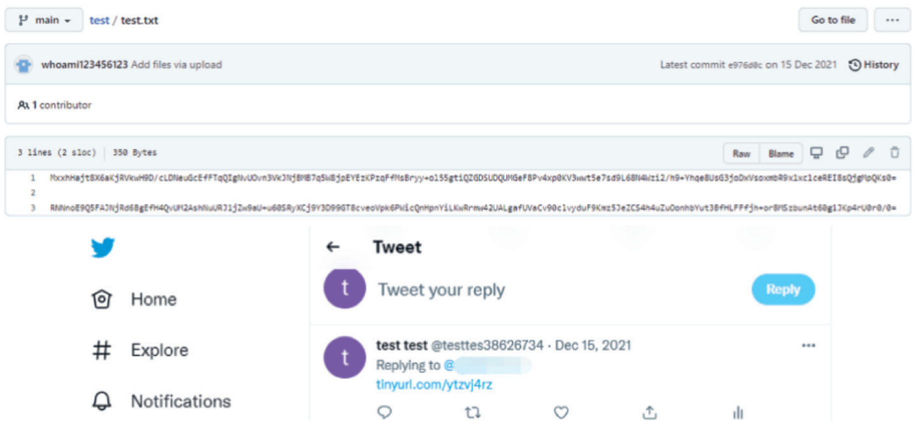


Figure 4: Real-world control channel and command channel testbed.



whoami123456123 Add files via upload — Latest commit e976d8c on 15 Dec 2021

1 contributor

3 lines (2 sloc) | 350 Bytes

```
MxxhHajt8X6aKjRVkwH9D/cLDNeuGcEfFTqQIgNvUOvn3VkJNjBMB7q5W8jpEYEzKPzqFfMsBryy+o155gtiQZGDSUDQUMGeF8Pv4xp0KV3wwt5e7sd9L68N4Wzi2/h9+Yhqe8UsG3joDxVsoxmbR9x1xc1ceREI8sQjgMpQKs0=

RNNnoE9Q5FAJNjRd6BgEfH4QvUM2AshNuUR J1jZw9aU+u60SRyXCj9Y3D99GT8cveoVpk6PWicQnHpnYiLKwRrmw42UALgafUVaCv90c1vyduF9Kmz5JeZCS4h4uZuOonhbYut3BfHLFFfjh+orBMSzbunAt60g1JKp4rU0r0/0=
```

test test @testtes38626734 · Dec 15, 2021

Replying to @[redacted]

tinyurl.com/ytzvj4rz

Figure 5: The GitHub account to store the ciphertext of the command and the Twitter accounts where we post the address of the command (i.e., short URL).

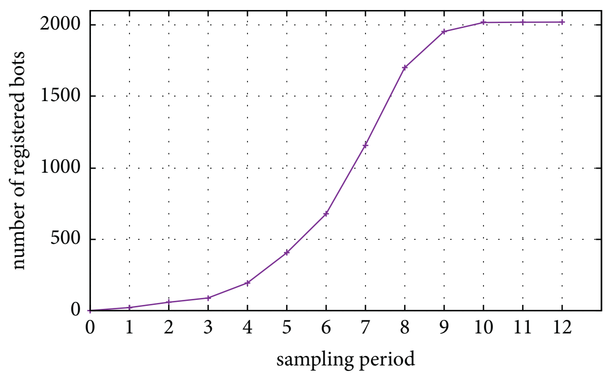
Table 3: Survival times versus the number of command and control channels.

Survival time (hours)	24	48	72
Number of command channels	1	1	1
Number of control channels	20	20	20

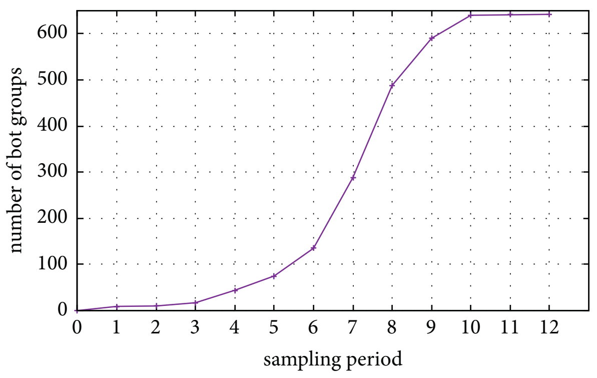
We have proven the robustness of the command channel and the control channel. Even if the return channel and registration channel are blocked, the botmaster can still contact its bots by changing the e-mail and server address through the command channel and the control channel.

4.2. Automatic Grouping. To verify the grouping effect of the MCG botnet, this paper uses 2020 docker nodes to simulate IoT devices to carry out simulation experiments. Using six high-performance services, the CPUs are all Intel(R) Xeon(R) Gold 6248R CPU @ 3.00 GHz, and the RAMs are all DDR4, 503G. The operating systems are Ubuntu20.04, Ubuntu18.04, and Ubuntu16.04.

During initialization, 5 of the docker nodes are initialized as bots, which are 5 groups, respectively. As to the C&C method in the docker node experiment in Section 4.2, our major goal is to verify the automatic bot grouping capability, which is independent of the specific C&C method. Therefore, in the docker node experiment, we use neither the



(a)



(b)

Figure 6: Automatically grouping experiments. (a) Changes in the number of bots. (b) Changes in the number of bot groups.

Table 4: Statistics of bot grouping.

Number of bots in a single group	≥15	≥10	≥5	≥1	Total
Number of groups	5	24	107	506	642
Number of bots	92	272	703	953	2020

proposed C&C method nor any other methods. As shown in Figure 6(a), the number of registrations of bots increases exponentially. Due to the limited number of docker nodes, the number of bots will gradually stabilize. Therefore, the number of registrations of bots and the number of groups show an overall S-shape. As shown in Table 4, it can be seen that, with the continuous increase of bots, the number of bot groups has also increased sharply.

In the initial stage (period: 0–3), due to the small number of bots, the speed of new bots joining the botnet is slower and the number of bot groups is small. With the increase in the number of bots, it will soon enter the explosive stage (period: 3–8), and bots show an exponential growth trend. At this stage, due to the large increase in bots, the number of groups also shows an exponential growth trend. However, at this time, most bot groups contain fewer bots. As shown in Table 4, there are a total of 136 groups with bots greater than or equal to 5, as well as a total of 1067 bots, which is more than half of the total bots.

As shown in Figure 6(b), it can be seen from the simulation experiment that the MCG botnet has better automatic grouping ability. Even if devices such as honeypots join the botnet, a honeypot can only affect one group at most and will not interfere with the normal operation of the entire botnet. This simulation experiment shows that the MCG botnet greatly improves survivability through automatic grouping.

5. Countermeasures

Compared with other botnets, the MCG botnet's main features are as follows: Automatic grouping botnet is to reduce the harm caused by bot program leakage. The communication between the botmaster and its bots is disassembled according to function, and the social network is used as the control channel. A reliable C&C channel is built as the basis for botnet communication.

Therefore, the focus of detecting the MCG botnet should be to strengthen the monitoring and censorship of social networks. Identity authentication is also an effective approach when using public services. If social network platforms and users can strengthen the review of the comments section, especially some comments that are not relevant to blog posts, it will pose a greater threat to such botnets. In addition, public services that are free to use and registration-free should use verification codes and other methods to verify the user's identity.

When we use news accounts as control channels, the bot needs to access them regularly to get the address of the command channel. This can lead to a rise in page views of these news accounts, attracting the attention of defenders. If a bot is captured (e.g., by honeynet), the social network accounts used by the MCG botnet will be exposed. Defenders can report these accounts to interrupt the operation of the MCG botnet. However, there are many Twitter accounts and a huge number of many other social network accounts that the botmaster can choose as control channels. More importantly, a parent group is able to dynamically define the social network account that the bots in its child group should use, making the control channels of different bot groups completely isolated. In other words, defenders may interrupt the operation of certain groups, but the remaining groups would be still working.

The MCG botnet has group management and anti-tracking capabilities. However, like most botnets, when the MCG botnet performs malicious actions such as DDoS attacks, a large number of bots will generate very similar traffic behaviors. This is very critical to combat botnets. Defenders can collect enough traffic information. Then they can analyze key information such as botnet communication methods, network topology, botnet size, and types of infected devices through machine learning and data mining. Further, they cooperate with relevant social network service providers or equipment manufacturers to propose targeted measures against such botnets.

6. Conclusion

This paper proposed a multichannel self-grouping social botnet called MCG botnet. According to the communication requirements of the MCG botnet, the MCG botnet is architected comprising four channels, namely, command channel, control channel, return channel, and registration channel. A botnet grouping strategy is designed based on the Diffie–Hellman key exchange method. It greatly enhanced the MCG botnet's survivability, since containing one group does not affect the survivability of all the remaining groups. We perform experiments on Twitter and GitHub to verify the robustness of the C&C channel in a real environment. We also set up a simulation environment containing 2020 docker nodes to test the grouping ability of the botnet. It has propounded implications on personalized group control and antitracking of large-scale IoT botnets.

Data Availability

The original data in the research process can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest in the manuscript.

Acknowledgments

This work was supported in part by National Natural Science Foundation (61972313), Postdoctoral Science Foundation (2019M663725 and 2021T140543), and the Fundamental Research Funds for the Central Universities, China. Xiaobo Ma is an XJTU Tang Scholar supported by Cyrus Tang Foundation.

References

[1] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pp. 41–52, Rio de Janeiro, Brazil, October 2006.

[2] B. K. Mohanta, D. Jena, U. Satapathy, and S. Patnaik, "Survey on IoT security: challenges and solution using machine learning, artificial intelligence and blockchain technology," *Internet of Things*, vol. 11, Article ID 100227, 2020.

[3] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–782 743, 2019.

[4] R. Vinayakumar, M. Alazab, S. Srinivasan, Q.-V. Pham, S. K. Padannayil, and K. Simran, "A visualized botnet detection system based deep learning for the internet of things networks of smart cities," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4436–4456, 2020.

[5] S. Soltan, P. Mittal, and H. V. Poor, "Blackiot: iot botnet of high wattage devices can disrupt the power grid," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 15–32, Princeton University, Princeton, NJ, USA, 2018.

[6] G. Vormayr, T. Zseby, and J. Fabini, "Botnet communication patterns," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2768–2796, 2017.

[7] N. Jiang, J. Cao, Y. Jin, L. E. Li, and Z.-L. Zhang, "Identifying suspicious activities through dns failure graph analysis," in *Proceedings of the 18th IEEE International Conference on Network Protocols*, pp. 144–153, Kyoto, Japan, October 2010.

[8] A. H. R. A. Awadi and B. Belaton, "Multi-phase irc botnet and botnet behavior detection model," *International Journal of Computer Applications*, vol. 66, 2015.

[9] G. Sagirlar, B. Carminati, and E. Ferrari, "Autobotcatcher: blockchain-based p2p botnet detection for the internet of things," in *Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pp. 1–8, IEEE, Philadelphia, PA, USA, October 2018.

[10] T. Sengupta, S. De, and I. Banerjee, "A closeness centrality based p2p botnet detection approach using deep learning," in *Proceedings of the 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1–7, IEEE, Kharagpur, India, July 2021.

[11] S. Y. Yerima and M. K. Alzaylaee, "Mobile botnet detection: a deep learning approach using convolutional neural networks," in *Proceedings of the 2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pp. 1–8, IEEE, Dublin, Ireland, June 2020.

[12] S. Almutairi, S. Mahfoudh, S. Almutairi, and J. S. Alowibdi, "Hybrid botnet detection based on host and network analysis," *Journal of Computer Networks and Communications*, vol. 2020, Article ID 9024726, 16 pages, 2020.

[13] C. Liu, W. Lu, Z. Zhang, P. Liao, and X. Cui, "A recoverable hybrid c&c botnet," in *Proceedings of the 2011 6th International Conference on Malicious and Unwanted Software*, pp. 110–118, IEEE, Fajardo, PR, USA, October 2011.

[14] J. Zhang, R. Zhang, Y. Zhang, and G. Yan, "The rise of social botnets: attacks and countermeasures," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1068–1082, 2016.

[15] N. Pantic and M. I. Husain, "Covert botnet command and control using twitter," in *Proceedings of the 31st Annual Computer Security Applications Conference*, pp. 171–180, Los Angeles, CA, USA, December 2015.

[16] T. Yin, Y. Zhang, and S. Li, "Dr-snbot: a social network-based botnet with strong destroy-resistance," in *Proceedings of the 2014 9th IEEE International Conference on Networking, Architecture, and Storage*, pp. 191–199, IEEE, Tianjin, China, August 2014.

[17] S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, and N. Borisov, "Stegobot: a covert social network botnet," in *International Workshop on Information Hiding*-Springer, Berlin, Heidelberg, 2011.

[18] A. Compagno, M. Conti, D. Lain, G. Lovisotto, and L. V. Mancini, "Boten elisa: a novel approach for botnet c&c in online social networks," in *Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 74–82, IEEE, Florence, Italy, September 2015.

[19] S. Zhao, P. P. Lee, J. C. Lui, X. Guan, X. Ma, and J. Tao, "Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service," in *Proceedings of the 28th Annual Computer Security Applications Conference*, pp. 119–128, Orlando, Florida, USA, December 2012.

[20] D. Wu, B. Fang, J. Yin, F. Zhang, and X. Cui, "Slbot: a serverless botnet based on service flux," in *Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pp. 181–188, IEEE, Guangzhou, China, June 2018.

[21] S. Lee and J. Kim, "Fluxing botnet command and control channels with url shortening services," *Computer Communications*, vol. 36, no. 3, pp. 320–332, 2013.

[22] C.-J. Chew, Y.-C. Chen, J.-S. Lee, C.-L. Chen, and K.-Y. Tsai, “Preserving indomitable ddos vitality through resurrection social hybrid botnet,” *Computers & Security*, vol. 106, Article ID 102284, 2021.

[23] S. T. Ali, P. McCorry, P. H.-J. Lee, and F. Hao, “Zombiecoin: powering next-generation botnets with bitcoin,” in *International Conference on Financial Cryptography and Data Security*, pp. 34–48, Springer, Berlin, Heidelberg, 2015.

[24] D. Frkat, R. Annessi, and T. Zseby, “Chainchannels: private botnet communication over public blockchains,” in *Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1244–1252, IEEE, Halifax, NS, Canada, July 2018.

[25] X. Luo, P. Zhang, M. Zhang, H. Li, and Q. Cheng, “A novel covert communication method based on bitcoin transaction,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2830–2839, 2021.

[26] M. Baden, C. F. Torres, B. B. F. Pontiveros, and R. State, “Whispering botnet command and control instructions,” in *Proceedings of the 2019 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 77–81, IEEE, Rotkreuz, Switzerland, June 2019.

[27] Tiny, “Tinyurl.com - shorten that long url into a tiny url,” 2020, https://tinyurl.com/.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 4593002, 11 pages
https://doi.org/10.1155/2022/4593002

Research Article

Natural Backdoor Attacks on Deep Neural Networks via Raindrops

Feng Zhao,[1] **Li Zhou**,[1] **Qi Zhong**,[2] **Rushi Lan**,[1] **and Leo Yu Zhang**[2]

[1]*Guangxi Key Laboratory of Image and Graphic Intelligent Processing, Guilin University of Electronic Technology, Guilin 541004, China*
[2]*School of Information Technology, Deakin University, Geelong, VIC 3216, Australia*

Correspondence should be addressed to Qi Zhong; zhongq.lk@outlook.com and Rushi Lan; rslan2016@163.com

Received 12 December 2021; Accepted 1 March 2022; Published 26 March 2022

Academic Editor: Weizhi Meng

Recently, deep learning has made significant inroads into the Internet of Things due to its great potential for processing big data. Backdoor attacks, which try to influence model prediction on specific inputs, have become a serious threat to deep neural network models. However, because the poisoned data used to plant a backdoor into the victim model typically follows a fixed specific pattern, most existing backdoor attacks can be readily prevented by common defense. In this paper, we leverage natural behavior and present a stealthy backdoor attack for image classification tasks: the raindrop backdoor attack (RDBA). We use raindrops as the backdoor trigger, and they are naturally merged with clean instances to synthesize poisoned data that are close to their natural counterparts in the rain. The raindrops dispersed over images are more diversified than the triggers in the literature, which are fixed, confined, and unpleasant patterns to the host content, making the triggers more stealthy. Extensive experiments on ImageNet and GTSRB datasets demonstrate the fidelity, effectiveness, stealthiness, and sustainability of RDBA in attacking models with current popular defense mechanisms.

1. Introduction

Internet of Things (IoT) devices have infiltrated many industries and are now part of our daily lives. As a consequence, a vast amount of data will be created. Deep learning technology has been shown to be extremely effective in processing enormous volumes of high-dimensional data, and it is now widely employed in a variety of IoT applications, such as intelligent driving [1], computer vision [2,3], natural language processing [4–6], and *etc*. In another aspect, neural networks used in IoT are evolving into deeper and wider architectures to perform well in various tasks. This indicates that there are massive parameters to learn and a lot of computing resources to consume. Such requirements boost the development of machine learning-related industries, including machine learning as a service (MLaaS). In essence, various giant companies, such as Google and Amazon, have launched their own MLaaS platforms to facilitate users to outsource their model training projects. However, security vulnerabilities in deep neural networks may arise in any stage of the supply chain, including but not limited to unprotected open channels, unreliable data sources, and unreliable training processes.

Studies have shown that deep neural networks are prone to attacks from different stages, including inference-stage attacks [7–9] and training stage attacks. Inference-stage attacks, best known for adversarial attacks [10–13], generally aim to mislead the deep neural network to produce high-confidence error prediction results for the test data during inference. They are usually achieved by adding subtle input-specific perturbations to the test data before querying the target model.

Training-stage attacks usually refer to backdoor attacks [14], which intend to manipulate the model predictions for those attacker-specified instances by poisoning some normal training data. Specifically, attackers inject some patterns, dubbed triggers, into clean samples. These modified data are also referred to as poisoned data or backdoor samples, and they are assigned predefined target labels. By involving backdoor samples and normal data for model training, the trained model is thus embedded with backdoors. At the inference stage, the backdoored model performs normally on benign inputs but predicts the target labels for instances containing triggers. This type of attack is stealthy since the victim model has state-of-the-art performance on clean inputs, which is indistinguishable from its corresponding clean model, while its backdoor behavior can only be activated by attacker-specified (unknown) inputs. In a nutshell, the potential malicious behavior could result in dire consequences in some security-critical areas, such as autonomous driving [15], face recognition [16], and speaker recognition [17–19], which will also cause serious obstacles to the DNN deployment and development.

In the literature, the most popular and effective backdoor triggers are simple, fixed, or unpleasant patterns, i.e., different clean data are patched with the same trigger in a fixed position without considering the host data content. In addition, the poisoned instances are generated by simply stamping triggers into benign host samples. For example, the trigger of BadNets [14], as shown in Figure 1, is a black and white pixel block in the bottom right corner of an image. Such poisoned data will inevitably have abnormal distributions and appear unnatural, raising the suspicions of model developers/users. They can be easily filtered out before the model training stage or rejected before the model inference. On the other hand, researchers have made tremendous efforts to improve the robustness of DNN models, and various backdoor countermeasures have been proposed to remove or suppress the backdoor behaviors of DNN models. It has been demonstrated that most of the existing backdoor methods can be successfully alleviated by some current popular defenses, e.g., fine-tuning [20], fine-pruning [21], and Grad-CAM based defenses [22,23].

Based on this understanding, in this work, we introduce a novel, simple, but effective backdoor attack method using raindrops, dubbed raindrops backdoor attack (RDBA). Specifically, we perform two different blur operations on uniformly distributed noise to simulate water droplets in real scenes with different sizes and directions and make them have motion blur. We then merge the raindrops trigger with a small portion of clean training samples to generate natural-looking poisoned data. Finally, the to-be-produced backdoored model is obtained through a generic training procedure.

Compared with the existing backdoor injection approaches, RDBA has the following advantages: (1) the raindrops are evenly distributed across clean samples and their natural features blend well with these host data, so the poisoned instances can hardly be distinguished by naked eyes or Grad-CAM based methods; (2) the backdoor triggers are crafted based on natural phenomenon raindrops rather than on some unpleasant patterns (e.g., BadNets) or obvious outliers (e.g., Blending), implying that the poisoned instances are closer to natural inputs and a misclassification caused by RDBA could be considered a normal misclassification.

In summary, the main contributions of this paper are as follows:

(i) In this paper, we propose a backdoor attack method RDBA, which uses natural behavior raindrops to embed backdoors in image classification scenarios

(ii) We simulate natural raindrops through a series of blur transformations on uniformly distributed noise, and the resulting raindrops are evenly dispersed and naturally blended on each clean sample

(iii) Two datasets are used to evaluate the fidelity, effectiveness, stealthiness, and sustainability of the proposed method in attacking two neural networks with and without defenses

The rest of this article is organized as follows. Section 2 discusses related work. Section 3 introduces the threat model and attack goals. Section 4 elaborates on the details of the proposed RDBA method. The experimental results are analyzed in Section 5 and the conclusions are drawn in Section 6.

2. Related Work

2.1. Backdoor Attacks. Backdooring DNNs refers to a technique that is able to maliciously manipulate the model predictions on specific inputs by poisoning a small portion of clean data during training or fine-tuning. Backdoor attacks have posed serious threats to the model supply chain and have attracted lots of attention from both industry and research community. Specifically, the attacker-crafted triggers are injected into some benign training data to create poisoned samples. In the inference, a neural network trained on these poisoned data will active abnormal behaviors when feeding inputs containing the triggers but behave normally on benign inputs.

Existing backdoor attacks can be divided into poison-label attacks and clean-label attacks according to whether the labels of poisoned samples are changed. In poison-label attacks, the labels of poisoned data are replaced with predefined target labels. As a result, when the backdoored model detects the triggers, its predictions will be the target labels. One of the most popular works that revealing backdoor threat in the machine learning training stage is the BadNets [14]. The authors used a simple binary pixel block at bottom right corner of the image as the backdoor trigger, as shown in Figure 1, and the poisoned data was created by stamping the trigger to a benign instance and changing its label to the target label. However, such kind of attack can be easily detected either by human inspection or by backdoor detection mechanisms due to the fact that triggers are outliers of the host images. As a remedy, Chen et al. in [24] blended triggers with benign images to generate poisoned images, as shown in Figure 1, where the *hello kitty* trigger overlaps with clean samples with a certain transparency. However, the above discussed triggers are fixed patterns in fixed locations, as will be shown in Section 2.2, most existing poison-label backdoor attacks are easily mitigated by current popular defenses.

In clean-label backdoor attacks, the poisoned training data still preserve their ground-truth labels, and they look like their source instances in the input space or at a pixel

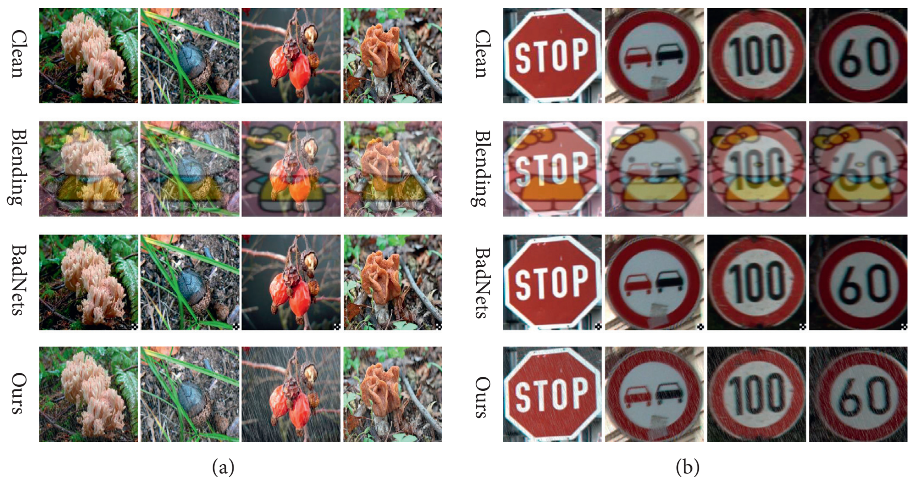


FIGURE 1: Different backdoor instances. The trigger crafted by the BadNets is a black and white pixel block at the bottom right hand corner of the image. In the specific case generated by the Blending, the "hello kitty" is used as the trigger to overlap with clean samples. In the case generated by RDBA, the trigger is evenly distributed raindrops.

level. One typical work was proposed by Shafahi et al. in [25], in which the authors crafted poison data by adding unnoticeable perturbations to clean samples from the target class of the training set. The poisoned data appear like their clean data counterparts, however, in latent feature space, they are closer to the target samples (i.e., the clean data from a certain non-target class of the testing set). During inference, the targeted inputs would be misclassified as the target class. Zhu et al. in [26] pointed out that the work of [25] is not suitable in the black-box setting because the victim network is not accessible. And they proposed an improved version of [25] in [26] by leveraging the convex polytope attack to craft poison data. Since the content of the poisoned data is consistent with their labels, these data will be considered as benign samples even by human inspection. Accordingly, clean-label attacks are more stealthy compared with poison-label attacks. However, it may be because the trigger is a particular set of testing data rather than a universal pattern, the attack success rate of clean-label attacks is relatively low, e.g., in [25], for a 10-class classification setting, the target success rate is 60%. Accordingly, in this work, we only target the poison-label attacks and put forward a backdoor method based on raindrops, which will be introduced in Section 4.

2.2. Backdoor Defenses. Backdoor defenses aim to detect or mitigate backdoor attacks before or after model deployment. In the current literature, various techniques have been proposed. Here, we list the main-stream techniques for backdoor defense before the deployment of DNN models, which are the most commonly used in practice.

2.2.1. Fine-Tuning. Fine-tuning is one of the practical and lightweight choices to get a well-performed model when the labeled training data is insufficient. Researchers have found that deep learning models suffer from catastrophic forgetting [20] of previously learned tasks when training on a series of new tasks. The rationale for employing fine-tuning to defend against backdoor attacks is that learning new tasks will generally lead to large changes of the model weights, which will disrupt previously learned trigger representations [20]. The fine-tuning defense takes advantage of this catastrophic forgetting phenomenon to drive the victim model to forget the implanted backdoors. That is, if defenders train a model on the top of the victim using some new clean training data, then the resultant model may drain out of memory and forget the backdoor since it does not encounter any triggers from new data during fine-tuning. However, in reality, fine-tuning alone is not always effective as expected when it comes to defense backdoors. This is because, neurons associated with a backdoor are disentangled from neurons associated with the original tasks, and their weights have little contribution to the original (or new) tasks. During fine-tuning, the weights of backdoor-related neurons will keep unchanged due to lack of driven-force, and the backdoor remains.

2.2.2. Fine-Pruning. Further to fine-tuning, the authors of [14] observed that the neurons activated by benign samples and those activated by trigger-containing samples do not overlap. In other words, there are neurons that can only be activated by triggers and remain dormant when inputs are benign data. In view of this observation, removing these trigger-sensitive neurons, dubbed backdoor neurons, can help to disable the backdoor without impairing model performance on normal data, as suggested by the neuron pruning defense hypothesis. However, Liu et al. in [21] further found that the subset of neurons activated by benign inputs and the subset of neurons activated by malicious inputs can overlap. Backdoors can also be triggered by suppressing neurons activated by benign inputs. In this case, pruning neurons alone will inevitably result in performance loss on benign inputs. Considering such drawbacks, Liu et al. proposed the fine-pruning defense in [21], which combines the merits of neuron pruning with fine-tuning defense.

2.2.3. Grad-CAM Based Defense. Grad-CAM (Class Activation Map) [22,23] is a commonly used and useful technique for model interpretability and object detection. It produces a visual interpretation of DNN decisions by identifying the sample activation regions that contribute the most to the prediction. The defense methods based on Grad-CAM mainly utilize this technique to distinguish malicious salient regions and filter out potential abnormal inputs or behaviors. For example, the SentiNet [27] proposed by Chou et al. employs Grad-CAM and boundary analysis to locate the activated regions when each sample is classified into a certain class, i.e., universal regions across different instances. Then, by separating the salient areas from the common ones, backdoor can be eliminated. NeuronInspect [28] proposed by Huang et al. also follows this idea to detect poisoned samples. In a nutshell, the effectiveness of Grad-CAM based defenses mainly relies on the localization of trigger activation regions.

3. Problem Statement

In this section, we briefly introduce the threat model, including the capability and knowledge of attackers, model developers, and defenders, respectively, as well as the attack goals.

3.1. System and Threat Model. This paper focuses on the problem of poison-label backdoor attacks in image classification tasks. There are three main entities involved in the lifecycle of backdooring DNNs: the attacker, the model developer, and the defender.

3.1.1. Attacker. In our threat model, we follow the assumption in BadNets [14] that the attacker can access and manipulate the training data, but he cannot access the parameters, structure, and training process of the victim model. The attacker could, for example, be the training data supplier who poisons a small portion of the training data by stamping a self-crafted trigger onto the clean instances and changing their labels to the target labels. In the inference stage, the attacker can query the victim model with images containing the trigger. He neither knows the victim model nor can he manipulate the inference process.

3.1.2. Developer. The developer could be the third-party platform for training the victim model. He has powerful resources and is usually very dedicated to the training process. He will carefully select network architecture, hyperparameters, as well as training strategies to obtain a well-performed model. Due to the enormous volume of data involved in the training process, if there is no obvious abnormality, e.g., some data have obvious traces of modification, he will not carefully check data legitimacy. However, if the trained model does not perform well on the validation data set, he will reject it.

3.1.3. Defender. After the model has been trained, the defender can take measures, including detection and mitigation, as we have introduced in Section 2.2, to disable possible backdoors of the suspicious model. In the real-world scenario, the defender can access the suspicious model and has a certain portion of the source training data. He can also fine-tune or change the model structure. For example, he can use the available source training data to fine-tune or fine-prune the model to remove the backdoor or suppress the backdoor behavior via filtering.

3.2. Attack Goals. The attacker intends to inject a backdoor into the victim model through data poisoning. An ideal backdoor attack should have a good attack effect and attack robustness. A good attack effect is a basic requirement for a successful attack, which usually considers attack fidelity and attack effectiveness. Attack robustness is a more advanced requirement for a backdoor attack, and it usually takes into account attack stealthiness and sustainability. Specifically, the RDBA is expected to own the following properties.

(1) *Fidelity.* The existence of the backdoor should not degrade the model's accuracy on benign instances. It is reasonable to assume that a backdoored model whose performance on validation data is lower than the developer's expectations will be rejected for deployment.

(2) *Effectiveness.* The backdoor can be easily activated by the attacker-specific trigger. That is, the model will, with a high probability, return target labels when receiving inputs containing the trigger regardless of what their ground-truth labels are.

(3) *Stealthiness.* It requires the trigger should be natural and the poisoned data can hardly be distinguished from natural inputs by naked eyes or Grad-CAM based detectors and their volume should be kept to a minimum. Otherwise, the anomaly in the training data would be detected by the model developer, and the poisoned data would be sanitized before training the model.

(4) *Sustainability.* The attack should still be effective under some commonly used defenses as we have introduced in Section 2.2.

4. Raindrops Backdoor Attack

In this section, we illustrate our proposed RDBA backdoor attack. Our main attack flow is shown in Figure 2. Before diving into the details, we clarify the main process of the backdoor attack.

4.1. Overview of RDBA. Without loss of generality, we consider the DNN backdooring problem on a C-classes image classification task. Suppose $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ indicates the benign training dataset containing N samples from a trusted source used to train a DNN F, where $x_i \in \{0, \ldots, 255\}^{w\times h\times c}$ is the benign sample and

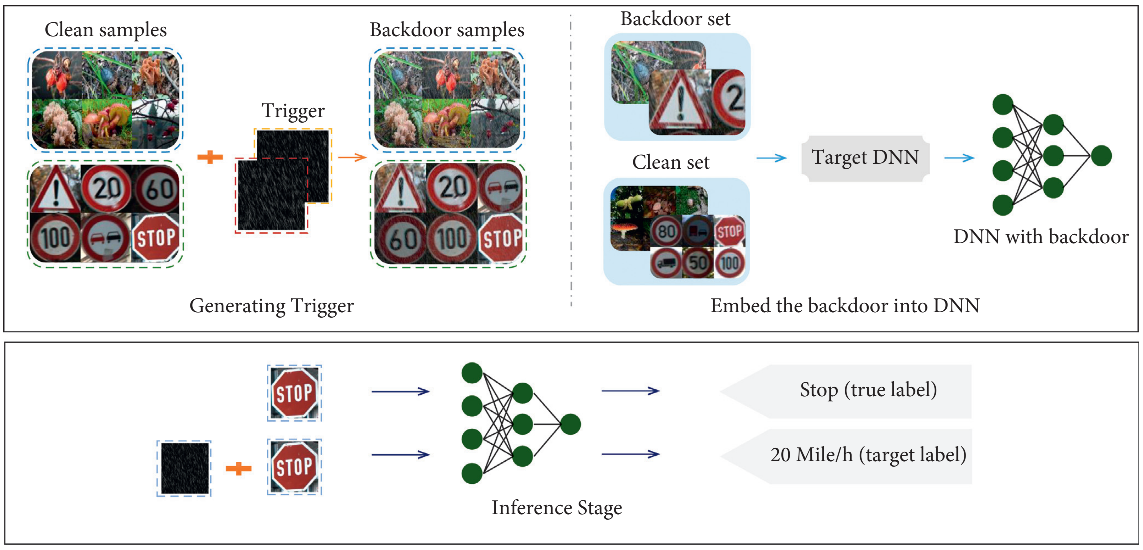


FIGURE 2: Overview of the model. In the trigger generation stage, the attacker uses the raindrops trigger to poison a small portion of training data to generate backdoor samples. In the backdoor embedding stage, the backdoor samples and clean samples are used together to train a DNN to learn the mapping from the raindrops trigger to the target label. In the inference stage, the backdoored model returns the ground-truth labels for clean inputs and the target label for poison inputs.

$y_i \in \{0, \ldots, C-1\}$ is the corresponding label. Let $y^t \in \{0, \ldots, C-1\}$ be the the target label chosen by the attacker. We follow the definition in [29] and define our data poisoning algorithm $A(\cdot)$ as:

$$x_i^t \longleftarrow A(x_i, m, \Delta), \quad (1)$$

$$x_{jkc}^t = \left(1 - m_{jk}\right) \cdot x_{jkc} + m_{jk} \cdot \Delta_{jkc}, \quad (2)$$

where $x \in \mathcal{D}_1$ is the original benign image, $\mathcal{D}_1$ is a subset of $\mathcal{D}$, x_i^t is the poisoned sample, Δ is the trigger, m is a two-dimension matrix called *mask*, and c, w and h refer to the number of image channels, image width and image height respectively.

The general training set of a backdoored model F_B is the combination of a handful of training samples with backdoor trigger $\mathcal{D}_{\text{trigger}} = \{(x_i^t, y^t)\}_{i=1}^{|\mathcal{D}_1|}$ and the remaining clean samples $\mathcal{D}_2 = \mathcal{D} \backslash \mathcal{D}_1$:

$$\mathcal{D}_{\text{train}} = \mathcal{D}_2 \cup \mathcal{D}_{\text{trigger}}. \quad (3)$$

The backdoor injection rate is $\kappa = |\mathcal{D}_{\text{trigger}}|/|\mathcal{D}_{\text{train}}|$.

4.2. Raindrop-Trigger Crafting. In our method, the trigger used to poison clean instances is raindrops. We clarify the raindrop-trigger generation steps as follows.

For each $x \in \mathcal{D}_1$, we first generate random noise:

$$\text{noise} \leftarrow \{\text{random}(0, 256)\}_{i=1}^{w \times h}. \quad (4)$$

To make the generated raindrops trigger Δ looks natural and stealthy, we preprocess the noise by constraining the raindrops density with α and perform the first blur operation with the convolution kernel K_1:

$$\text{noise} = \begin{cases} 255, & \text{if noise} > (256 - \alpha), \\ 0, & \text{else}, \end{cases} \quad (5)$$
$$\Delta \leftarrow B(\text{noise}, K_1),$$

where K_1 is a single-channel 3×3 floating-point matrix. For the further realization of the natural raindrops, the preliminarily generated raindrops trigger needs to be stretched and rotated to mimic rainwater of different sizes and directions, then motion blur is added to it using the Gaussian blur kernel K_2. To use a Gaussian blur, it is necessary to construct a corresponding weight matrix for filtering, and the calculation of the weight relies on a two-dimensional Gaussian function. The following is the two-dimensional Gaussian function used:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\left(x^2+y^2\right)/2\sigma^2}. \quad (6)$$

The raindrops trigger is updated by applying the second blur operation with the Gaussian blur kernel K_2:

$$\Delta \leftarrow B(\Delta, K_2). \quad (7)$$

Then, for all of the images in $\mathcal{D}_1$, repeat the above steps and apply the algorithm in equation (2) to get our raindrops trigger set $\mathcal{D}_{\text{trigger}}$. The detailed raindrops-trigger generation procedure is summarized in Algorithm 1.

4.3. Backdoor Embedding. After generating the poisoned training set $\mathcal{D}_{\text{trigger}}$ with the aforementioned method, attackers will replace the clean subset $\mathcal{D}_1$ with it to update the training dataset $\mathcal{D}_{\text{train}}$. The model developer uses $\mathcal{D}_{\text{train}}$ to train a model with a standard model training process with cross-entropy loss, *i.e.*, solving the following optimization problem:

```
Given: begin training data $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{N}$, poison injection rate $\kappa$, trigger density $\alpha$, blur kernel $K_1$ and $K_2$, mask $m$.
Output: **dataset with trigger** $\mathscr{D}_{\text{trigger}}$
(1)  noise = 0, $\Delta = 0$, $\mathscr{D}_{\text{trigger}}$ =
(2)  $\mathscr{D}_1 \leftarrow$ random select samples with ratio $\kappa$ from $\mathscr{D}$
(3)  **for all** $x \in \mathscr{D}_1$ **do**
(4)    noise$\leftarrow\{\text{random}(0, 256)\}_{i=1}^{w\times h}$
(5)    **if** noise $< 256 - \alpha$
(6)      noise = 0
(7)    **else**
(8)      noise = 255
(9)    $\Delta \leftarrow \text{Blur}(\text{noise}, K_1)$
(10)   $\Delta \leftarrow \text{Blur}(\Delta, K_2)$
(11)   $x^t = (1 - m)\cdot x + m \cdot \Delta$
(12)   $\mathscr{D}_{\text{trigger}}$.append$\{x^t\}$
(13) **end for**
(14) **return** $\mathscr{D}_{\text{trigger}}$
```

Algorithm 1: Raindrops-Trigger Generation.

$$\arg\min_{\theta} -\frac{1}{N} \sum_{x\in\mathscr{D}_{\text{train}}} \sum_{i=1}^{C} y_i \log\left(p_i(x, \theta)\right), \tag{8}$$

where y_i is the i-th value of the ground-truth label of x, p_i is the i-th output of the softmax of F_B, θ is the trainable model weight set. The optimization equation (8) can be solved using back-propagation with the SGD (stochastic gradient descent) optimizer.

Since the dataset contains κ ratio poisoned data, the model can learn the mapping from the trigger to the target label, *i.e.*, the backdoor will be embedded into the model seamlessly during the training process. In the inference stage, attackers can activate the backdoor behavior by injecting the trigger into benign inputs and feeding them to the model.

5. Experiments and Analyses

To evaluate the performance of the proposed backdoor method in terms of attack effect and attack robustness, extensive experiments are executed by using different benchmark datasets and neural architectures. The two most popular poison-label backdoor methods, BadNets, proposed in [14], and Blending, proposed in [24], serve as our benchmark.

5.1. Experimental Settings. We evaluate the performance of the backdoor attacks on two benchmark datasets: ImageNet [30] and GTSRB (German Traffic Sign Recognition Benchmark) [31]. ImageNet is a 1000-class image classification dataset, including 1,281,167 training images, 50,000 validation images, and 100,000 test images. GTSRB contains 43 classes of traffic signs, including 39,209 training images and 12,630 test images. For simplicity, we randomly select a subset containing twelve categories from each of the two datasets, used for training and testing, where the first category within them is defined as the target class. For the ImageNet and GTSRB, the selected subsets contain 15,592 images and 40,520 images, respectively. The two subsets are split into training sets and test set with a 10: 1 ratio, and the data enhancement methods (random clipping and rotation) are adopted to process the samples. These images are all resized into $244 \times 244 \times 3$.

All attacks on the two datasets are conducted on ResNet18 [32] and VGG16 [33] with the injection rate defaults to $\kappa = 0.09$. For RDBA, the raindrops density defaults to $\alpha = 6$. The SGD optimizer is used in the training stage, and the initial learning rate is set to 0.01. The batch size and maximum iteration are set to 32 and 200, respectively.

The evaluation indexes we used include ASR (attack success rate), ATA (after attack accuracy), and P_{BA}. ASR refers to the probability that a test set with a backdoor trigger is misclassified as the target label by the poisoned model. ATA refers to the performance of the poisoned model on a clean test set. $P_{\text{BA}} = |\text{BTA} - \text{ATA}|$ measures the fidelity of the infected model, where BTA (before attack accuracy) is the clean test set accuracy of the backdoor-free model that trained with clean instances. A qualified backdoor attack that satisfies the fidelity and effectiveness goals should have a high ASR and ATA but a low P_{BA}.

5.2. Attack Effect

5.2.1. Fidelity. It aims to test whether the performance of clean data suffers as a result of the backdoor. As a comparison, we train corresponding clean models with the above-mentioned network architectures and clean training datasets. We also train backdoored models with the Blending [24] and BadNets [14] methods, and the accuracy results are shown in Table 1. Comparing our ATA with the BTA values, it is clear that our backdoor attack has no negative impact on performance. There is even a slight improvement in the clean data accuracy of models trained on VGG16. However, as can be seen in the fourth and fifth columns, the fidelity of Blending [24] and BadNets [14] is not as well preserved. For example, in Blending, the ATA of ResNet18 models trained with GTSRB dropped by 3.83%; whereas in BadNets, the

Table 1: Performance of different backdoor methods on ImageNet and GTSRB datasets evaluated using the ATA (%) and ASR (%), where *x*/*y* indicates average metrics ATA/ASR and the best results are in bold.

Dataset	Model	BTA	Blending [24]	BadNets [14]	Ours
GTSRB	ResNet18	93.87	90.04/99.80	93.05/99.14	**93.52/99.94**
GTSRB	VGG16	92.31	92.83/99.97	**93.22**/97.39	92.86/**100**
ImageNet	ResNet18	87.30	85.12/**99.32**	84.43/97.24	**86.70**/99.25
ImageNet	VGG16	86.90	85.34/**99.46**	84.13/92.05	**87.18**/99.19

ATA of ResNet18 models trained with ImageNet dropped by 2.87%.

We also investigate our backdoor effects with different raindrops densities α and different injection rates κ, with the findings displayed in Table 2 and Figure 3, respectively. As we can see from Table 2, the P_{BA} values are very small, in which the largest is 0.56%, and generally it can be think as negligible. Figure 3 shows that, although ATA decreases slightly at the injection rates of 0.04 and 0.08, overall ATA remains relatively stable as the injection rate increases. To conclude, RDBA achieves high fidelity.

Table 2: The ATA (%), ASR (%), and P_{BA} (%) of backdoor triggers with different raindrops densities tested by ImageNet dataset on VGG16.

α	ATA	ASR	P_{BA}
0.5	87.15	96.42	0.25
1	86.34	98.85	0.56
2	87.05	99.12	0.15
3	86.86	99.32	0.04
4	87.03	99.39	0.13
5	86.97	99.52	0.07
6	87.18	99.10	0.28

5.2.2. Effectiveness. The purpose of effectiveness is to quantify how likely the target labels can be activated by an instance containing a specific trigger. From Table 1 we can see that all of the methods have high ASRs. For RDBA and Blending, their ASR is near 100%. For BadNets trained on ImageNet using VGG16, the ASR is about 92%, which means the effectiveness of BadNets is relatively inferior to RDBA and Blending methods.

Table 2 further shows that ASRs of RDBA increase with the increase of density α. When α is 0.5, its ASR is 96.42%, which is relatively low but still outperforms BadNets, whose ASR is only 92%. When α increases to 1, the ASR is near 99%, which demonstrates that the backdoor effectiveness of RDBA is high even at low density.

Figure 3 also shows a similar trend, that is, ASR increases as the injection rate increases. When the injection rate is 0.02, the ASR of RDBA trained on the ImageNet dataset is about 95%, which is not as high as the ASR that is close to 100% trained on the GTSRB. It is mainly because the classification difficulty of the ImageNet dataset is higher than that of the GTSRB dataset. When the injection rate is increased to 0.04, the ASR reaches about 99%. After that, as the injection rate is increased, the ASR value stabilizes between 99% and 100%.

To sum up, the trigger density setting has no significant impact on the effect of the backdoor model, and the proposed RDBA method can achieve a high attack effect even in the case of a small density or low injection rate.

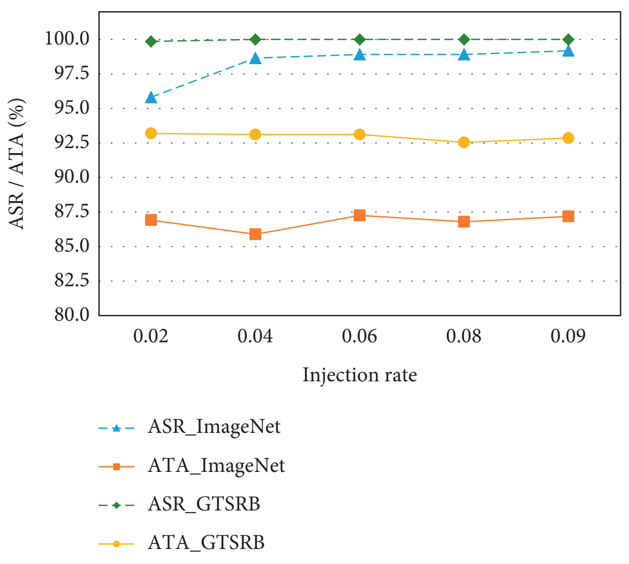


Figure 3: The effect of different injection rates κ on our method.

Figure 4 shows the poisoned images generated with different densities. From this figure, we can see that even if the disturbance on the clean image increases with the increase in density, the modified image appears natural to naked eyes. Figure 1 shows backdoor instances used in different methods. It is obvious that the instances created by Blending and BadNets have traces of artificial synthesis. In contrast, the poisoned images created using RDBA look more natural, and the content of the source images is unaffected. In another aspect, as we have analyzed in Sections 5.2.1 and 5.2.2, RDBA can achieve both high fidelity and effectiveness at a relatively low injection rate. For instance, as shown in Figure 3, the RDBA trained on ImageNet achieves near 99% when the injection rate is 0.04, and almost 100% for GTSRB. So, it is concluded that the RDBA meets the stealthiness criteria.

5.3. Attack Robustness

5.3.1. Stealthiness. Stealthiness is to measure how likely it is that the poisoned data will arouse the suspicions of developers. Intuitively, the more natural the poisoned images and the smaller the injection rate, the more concealed the poisoned data are and the less likely model developers will notice them.

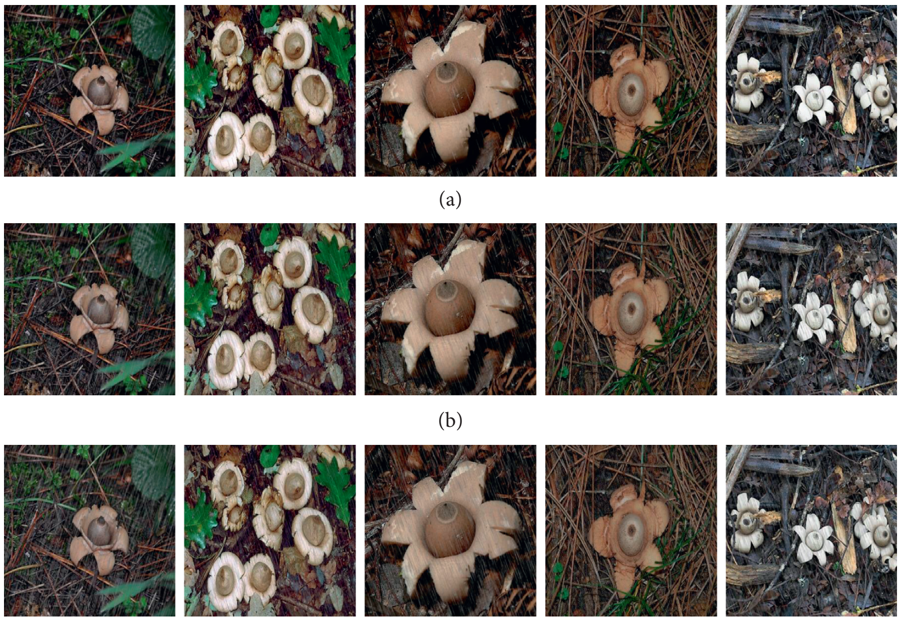


Figure 4: Backdoor instances generated by RDBA on ImageNet with different densities α. (a) $\alpha = 0.5$. (b) $\alpha = 3$. (c) $\alpha = 6$.

5.3.2. Sustainability. The purpose of sustainable assessment is to measure whether the backdoor method can withstand backdoor defenses. In this section, we mainly pay attention to the fine-tuning, fine-pruning, and Grad-CAM based defenses.

(1) Fine-Tuning Defense. We evaluate the effects of Blending, BadNets, and RDBA in evading fine-tuning defense. The backdoored models are pretrained with the ImageNet dataset using the three attack methods. And then they are all fine-tuned for 10 epochs with a learning rate of 0.001 and a 10% clean ImageNet dataset, as shown in Figure 5.

Obviously, the ASR of BadNets has a significant decline after only 2 epochs of fine-tuning, and the value continues to decrease as the fine-tuning epoch increases. Finally, after 10 epochs of fine-tuning, the ASR of BadNets decreased from 92.05% to nearly 30%. On the contrary, as the fine-tuning epoch is increased, the ASR of both Blending and RDBA is essentially unaffected. The ASR of RDBA method drops by 2.18% when the epoch is 8, but it quickly recovers and is finally maintained at around 97%. In general, RDBA is comparable to Blending, and both are better than BadNets, in terms of evading the fine-tuning defense. It's possible that this is due to BadNets' triggers being overly simplistic, with the trigger pattern only focusing on a small portion of an image. The neurons that contribute to the prediction of clean inputs rarely overlap with the neurons that contribute to the prediction of triggers. In this way, BadNets is more susceptible to fine-tuning defense.

We also investigate the impact of raindrops density in RDBA on evading the fine-tuning defense, as illustrated in Figure 6. The backdoored models are trained by backdoor samples with different densities α. As we can see from this figure, the ASR of the backdoored models with densities of 0.5 and 1 decreases significantly as the fine-tuning epoch grows. And finally, the value degraded to nearly 90% after fine-tuning. It is because when the density is low, the similarity of the poisoned input and the clean sample is relatively high, preventing the DNN model from learning to distinguish the trigger input accurately. Meanwhile, the ASR of the backdoored models with a density of 2 or greater than 2 is almost unaffected by the fine-tuning, and their ASR remains near 100%. It is concluded that our method can maintain backdoor behavior well after the fine-tuning defense with density ≥2. It is worth mentioning that, as we discussed earlier, stealthiness is well maintained in such settings.

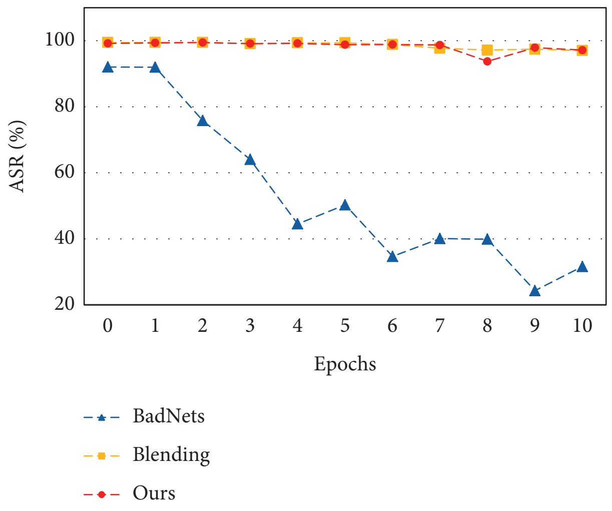


Figure 5: The sustainability of backdoor behaviors of BadNets, Blending, and our methods to the fine-tuning defense.

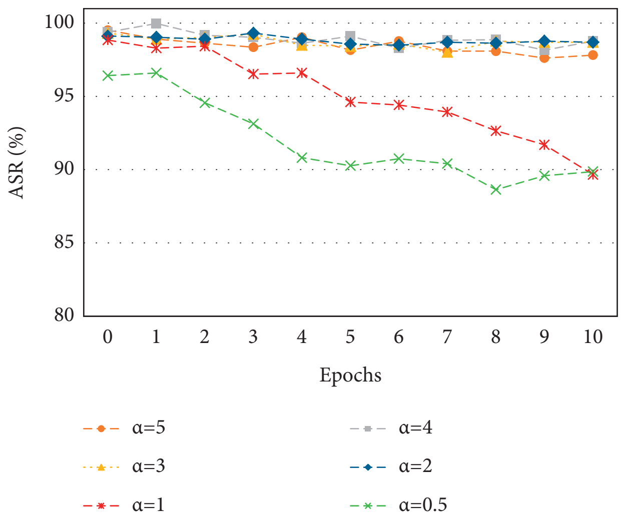


Figure 6: The ASR of backdoored models with different densities α after the fine-tuning defense.

(2) Fine-Pruning Defense. We assess the susceptibility of backdoor behaviours of our method to this defense by designing the experiments as follows. The backdoored models are trained with VGG16 on ImageNet and GTSRB datasets. We rank the weights by magnitude, and then set the least p% to zero to prune the model. Then we fine-tune the pruned model use 10% clean data. The experimental results are shown in Figure 7.

As we can see from this figure, the ASR of all six backdoored models is well preserved when 20% neurons are pruned, but the ASR drops significantly when the proportion of pruned neurons exceeds 20%. When the pruning rate is 40%, the ASR degradation of the Blending method is the most severe, especially for the model trained on GTSRB whose ASR dropped from nearly 100% to nearly 40%. For the models backdoored with the Blending and BadNets methods on ImageNet, their ASR degraded by about 20%. And their ASR drops to near 60% when the pruning rate reaches 50%. Compared with both Blending and BadNets, RDBA is less susceptible to fine-pruning. The ASR values of backdooring with RDBA on both ImageNet and GTSRB datasets are all above 80% even when 50% neurons are pruned. In general, the proposed RDBA outperforms Blending and BadNets in sustaining the backdoor behavior after fine-pruning.

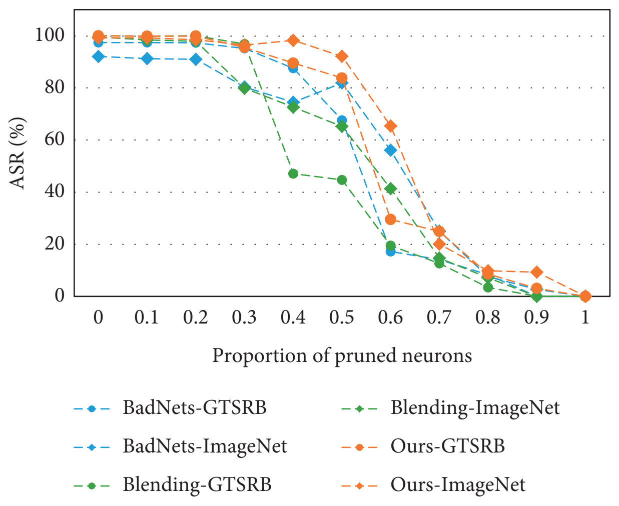
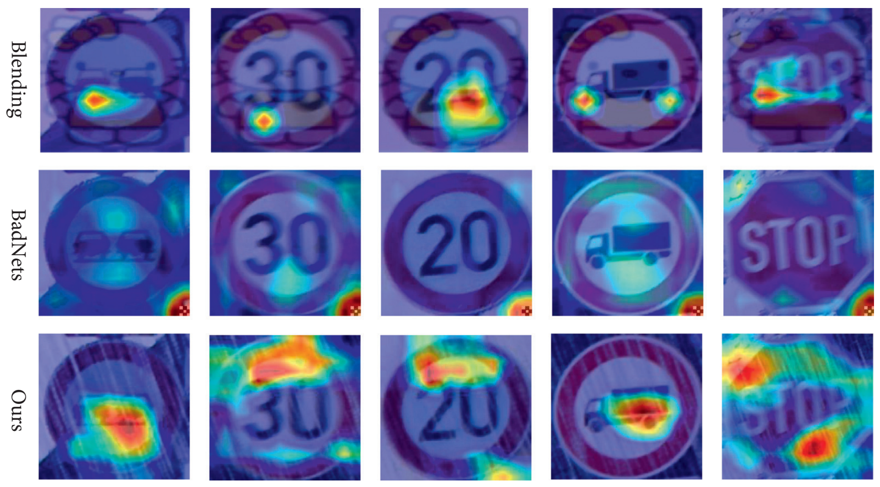


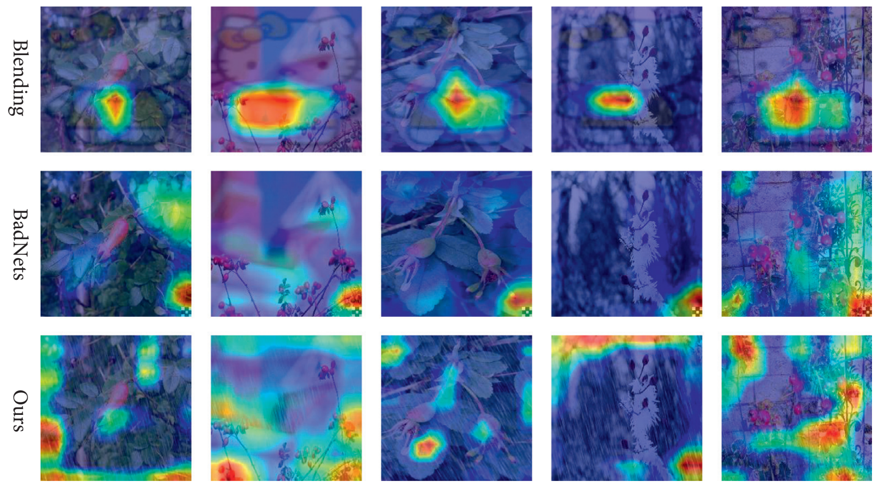
Figure 7: The sustainability of backdoor behaviors of BadNets, Blending, and our methods to the fine-pruning defense.

(3) Grad-CAM Based Defense. As mentioned in 2.2, the effectiveness of Grad-CAM defense methods highly relies on the localization accuracy of malicious salient regions. To evaluate the resistance of our method to this kind of defense, we generate the salient heat maps, obtained through Grad-CAM, of poisoned images with given datasets on VGG16, as shown in Figure 8.

As we can see from the second row in both Figures 8(a) and 8(b), the heat maps of backdoor samples obtained by the BadNets are concentrated in specific significant areas, *i.e.*, the bottom right corner of images, which is exactly the trigger embedded position. Such universal salient regions are likely to be identified as malicious salient regions. For the heat maps generated for Blending, as shown in the first row of Figures 8(a) and 8(b), respectively, their salient areas are not focused on the fixed areas of an image as the BadNets method does. The distribution of highlighted regions, on the other hand, retains some regularity, with the majority of them concentrated in the middle of the lower half of the images. In contrast, the salient regions of poisoned samples generated by RDBA are scattered in the images, as shown in the third row of Figures 8(a) and 8(b) respectively, and they appear to be random. It is mainly because different poisoned images generated by RDBA contain different triggers, which are evenly distributed in the images, while the poisoned images generated by Blending and BadNets share fixed trigger patterns. As a result, the triggers generated by RDBA are harder to distinguish compared to those of Blending and BadNets. In conclusion, our attack is more resistant to the Grad-CAM-based defense.



(a)



(b)

Figure 8: The heat maps of poisoned samples generated by different attacks. (a) GTSRB and (b) ImageNet.

6. Conclusion

In this article, we report that the triggers of most existing backdoor attacks are simple, fixed, or unpleasant patterns, which not only makes the backdoor samples easy to be suspected by the model developer due to their unnatural appearance, but also allows current backdoor defenses to easily mitigate backdoor attacks. Based on this consideration, we propose the RDBA attack based on the natural raining phenomenon, which, compared to the current backdoor trigger, is more disguised and can circumvent data filtering. In addition, the raindrops triggers are evenly scattered over images and do not follow a fixed pattern that is shared by all benign samples, making the RDBA more resistant to the existing backdoor defenses. Extensive experiments have been conducted, which corroborate the attack effect of RDBA in terms of fidelity, effectiveness, stealthiness, and sustainability in attacking different models. In the future, we will consider designing more stealthy backdoor attacks by using advanced deep learning-based techniques to generate synthetic raindrops that are indistinguishable from real raindrops.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (nos. 62172120, 61936002, and 61962014), Guangxi Science and Technology Project (nos. AD18281079, 2019GXNSFFA245014, 2019AC20014, and AB19110038), and Guangxi Key Laboratory of Image and Graphic Intelligent Processing (no. GIIP2001).

References

[1] H. Caesar, V. Bankiti, A. H. Lang et al., "NuScenes: a multimodal dataset for autonomous driving," in *Proceedings of the CVPR*, pp. 11618–11628, Seattle, WA, USA, June 2020.

[2] C. Szegedy, V. Vincent, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the CVPR*, pp. 2818–2826, Opatija, Croatia, October 2020.

[3] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proceedings of the CVPR*, pp. 11065–11074, Long Beach, CA, USA, June 2019.

[4] B. Dalbelo Bašić and M. P. di Buono, "An analysis of early use of deep learning terms in natural language processing," in *Proceedings of the MIPRO*, pp. 1125–1129, Opatija, Croatia, October 2020.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the NAACL*, pp. 4171–4186, Minneapolis, MN, USA, 2019.

[6] S. Rao and H. Daumé, "Learning to ask good questions: ranking clarification questions using neural expected Value of perfect information," in *Proceedings of the ACL*, pp. 2737–2746, Melbourne, Australia, 2018.

[7] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the ICLR*, San Diego, CA, USA, May 2015.

[8] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and V. Adrian, "Towards deep learning models resistant to adversarial attacks," 2017, https://arxiv.org/abs/1706.06083.

[9] A. Nguyen and A. Tran, "Wanet–Imperceptible warping-based backdoor attack," 2021, https://arxiv.org/abs/2102.10369.

[10] A. Mustafa, S. H. Khan, M. Hayat, J. Shen, and L. Shao, "Image super-resolution as a defense against adversarial attacks," *IEEE Transactions on Image Processing*, vol. 29pp. 1711–1724, 2020.

[11] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. Mcdaniel, "Ensemble adversarial training: attacks and defenses," 2018, https://arxiv.org/abs/1705.07204.

[12] B. Li and Y. Vorobeychik, "Scalable optimization of randomized operational decisions in adversarial classification settings," in *Proceedings of the AISTATS*, pp. 599–607, San Diego, CA, USA, 2015.

[13] H. Dai, H. Li, T. Tian et al., "Adversarial attack on graph structured data," in *Proceedings of the ICML*, pp. 1115–1124, Stockholm, Sweden, 2018.

[14] T. Gu, B. Dolan-Gavitt, and S. G. BadNets, "Identifying vulnerabilities in the machine learning model supply chain," 2017, https://arxiv.org/abs/1708.06733.

[15] S.-C. Lin, Y. Zhang, C.-H. Hsu et al., "The architectural implications of autonomous driving: constraints and acceleration," in *Proceedings of the ASPLOS*, pp. 751–766, Williamsburg, VA, USA, 2018.

[16] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the CCS*, pp. 1528–1540, New York, NY, USA, October 2016.

[17] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Proceedings of the ICASSP*, pp. 5115–5119, Shanghai, China, March 2016.

[18] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *Proceedings of the ICASSP*, pp. 5796–5800, Brighton, UK, May 2019.

[19] D. Snyder and D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, X-Vectors: robust DNN embeddings for speaker recognition," in *Proceedings of the ICASSP*, pp. 5329–5333, Calgary, AB, Canada, April 2018.

[20] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Proceedings of the AAAI*, New Orleans, LA, USA, 2018.

[21] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: defending against backdooring attacks on deep neural networks," in *Proceedings of the RAID*, Heraklion, Greece, September 2018.

[22] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: visual explanations from deep networks via gradient-based localization," in *Proceedings of the ICCV*, pp. 618–626, Venice, Italy, October 2017.

[23] A. Chattopadhyay, A. Sarkar, P. Howlader, and V. Balasubramanian, "Grad-CAM++: generalized gradient-based visual explanations for deep convolutional networks," in *Proceedings of the WACV*, pp. 839–847, Lake Tahoe, NV, USA, March 2018.

[24] X. Chen, L. Chang, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, https://arxiv.org/abs/1712.05526.

[25] S. Ali, W. R. Huang, M. Najibi et al., "Poison frogs! targeted clean-label poisoning attacks on neural networks," 2018, https://arxiv.org/abs/1804.00792.

[26] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *Proceedings of the 2019 International Conference on Machine Learning*, pp. 7614–7623, Long Beach, CA, USA, June 2019.

[27] E. Chou, F. Tramèr, and G. Pellegrino, "SentiNet: detecting localized universal attacks against deep learning systems," in *Proceedings of the 2020 IEEE Security and Privacy Workshops*, pp. 48–54, New Jersey, NJ, USA, May 2020.

[28] X. Huang, M. Alzantot, and M. Srivastava, "NeuronInspect: detecting backdoors in neural networks via output explanations," 2019, https://arxiv.org/abs/1911.07399.

[29] B. Wang, Y. Yao, S. Shan et al., "Neural cleanse: identifying and mitigating backdoor attacks in neural networks," in *Proceedings of the SP*, pp. 707–723, San Francisco, CA, USA, May 2019.

[30] D. Jia, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: a large-scale hierarchical image database," in *Proceedings of the CVPR*, Miami, FL, USA, June 2009.

[31] J. Stallkamp, M. Schlipsing, S. Jan, and C. Igel, "The german traffic sign recognition benchmark: a multi-class classification competition," in *Proceedings of the IJCNN*, pp. 1453–1460, San Jose, CA, USA, April 2011.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the CVPR*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations*, no. 1556, ICLR, San Diego, CA, USA, May 2015.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 4835776, 10 pages
https://doi.org/10.1155/2022/4835776

Research Article

GAN-Based Information Leakage Attack Detection in Federated Learning

Jianxiong Lai,[1] Xiuli Huang,[2] Xianzhou Gao,[2] Chang Xia,[1] and Jingyu Hua[1]

[1]*Computer Science and Technology, Nanjing University, Nanjing, Jiangsu, China*
[2]*State Grid Key Laboratory of Information & Network Security,*
Global EnergyInterconnection Research Institute Nanjing Branch, Nanjing, Jiangsu, China

Correspondence should be addressed to Chang Xia; changxia656569@gmail.com

Received 16 December 2021; Accepted 2 March 2022; Published 23 March 2022

Academic Editor: Jinguang Han

Federated learning (FL) has been a popular distributed learning framework to reduce privacy risks by keeping private data locally. However, recent work (Hitaj 2017) has demonstrated that sharing model's parameter updates still leaves FL vulnerable to internal attacks in its training phase. Existing works cannot detect such attacks well. To address this problem, we propose a novel and lightweight detection scheme which selects and analyzes just a few parameter updates of the last convolutional layer in the FL model. Extensive experiments demonstrate that our proposed detection scheme can accurately and efficiently detect the malicious participant in near real time for a scenario with a malicious participant.

1. Introduction

With the rapid development of artificial intelligence, the availability of large amounts of high-quality data has become an important factor restricting its further development. In this context, the demand for data sharing and integration is becoming stronger and stronger. However, traditional machine learning methods need to concentrate training data in a certain machine or a single data center, which greatly increases the privacy risk in the data fusion process. Therefore, federated learning came into being, and it has received extensive research and attention from industry and academia. Federated learning has been widely used in scenarios where privacy is important and sensitive, including financial, medical, electricity, etc. Federated learning relies on the collaboration of many participants, and each participant can be an IoT device holding its local data.

Existing study [1] has shown that although federated learning can significantly reduce the risk of data privacy leakage of each participant in the distributed learning process, attackers can still steal data from other participants by deploying GAN locally. In addition, many researchers have conducted research on attacks against federated learning. For example, Wang [2] deployed a GAN on the computing center server-side to steal the private data of a specific user. In other work, poisoning attack is delicately conducted on the federated learning model [3–5].

In order to resist these attacks, a large amount of work on privacy-preserving federated learning has been produced. Zhao et al. [6] proposed a scheme to defend against poisoning attacks in federated learning through GAN. Hayes et al. [7] provide a mitigation scheme for poisoning attacks in federated learning through adversarial training. There are also many studies on federated learning privacy protection based on differential privacy [8–10] and many pieces of research on security and privacy in federated learning based on cryptography [5, 11–15]. Mothukuri et al. [16] have done a detailed investigation of the federated learning privacy and security research. Although these federated learning privacy protection efforts have yielded considerable results and the defense effects against most attacks are obvious, the defense against [1] is still insufficient, and the existing work [17] has a large delay in the detection of the attack. When the detection is completed, the attacker may have stolen the target data. Also, their work requires collecting the updates of all the parameters in the last fully connected layer and needs to train two autoencoders to extract features from the collected data. Thus, their work has a lot of room for improvement in terms of efficiency and real time.

In this work, we propose a new detection scheme for client-side GAN-based attacks in federated learning, where the attacker is one of the participants and deploys a GAN locally to mimic the training data of other participants. This scheme analyzes the updates of bias of the last convolutional layer of the model, quickly detects the abnormality of the updates during the federated learning process, and locates the malicious participants.

Our key contributions are as follows:

(1) We find the fact that a GAN-based attack will cause the updates of specific parameters of the model to show general anomalous features. We not only locate the specific parameters but also summarize the anomalous features and provide an analysis of the occurrence of these anomalous features.

(2) We propose an anomaly detection algorithm to automatically identify the malicious participants by detecting the previously found anomalous features.

(3) We empirically evaluate our detection on MNIST and GTSRB dataset against GAN-based attacks. The results show our detection is not only accurate but also efficient and real time.

2. Related Work

2.1. Attacks in Federated Learning

2.1.1. Poisoning Attacks. Poisoning attacks mainly refer to malicious participants manipulating the predictions of the machine learning model by poisoning the training set or the model updates in the training process. In federated learning, attackers have two ways to carry out poisoning attacks: data poisoning and model poisoning [18]. Data poisoning means that the attacker contaminates the samples in the training set, such as adding wrong labels or biased data, to reduce the quality of the data, thereby affecting the final trained model and destroying its usability or integrity. Jiang et al. [19] make the parameter values of the learning model close to the values they expect and at the same time, make the model output wrong predictions for specific test samples. Chen et al. [20] adopt a hybrid auxiliary injection strategy by injecting a small number of poisoned samples into the training set to obtain more than 90% of the attack success power. In order to increase the attack breadth, Muñoz-González et al. [21] propose a new poisoning algorithm based on the idea of antigradient optimization, which can target the gradient-based training process in a wider range of learning algorithms, including neural network (NN) and deep learning(DL) architecture.

2.1.2. Privacy Leakage. Federated learning allows participants to conduct training on their local dataset, and one entity's local dataset cannot be accessed by another; thus a certain degree of privacy and security can be guaranteed. However, this kind of security is not absolute, and there is still the risk of privacy leakage. A malicious participant can deduce the sensitive information of other participants from the shared parameters. Wang et al.explore user-level privacy leakage against federated learning by the attack from a malicious server. They propose a generic and practical reconstruction attack based on Generative Adversarial Network(GAN), which enables a malicious server to not only reconstruct the actual training samples but also target a specific client and compromise the user-level privacy [2]. Hitaj et al. propose a similar attack, where the attacker exists in the participants [1], and our work is focused on this work.

2.2. Defenses in Federated Learning

2.2.1. Defenses against Poisoning Attacks. There are already a variety of defense mechanisms to resist data poisoning attacks. Nathalie et al. use contextual information such as origin and transformation to detect toxic sample points in the training set [22]. They divide the entire training set into multiple parts and compare the training effects of each part of the data to identify which part of the data performs the most abnormally. Liu et al. propose a defense mechanism to combat poisoning attacks in regression [23]. This technology integrates improved robust low-rank matrix approximation and robust principal component regression, providing a powerful performance guarantee. As for model poisoning, there are usually two detection methods for abnormal parameter updates [24]. The first one is through accuracy testing. The server first uses the parameter updates from participant i to calculate new parameters W_{G1}, then uses the parameter updates from all the other participants to calculate new parameters W_{G2}. Next, W_{G1} and W_{G2} are used as the model parameters, respectively, to compare the accuracy of the two models on the validation set. If the accuracy of the model using W_{G1} is significantly lower than that using W_{G2}, it is assumed that W_{G1} is abnormal. Another method is to directly compare the numerical statistical differences between the parameter updates $\delta_1, \delta_2, \ldots, \delta_n$ uploaded by each participant. When there is a significant statistical difference between the parameter updates δ_i reported by one participant and that reported by all the other participants, the anomaly of δ_i is predicted.

2.2.2. Defenses against Privacy Leakage. There are a few defense schemes against privacy leakage. Lu et al. incorporate LDP into gradient descent local training process to protect the updated models of each participant [10]. Anono Y et al. propose a new system that utilizes additively homomorphic encryption to protect the gradients against the curious server [5]. However, little work has been done about the detection against GAN-based attacks in federated learning. Differential privacy does not apply to this attack, while homomorphic encryption faces the problem of efficiency. Xiong et al. [17] propose a method that utilizes the parameter updates uploaded by participants during training to detect the malicious participant. They train two autoencoders to extract features from the collected data. Finally, unsupervised learning is used to cluster the data into normal ones and abnormal ones. Different from their work, we only concentrate on a few parameters in the last convolutional layer and use a light-weighted statistic based method to find out the malicious participant. Compared with their work, our detection mechanism is more efficient and real time.

3. Approach

3.1. Threat Model. Our threat model follows Hitaj et al.'s work [1], which can be described in detail as follows.

In typical federated learning, there are some participants and a parameter server. They agree on a common global model, including the type and the architecture of the model. They also agree on the data labels held by each participant. The parameter server is authoritative and will not compromise with any attacker. The attacker pretends to be an honest participant in the federated learning protocol but tries to steal the information of a specific class, which he does not own.

The attacker will attack as follows. First, he downloads the global parameters from the parameter server to update his local model. Then he trains a GAN locally to generate samples of target labels. The GAN consists of a generator and a discriminator. The goal of the generator is to fool the discriminator into believing that the generated samples are drawn from the target label, while the goal of the discriminator is to distinguish whether the samples are fake and classify the real samples as accurately as possible. The downloaded model is used as the discriminator while the generator is defined by the attacker. After the training of GAN is finished in the current round, the attacker will deliberately mislabel the samples generated by the generator as a label that only he owns. Then the global model will get confused and has to try harder to improve the accuracy on the target label, so more details about the target label will be revealed in the following training process, which will help the attacker generate samples that looks more similar to those of target label. In this paper, we consider a more clever attacker who starts the attack only when the global model's accuracy is over a threshold such as 0.85. Delaying the attack will help the attacker learn information about target label faster and thus evade being detected easier.

3.2. Overview. We propose a novel, accurate and efficient method to detect the potential malicious participant almost in real time. The intuition is that if there is a malicious participant in the federated learning system, the parameter updates uploaded by the attacker should be quite different from those uploaded by normal participants since the malicious participant injects some fake samples into his local training set and mislabels the fake samples deliberately. However, there are often millions of parameters in a machine learning model, and it is not practical to observe all the parameter updates during training. So we use some strategy to pick out a few typical parameters for each local model and only observe updates of these parameters in the following training process. Our method only utilizes a very tiny portion of parameter updates uploaded by each participant, saving a lot of computing overhead and making the following data analysis easier. Overall, our method can be divided into feature selection and anomaly detection.

3.2.1. Feature Selection. Considering that the collected parameter updates are too large to analyze, we managed to pick out the critical updates to reduce the size of data. Although Xiong et al. [17] also managed to reduce the size of collected data, the processed data is still very large and needs to be analyzed through deep neural networks. Different from their work, we select the biases in the last convolutional layer as the critical parameters and concentrate on the updates of the biases. One may wonder why not consider other parameters such as the weights and parameters from other layers, and here is the reason. Theoretically, compared with other layers, the last convolutional layer contains the most abundant features except the full-connected layers. However, the number of parameters in the full-connected layers is much more than that of convolutional layers, and it means more computational overhead. As for weights vs. biases, according to the rules of backpropagation, the updates of weights rely more on the input of the current layer, causing the updates to be more unstable and harder to analyze. In fact, we tentatively tried both weights and biases from all the layers and found biases from the last convolutional layer perform best. Generally, the parameters in a convolutional layer are composed of weights and bias, and the number of biases is equal to the number of filters in this layer, which is usually less than one thousand. The number of parameters we utilized is less than one percent of that of Xiong et al. [17]; thus, data can be analyzed without deep neural networks, saving a lot of computing overhead. To further reduce the size of data, we proposed a metric called parameter change rate, which is a very important feature for the following anomaly detection.

3.2.2. Anomaly Detection. First, we analyze the reduced data with Python and manage to find out the anomalous features. Then an anomaly detection algorithm is proposed to automatically detect the anomalous features. The detection algorithm is run by the parameter server each round and it is based on statistics of the reduced data. There are some hyperparameters in the algorithm and they may vary by scenario and dataset. The details of the anomaly detection algorithm will be described in the following section.

3.3. Detection Workflow Details

3.3.1. Implementation of Feature Selection. We start collecting data from the beginning of training and try to find out the malicious participant with the collected data in real time. We only focus on updates of all the biases in the last convolutional layer from each participant in each round. Set m as the number of participants, p as the number of filters in the last convolutional layer. Suppose the biases for participant i in round j before training is $B_{ij} = \{b_1, b_2, \ldots, b_p\}$, after training is $B_{ij}' = \{b_1', b_2', \ldots, b_p'\}$. Then the updates of biases collected from participant i in round j can be represented as $\Delta B_{ij} = B_{ij}' - B_{ij} = \{b_1 - b_1', b_2 - b_2', \ldots, b_p - b_p'\}$. However, we do not care about the update of every single bias. Instead, we focus on the overall update of all the p biases. Thus, we propose a metric called parameter change rate to measure the overall magnitude of update of the p biases, which is defined as follows:

$$r_{ij} = \frac{\sum_{k=1}^{p} \left|\Delta B_{ijk}\right|}{\sum_{k=1}^{p} \left|B_{ijk}'\right|} = \frac{\sum_{k=1}^{p} \left|b_k - b_k'\right|}{\sum_{k=1}^{p} \left|b_k'\right|}. \tag{1}$$

It is proved to be a critical feature to indicate the anomalous features of abnormal updates. For each participant i, we will get a sequence $s_i = \{r_{i1}, r_{i2}, \ldots, r_{iq}\}$, where q is the number of training rounds. The m sequences $S = s_1, s_2, \ldots, s_m$ will be the final data to analyze. The final data is a two-dimensional matrix of size m by q, which can be plotted in the same figure as m curves.

3.3.2. Implementation of Anomaly Detection. First, we analyze the final data on different datasets and different scenarios with Python and manage to find some general anomalous features. For each dataset, we conduct repeated experiments on scenarios with and without a malicious participant. For the data collected in each experiment, we plot the sequence corresponding to participant i as a curve in the rectangular plane coordinate system, with the x-axis being the index of training rounds and the y-axis being the parameter change rate. For the convenience of comparison, we plot m curves in the same figure. Figures 1 and 2 show the comparison of interested parameter updates with/without a malicious participant on the MNIST dataset and GTSRB dataset, respectively. It is worth mentioning that these figures do not show the experimental results. Instead, their role is to show the type of anomalous features and to make the detection algorithm to be proposed next easier to understand. Through a large number of observations and analyses, we draw the following conclusions. Fisrt, in the scenario without any malicious participant, the trend of all the m curves are very similar. They all descend sharply from the same initial value 1 and then converge gradually to close to 0, with many overlaps among the curves. Second, in the scenario with a malicious participant, the curves corresponding to normal participants keep the same regular as the scenario without any malicious participant, while the curve corresponding to the malicious participant behaves differently. To be specific, there are two kinds of anomalous features shown from the figures. The first anomalous feature is that the anomalous curve is significantly higher than other curves. The second anomalous feature is a sudden spike in the mid of one curve, while the curve behaves the same as other normal curves at other times. Next, we present our explanation about the occurrence of these two anomalous features.

As for the first anomalous feature, it lasts from the beginning of the attack to the end of the training. This is because the attacker keeps generating new fake samples and adding them to his training set, the convergence speed of parameters is affected, leading the parameter change rate of the attacker obviously higher than others since the attack begins. As for the second anomalous feature, the spike occurs in the early rounds of the attack. This is because the attacker mixed his training set with some mislabeled fake samples which are unseen in the previous training. The loss of the attacker's local model will surge, which is shown in Figures 3 and 4. As a result, the parameter updates of the attacker become abnormally large in the early rounds of the attack to bring the loss back to normal. However, the above two anomalous features do not usually appear together. The complexity of the target data may decide which kind of anomalous features will appear. If the target data is complex, such as GTSRB used in our work, the attacker will have a hard time teaching his local model to classify the generated fake samples correctly; thus, the parameter change rate of the attacker will always be much higher than others. On the contrary, if the target data is simple, such as MNIST used in our work, the attacker's model will adjust itself to the generated fake samples quickly; thus, the parameter change rate will just be obviously higher than others in the first several rounds since the attack begins.

Finally, we propose an anomaly detection algorithm to detect the anomalous features analyzed above. It only utilizes the comparison of parameter change rate and its related statistical information. The detection of the second anomalous feature is very similar to outlier detection in time series, although the data we collected is quite different from time series. There is a lot of related work on anomaly detection for time-series data [25–27]. Inspired by the idea of employing a window-based forecasting model for time-series data [27], we develop our anomaly detection algorithm with a sliding window used. However, different from the sliding window in [27], which is used to predict future values, our sliding window is used to calculate a statistic in it. Our malicious participant detection algorithm works as shown in Algorithm 1. First, we try to detect the first anomalous feature by directly comparing each participant's parameter change rate with others. If the parameter change rate of participant j is much higher than others, i.e., higher than Gt-Thr1 times the mean of others' parameter change rate, and this situation lasts for consecutive Rd-Thr rounds, then participant j will be judged as malicious. If we fail to detect the first anomalous feature at the current round, we will immediately start to detect the second anomalous feature by comparing the maximum fitted slope of points in the sliding window of participant j with others. If the maximum fitted slope of participant j is greater than Gt-Thr2 times the mean of others' maximum fitted slope, then we conclude that participant j is malicious. The reason why we divide the detection algorithm into two parts is that there are two different anomalous features found from the collected parameter change rate. They are so different that it is hard to find a unified method to detect two anomalous features simultaneously. So we divide the detection algorithm into two parts to detect two anomalous features, respectively. One may worry that it will cause false positives

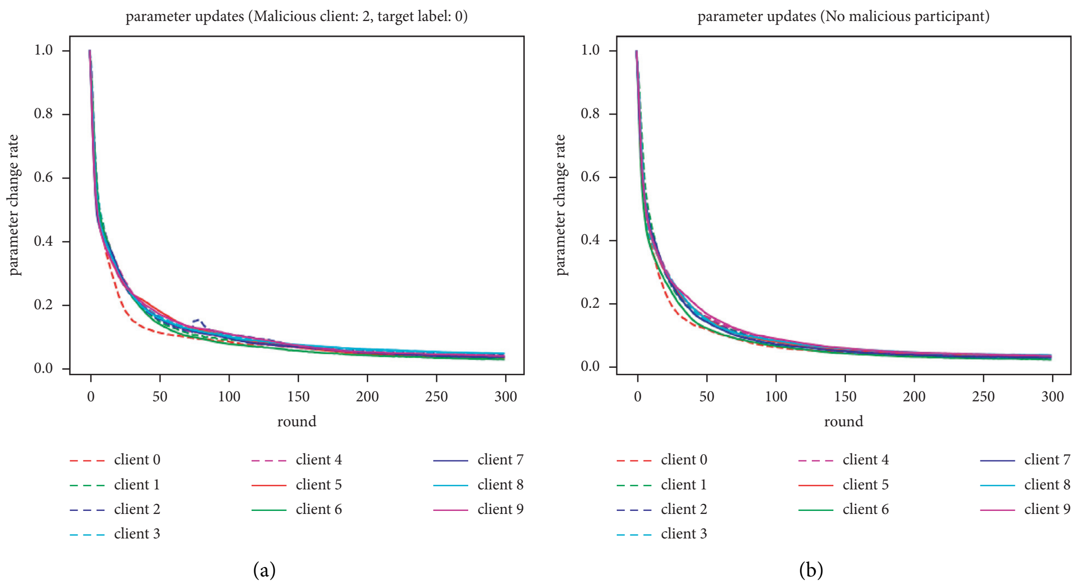


Figure 1: Comparison of interested parameter updates with/without a malicious participant on MNIST dataset. (a) with a malicious participant, (b) without a malicious participant.

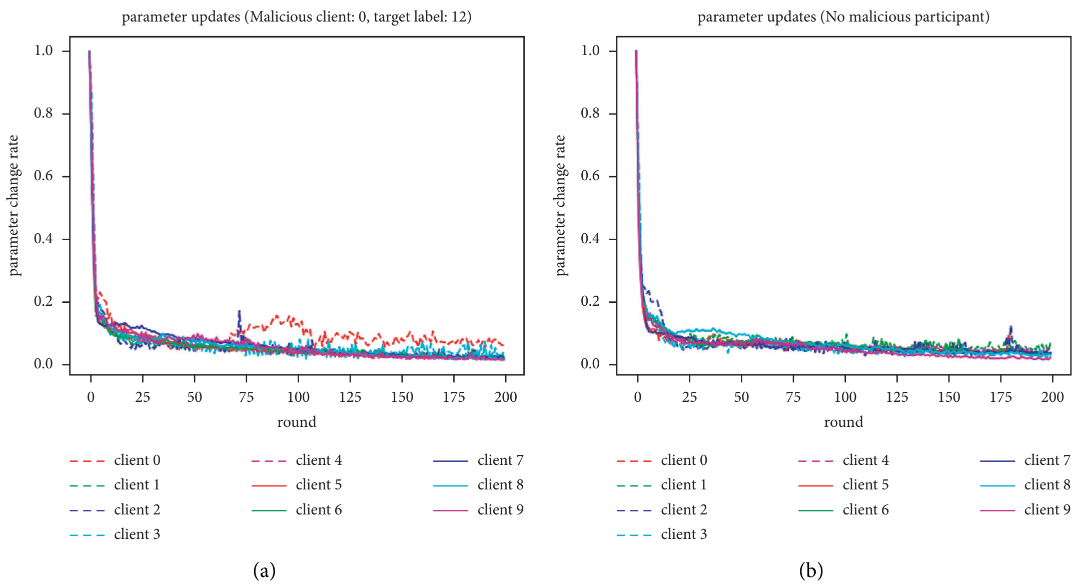


Figure 2: Comparison of interested parameter updates with/without a malicious participant on the GTSRB dataset. (a) With a malicious participant, (b) without a malicious participant.

easily. However, as long as the second part is designed well and the hyperparameters are chosen appropriately, few false positives will be caused. In fact, the second part eliminates false negatives rather than causing false positives. It is worth mentioning that there are 6 hyperparameters in the algorithm and we introduce them brifely in Table 1. One may doubt whether these hyperparameters are necessary, and here is the explanation. If we remove either Rd-Thr or Gt-Thr1, the algorithm will become too radical and cause lots of false positives. As for Win-Size and Sl-Step, they are indispensable parameters for a sliding window. Gt-Thr2 and Sp-Thr control the allowed steepening in the curve and removing them will also cause many false positives.

4. Evaluation

4.1. Experimental Setup

4.1.1. Datasets. We conducted an experiment on two widely used datasets, MNIST and GTSRB. MNIST is a large collection of handwritten digits. It has a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28×28

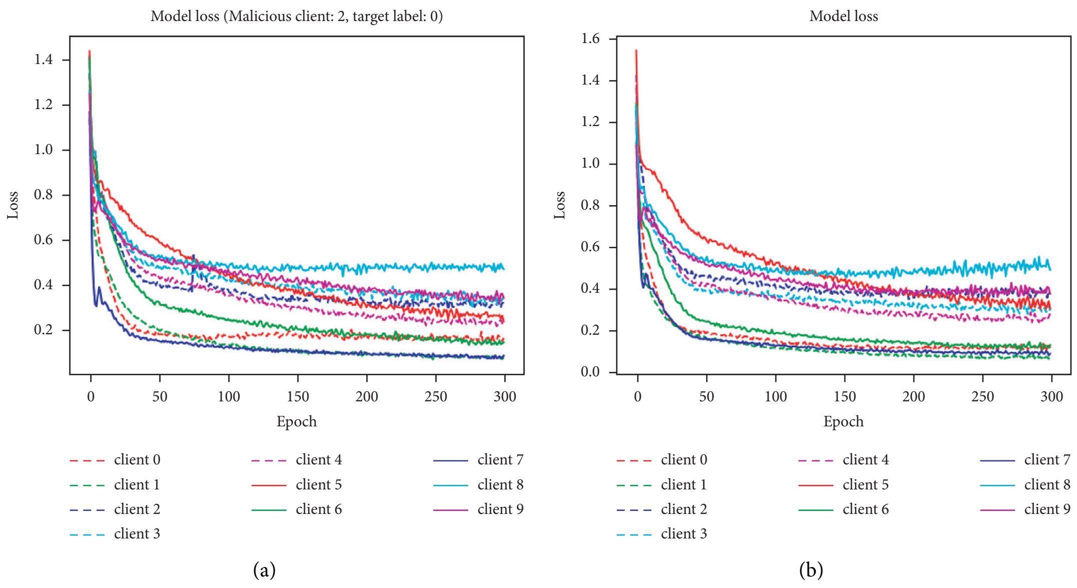


Figure 3: Comparison of local training loss with/without malicious participant on MNIST dataset. (a) With a malicious participant, (b) without a malicious participant.

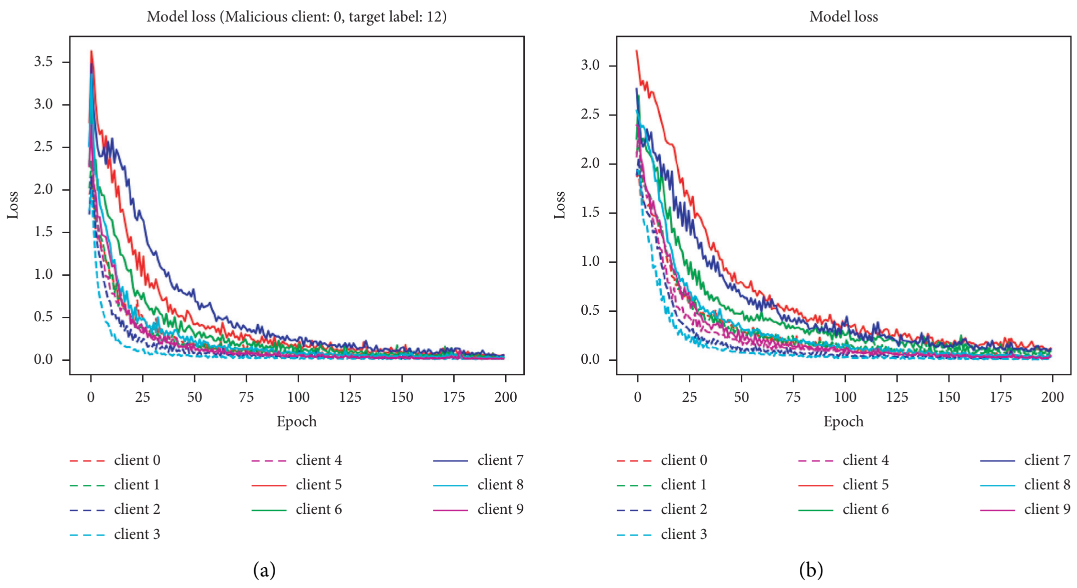


Figure 4: Comparison of local training loss with/without a malicious participant on the GTSRB dataset. (a) With a malicious participant, (b) without a malicious participant.

grayscale image. German Traffic Sign Recognition Dataset (GTSRB) is an image classification dataset consisting of photos of traffic signs. The images are classified into 43 classes. The training set contains 39209 labeled images and the test set contains 12630 images. The image size is $64 \times 64 \times 3$.

4.1.2. Scenario Settings. We consider both scenarios with a malicious participant and without any malicious participant on MNIST and GTSRB datasets; thus, our experiments can be divided into four parts. For each part, we generate 100 samples and run the detection algorithm on each sample. Here, sample means the parameter change rate of every participant during each round. For scenarios without any malicious participant, we run the same code 100 times to generate 100 samples. Due to the randomness during training, the 100 samples are not exactly the same. For scenarios with a malicious participant, we take each participant in turn as malicious and generate 10 samples for the malicious participant. As there are 10 participants in our settings, it leads to 100 samples in total. The settings of the two scenarios are all the same, except that in the scenario

```
input: m: number of participants;
R: number of training rounds;
: updates of our interested parameters, whose size is $R \times m$;
output: a list of malicious participants;
(1) suspects ←∅;
(2) cnt ← [] *$m$;
(3) for$i = 1 to R$do
(4)     for$j = 1 to m$do
(5)         avg ← mean of $W$ except for $W_{ij}$
(6)         if$W_{ij} > Gt\text{-}Thr1^{*} avg$ then
(7)             cnt[j] ← cnt[j] + 1;
(8)         if $cnt[j] \geq Rd\text{-}Thr$ then
(9)             suspects ← suspects ∪{$j$};
(10)              break;
(11)          else
(12)              cnt[j] ← 0;
(13) if $suspects$ $is$∅then
(14)          max_slope ← [-inf] *$m$;
(15)          for$i$ = Win − Size;$i< = R;i+ = Sl$ − Stepdo
(16)              for$j = 0;j < m;j++$do
(17)                  k,b ← slope and intercept fitted with the least squares method on $W_{[i-Win-Size:\ i]j}$;
(18)                  max_slope[j] ← max(max_slope[j], k)
(19)              for$j = 0;j < m;j++$do
(20)                  avg ← mean of max_slope except for max_slope[j];
(21)                  if $max\_slopes[j] > Sp\text{-}Thr$ and $max\_slopes[j] > Gt\text{-}Thr2^{*} avg$ then
(22)                      suspects ← suspects ∪{$j$};
(23)                      break;
(24) return suspects
```

Algorithm 1: Malicious participant detection.

with a malicious participant, the attacker will additionally train a GAN locally and inject the generated fake data into the original training set.

The attack part of our experiment follows the setup of [1], while we consider a more clever attacker who starts the attack only when the accuracy of the global model reaches some threshold. The threshold is 0.85 for MNIST and 0.6 for GTSRB. The detection algorithm is run by the parameter server each round before it aggregates all the parameters update from all the participants.

4.1.3. Hyperparameter Configurations. We use different hyperparameter configurations on different datasets. As for the attack part of the MNIST dataset, we set the global epoch as 300, the local epoch as 1, and the batch size as 2048. As for the training of GAN, the epoch is set as 1 and the batch size is set as 2048. The number of samples merged with the training set is 500. The attack starts as soon as the accuracy of the global model on the validation set reaches 0.85. There are 10 participants and participant i owns the data of label i. We have each participant taking turns as a malicious participant and generate a target label for him randomly. We apply the Adam optimizer and set the learning rate to be 0.001. In the detection part, to show the impact of the input parameters of our detection algorithm, we try some combinations of Rd-Thr, Gt-Thr1, Win-Size, and Sl-Step, Gt-Thr2 and Sp-Thr.

As for the attack part of the GTSRB dataset, the global epoch is 200, the local epoch is 1, and the batch size is 512. With regard to the training of GAN, we set the epoch as 3 and batch size as 256. There are 300 samples merged with the training set. The threshold of accuracy to start an attack is 0.6. There are 10 participants and we distribute the total 43 labels as evenly as possible. First, we distribute 4 labels to each participant,; then, the left 3 labels are distributed to the first 3 participants. Each participant takes turns as the malicious participant and randomly picks a target label. Adam optimizer is applied and the learning rate is set as 0.001. As for detection, different combinations of Rd-Thr, Gt-Thr1, Win-Size and Sl-Step, Gt-Thr2, and Sp-Thr are tried.

4.1.4. Evaluation Metrics. As for scenario with a malicious participant, we use the following two metrics: (1) Recall: The number of samples where the malicious participant is found, divided by the number of total samples. The higher the recall rate, the less the algorithm misses the malicious participant. Thus, the recall rate measures the ability of the algorithm to cover the malicious participant. When finding out the malicious participant, the algorithm may judge normal participants as malicious at the same time. This case also contributes to the recall rate. Thus, we use another metric called error rate to measure the accuracy of the algorithm. (2) error rate: The number of samples where a normal participant is judged as malicious, divided by the number of total samples. The lower the error rate, the less the algorithm causes false positives. Thus, it measures how correct the detection results are. As for the scenario without any

Table 1: Hyperparameters used in the detection algorithm.

Full name	Abbreviation	Meaning
roundsThreshold	Rd-thr	Number of consecutive rounds threshold used to detect the first anomalous feature
greaterThreshold1	Gt-Thr1	Multiple thresholds used to detect the first anomalous feature
windowSize	Win-size	Size of sliding-window used to detect the second anomalous feature
slidingStep	Sl-step	Sliding step of the sliding window used to detect the second anomalous feature
slopeThreshold	Sp-thr	Slope threshold used to detect the second anomalous feature
greaterThreshold2	Gt-Thr2	Multiple thresholds used to detect the second anomalous feature

malicious participant, since recall is meaningless for this scenario, we only use error rate as the metric to measure the correctness of our detection results. Actually, the error rate in this scenario is equal to the false-positive rate and we will call it a false-positive rate in the following sections.

4.2. Detection Results

4.2.1. Result MNIST. Table 2 shows the recall, ER, and FPR on MNIST corresponding to different Rd-Thr and Gt-Thr1, where Win-Size, Sl-Step, Gt-Thr2, Sp-Thr are fixed as 5, 2, 100, 0.002, respectively. We can see that the recall decreases as Rd-Thr increases and increases as the Gt-Thr1 increases, while both the ER and FPR decrease as Rd-Thr or Gt-Thr1 increases. Table 3 shows the recall, ER, and FPR on MNIST corresponding to different Win-Size and Sl-Step, where Rd-Thr, Gt-Thr1, Sp-Thr, and Gt-Thr1 are fixed as 3, 2, 100, 0.002, respectively. Since we always set Sl-Step as half of Win-Size, we only consider the effect of Win-Size. Conclusions can be drawn that the recall decreases as the Win-Size increases. Both the ER and FPR are not affected by Win-Size as they always keep zero. It is worth mentioning that the smaller the Win-Size is, the sooner the attacker is detected. The Win-Size can be regarded as the delay of our detection algorithm. Under the best hyperparameters setting, the recall is 0.99, and both ER and FPR are zero. It means in the scenario with a malicious participant, we only miss the attacker once in 100 samples. While in the scenario without any malicious participant, our detection algorithm does not make any mistake in 100 samples. The best Rd-Thr is 3, which means we can detect the attacker with a delay of 3 rounds.

4.2.2. Result GTSRB. Table 4 shows the recall, ER, and FPR on GTSRB corresponding to different Rd-Thr and Gt-Thr1, where Win-Size, Sl-Step, Gt-Thr2, and Sp-Thr are fixed as 10, 5, 100, 0.005, respectively. We can see that the recall slightly decreases as the Rd-Thr or Gt-Thr1 increases, while both the ER and FPR decline relatively largely as the Rd-Thr increases and decline sharply as the Gt-Thr1 increases. Table 5 shows the recall, ER, and FPR on GTSRB corresponding to different Win-Size and Sl-Step, where Rd-Thr, Gt-Thr1, Sp-Thr, and Gt-Thr1 are fixed as 4, 2, 100, 0.005, respectively. It is easy to see that the recall first increases and then decreases as the Win-Size increases. While the error rate is not affected by the Win-Size and keeps as 0.01, the FPR decreases as the Win-Size increases. Under the best hyperparameters setting, the recall is 0.97, the ER is 0.01, and the FPR is 0.02. It means in the scenario with a malicious participant, we only miss the attacker 3 times and misjudge a benign participant as an attacker once in 100 samples. While in the scenario without any malicious participant, our detection algorithm only makes a mistake twice in 100 samples.

Table 2: Recall, error rate (ER), and false positive rate (FPR) on MNIST corresponding to different roundsThreshold (Rd-Thr) and greaterThreshold1 (Gt-Thr1). The windowSize (Win-Size), slidingStep (Sl-Step), greaterThreshold2 (Gt-Thr2), and slopeThreshold (Sp-Thr) are fixed as 5, 2, 100, 0.002, respectively.

Rd-Thr	Gt-Thr1	recall	ER	FPR
3	1.2	0.91	0.59	0.79
3	1.5	0.99	0	0
5	1.2	0.88	0.56	0.79
5	1.5	0.99	0	0
7	1.2	0.84	0.51	0.74
7	1.5	0.99	0	0

Table 3: Recall, error rate (ER), and false positive rate (FPR) on MNIST corresponding to different windowSize (Win-Size) and slidingStep (Sl-Step). The roundsThreshold (Rd-Thr), greaterThreshold1 (Gt-Thr1), greaterThreshold2(Gt-Thr2), and slopeThreshold (Sp-Thr) are fixed as 3, 2, 100, 0.002, respectively.

Win-Size	Sl-Step	recall	ER	FPR
3	1	0.99	0	0
5	2	0.99	0	0
7	3	0.95	0	0
9	4	0.82	0	0
11	5	0.66	0	0
13	6	0.52	0	0

Table 4: Recall, error rate (ER), and false positive rate (FPR) on GTSRB corresponding to different roundsThreshold (Rd-Thr) and greaterThreshold1 (Gt-Thr1). The windowSize (Win-Size), slidingStep (Sl-Step), greaterThreshold2 (Gt-Thr2),and slopeThreshold (Sp-Thr) are fixed as 10, 5, 100, 0.005, respectively.

Rd-Thr	Gt-Thr1	recall	ER	FPR
2	1.5	1	0.86	1
2	2	0.98	0.11	0.31
4	1.5	1	0.63	0.94
4	2	0.97	0.01	0.02
6	1.5	0.98	0.34	0.75
6	2	0.96	0.01	0.01

4.2.3. Discussion. The experiments are conducted under the assumption that the attacker starts the attack after the model begins to converge. In this case, the attack is more effective

Table 5: Recall, error rate (ER), and false positive rate (FPR) on GTSRB corresponding to different windowSize (Win-Size) and slidingStep (Sl-Step). The roundsThreshold (Rd-Thr), greaterThreshold1 (Gt-Thr1), greaterThreshold2 (Gt-Thr2), and slopeThreshold (Sp-Thr) are fixed as 4, 2, 100, 0.005, respectively.

Win-Size	Sl-Step	recall	ER	FPR
4	2	0.94	0.01	0.58
6	3	0.94	0.01	0.1
8	4	0.96	0.01	0.04
10	5	0.97	0.01	0.02
12	6	0.95	0.01	0.02
14	7	0.96	0.01	0.02

and stealthy since the model about to converge contains enough information about the training set, and there is not much time left to detect the attacker. However, the chances are that the attacker may start the attack from the beginning, which deserves some discussion. In this case, different anomalous features may show, and traditional secure aggregation methods such as Krum [28] and trimmed mean [29] may work, with the cost of slowing the model's convergence.

5. Conclusion

In this work, we present a scheme to detect the GAN-based information leakage attack in FL, where the attacker is one of the participants in the FL system. We aim to detect the malicious participant accurately with a small computational overhead in real time. We only utilize the biases in the last convolutional layer and manage to find general anomalous features from updates of these biases. Then an anomalous detection algorithm based on statistics is proposed to detect the previously found anomalous features. We conduct extensive experiments to evaluate the effectiveness of our detection scheme. The results demonstrate that our proposed detection scheme can detect the malicious participant accurately and efficiently in near real time. [30–32].

Data Availability

The MNIST dataset used to support the findings of this study has been deposited in the website http://yann.lecun.com/exdb/mnist/. The GTSRB dataset used to support the findings of this study have been deposited in the website https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign?select = Train.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Science and Technology Project of State Grid Corporation of China: "Research on Power Data Security Collaboration Technology Based on Federated Learning" (Grant No. 5700-202190184A-0-0-00).

References

[1] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 603–618, February 2017.

[2] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: user-level privacy leakage from federated learning," in *Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 2512–2520, IEEE, Paris, France, April 2019.

[3] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, "PoisonGAN: generative poisoning attacks against federated learning in edge computing systems," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3310–3322, 2020.

[4] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *Proceedings of the 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 374–380, IEEE, August 2019.

[5] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, 2021.

[6] Y. Zhao, J. Chen, J. Zhang, D. Wu, J. Teng, and S. Yu, "PDGAN: a novel poisoning defense method in federated learning using generative adversarial network," in *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing*, pp. 595–609, Springer, Cham, Melbourne, Australia, December 2019.

[7] J. Hayes and O. Ohrimenko, "Contamination attacks and mitigation in multi-party machine learning," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 6604–6615, Montréal, Canada, December 2018.

[8] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321, Monticello, IL, USA, September 2015.

[9] X. Huang, Y. Ding, Z. L. Jiang, S. Qi, X. Wang, and Q. Liao, "DP-FL: a novel differentially private federated learning framework for the unbalanced data," *World Wide Web*, vol. 23, no. 4, pp. 2529–2545, 2020.

[10] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134–2143, 2019.

[11] Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[12] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: efficient homomorphic encryption for cross-silo federated learning," in *Proceedings of the 2020 USENIX Annual Technical Conference (USENIXATC 20)*, pp. 493–506, Boston, MA, USA, 2020.

[13] M. A. Rahman, M. S. Hossain, M. S. Islam, N. A. Alrajesh, and G. Muhammad, "Secure and provenance enhanced Internet of health things framework: a blockchain managed federated learning approach," *Ieee Access*, vol. 8, Article ID 205071, 2020.

[14] Z. Jiang, W. Wang, and Y. Liu, "FLASHE: additively symmetric homomorphic encryption for cross-silo federated learning," 2021, https://arxiv.org/abs/2109.00675.

[15] X. Zhang, A. Fu, H. Wang, C. Zhou, and Z. Chen, "A privacy-preserving and verifiable federated learning scheme," in *Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Dublin, Ireland, June 2020.

[16] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, D. Ali, and S. Gautam, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[17] Y. Xiong, F. Xu, and S. Zhong, "Detecting GAN-based privacy attack in distributed learning," in *Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Dublin, Ireland, June 2020.

[18] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proceedings of the 29th USENIX Security Symposium (USENIX Security 20)*, pp. 1605–1622, Boston, MA, USA, 2020.

[19] W. Jiang, H. Li, S. Liu, Y. Ren, and M. He, "A flexible poisoning attack against machine learning," in *Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Shanghai, China, May 2019.

[20] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, https://arxiv.org/abs/1712.05526.

[21] L. Muñoz-González, B. Biggio, A. Demontis et al., "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 27–38, Dallas, TX, USA, August 2017.

[22] N. Baracaldo, B. Chen, H. Ludwig, and J. A. Safavi, "Mitigating poisoning attacks on machine learning models: a data provenance based approach," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 103–110, Dallas, TX, USA, November 2017.

[23] C. Liu, B. Li, Y. Vorobeychik, and A. Oprea, "Robust linear regression against training data poisoning," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 91–102, Dallas, TX, USA, 2017.

[24] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proceedings of the International Conference on Machine Learning. PMLR*, pp. 634–643, Long Beach, CA, USA, November 2019.

[25] S. Weixing, Z. Yunlong, L. Fang, and H. Kunyuan, "Outliers and change-points detection algorithm for time series," *Journal of Computer Research and Development*, vol. 51, no. 4, 2014.

[26] M. Braei and S. Wagner, "Anomaly detection in univariate time-series: a survey on the state-of-the-art," 2020, https://arxiv.org/abs/2004.00433.

[27] Y. Yu, Y. Zhu, S. Li, and D. Wan, "Time series outlier detection based on sliding window prediction," *Mathematical Problems in Engineering*, vol. 2014, Article ID 879736, 14 pages, 2014.

[28] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[29] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: towards optimal statistical rates," in *Proceedings of the International Conference on Machine Learning. PMLR*, pp. 5650–5659, Stockholm, Sweden, March 2018.

[30] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the Artificial intelligence and statistics. PMLR*, pp. 1273–1282, San Francisco, CA, USA, February 2017.

[31] W. Y. B. Lim, N. C. Luong, D. T. Hoang et al., "Federated learning in mobile edge networks: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[32] I. Goodfellow, J. Pouget-Abadie, and M. Mirza, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 4769187, 16 pages
https://doi.org/10.1155/2022/4769187

Research Article

Blockchain-Based Privacy-Preserving Vaccine Passport System

Yangzhou Cao,[1] Jiageng Chen ,[2] and Yajun Cao[3]

[1]*Central China Normal UniversityWollongong Joint Institute, Central China Normal University Wuhan, Wuhan, China*
[2]*School of Computer, Central China Normal University Wuhan, Wuhan, China*
[3]*Yichang Center for Disease Control, Prevention Yichang, Yichang, China*

Correspondence should be addressed to Jiageng Chen; chinkako@gmail.com

Received 11 November 2021; Accepted 13 January 2022; Published 21 March 2022

Academic Editor: Weizhi Meng

In this study, we propose a blockchain-based privacy-preserving vaccine passport system for the global prevention and control of infectious diseases. The system operates a double-chain framework which consists of a public blockchain and a consortium blockchain. Among them, the combination of the immutability of the public blockchain and Internet of Things (IoT) technology in the supply chain ensures the openness and transparency of the cold chain logistics records of the vaccines covering the stages from auditing to the target vaccination hospitals. The system adopts the consortium blockchain to achieve the balance between the protection of users' vaccination privacy and auditing by the government departments. Specifically, a distributed system-based threshold signature is adopted in the vaccine qualification phase to resist collusion between the vaccine manufacturing company and vaccine approval institutions. The cryptographic tools such as the anonymous credentials, zero-knowledge protocols, and range proofs ensure that users do not disclose any private information other than proving that they have a legally valid vaccine passport when users display the vaccine passports to customs. At the same time, customs can apply various vaccine prevention policies based on the conditions on the specific vaccine passports. Regarding the security properties of the system, a formal security model is given along with the corresponding security proofs.

1. Introduction

With the outbreak of COVID-19 in early 2020, the global defense against the spread of COVID-19 has been severely tested. Following the outbreak, scientists, physicians, and vaccine manufacturers in various countries engaged in the development of vaccines for the coronavirus. On January 24, 2020, the Chinese Center for Disease Control and Prevention (CDC) successfully isolated the first coronavirus strain in China [1]. The National Pathogenic Microbial Resource Library released information and electron microscopy photos of this strain (Wuhan strain 01 of the novel coronavirus), as well as important authoritative information such as primers and probe sequences for nucleic acid detection of the novel coronavirus, all of which laid the foundation for vaccine development. On this basis, COVID-19 vaccines in each country were promoted from the R&D stage to the clinical trial stage. In the second half of 2020, COVID-19 vaccines developed in each country gradually were approved for marketing by various national approval authorities.

At the stage when COVID-19 vaccines were introduced into the market and society, vaccination would face social problems in various aspects. With the gradual introduction of COVID-19 vaccines, vaccine management and vaccination become important issues for national governments. Especially in emergency cases when the COVID-19 vaccine is not sufficient, it is vital for the privacy of vaccination information to be protected to prevent social conflicts. As the epidemic is effectively controlled in various regions, the people returning from various countries and regions are also a serious test for the prevention and control of the local epidemic. Therefore, the application of vaccine passports was born.

As countries around the world gradually recovered from the effects of the COVID-19 epidemic, urgent cultural communication and trade between countries led to the implementation of vaccine passports. On July 26, 2021, municipalities, wards, towns, and villages throughout Japan began accepting applications for the official certificate ("vaccine passport") for COVID-19 vaccine [2]. The key information of the vaccine certificate includes the individual's name, date of birth, passport number, type of vaccine used, and date of vaccination. The idea is that the certificates exempt travelers from Japan from quarantine and other antivirus measures after their arrival in overseas destinations. However, the Japanese government does not make such exemptions for people who enter Japan with vaccine passports issued by other nations for now, and the government is considering making vaccine passports digital. At 1 : 00 p.m. Vancouver time on August 23, the Premier of British Columbia held a press conference to announce the implementation rules for the British Columbia vaccine certificate. Starting from September 13, people attending indoor concerts, sporting events, movie theaters, and other nondiscretionary activities must receive at least one dose of the COVID-19 vaccine and show proof of it. On October 24, the vaccination requirement will be increased to 7 days after completing two doses of the vaccine before being allowed to enter certain public places with a vaccination card [3].

The vaccine passport should be an internationally recognized certificate of vaccination for COVID-19 [4] and possibly other types as well. In February 2021, the concept of the vaccine passport was still in the initial stages of controversy, and international opinion was divided. In the view of proponents, the emergence, use, and popularity of a vaccine passport would significantly mitigate the impact of the COVID-19 pneumonia outbreak on international travel and facilitate global economic recovery. In contrast, in the view of opponents, it is far from simple to establish a globally circulating and mutually recognized certification system that can effectively protect the privacy and ensure fairness.

The purpose of this study is to design protocols to ensure the transparency and privacy of vaccination, as well as the privacy of vaccine passports through the technology of cryptography to address the issues of privacy protection. However, we point out that the vaccine passports are subjected to a global consensus. It assumes that the design, implementation, and operation of the vaccine passport system should be supported and accepted by countries around the world.

1.1. Prior and Related Work. COVID-19 outbreak led to research on vaccine supply chain improvements. Many researchers in cryptography proposed blockchain-based systems for the distribution and management of vaccine supply chains. The idea is to take advantage of the non-tamperability of blockchain, and the nature of jointly maintaining a unified ledger to ensure the supply of vaccines is regulated and transparent. Meanwhile, with the update and development of IoT technology, IoT in the field of traditional commodity logistics has been migrated to the field of logistics and transportation of pharmaceutical products. Among them, the monitoring and supervision of environmental conditions of vaccines belonging to biological products in the process of cold chain logistics transportation can combine IoT devices with sensors. Specific sensors feedback to the CDC, which monitors the logistics of biologics, about the humidity, temperature, light protection, and other transport conditions during the cold chain transportation of vaccines. As vaccination users, they also own the right to know that vaccine production and transportation meet quality control. Cui et al. [5] proposed a blockchain-based vaccine tracking system to protect the entire vaccine cycle. The blockchain is used as a global, unique, and verifiable database to store all circulating databases. Antal et al. [6] used Ethernet's smart contract technology to achieve the integrity of guaranteed vaccine data and the immutability of registration for vaccinators, avoiding identity theft and imitation. Yong et al. [7] applied machine learning techniques to analyze and process data in the vaccine blockchain.

Abid's proposed vaccine platform [8] provides a sovereign user identity that gives users full control over their data and encrypts personally identifiable information to enhance privacy. The platform also leverages W3C verifiable credential standards to facilitate instant verification of COVID-19 proofs and allow users to share selected information with trusted parties. However, the platform's privacy is protected by hashing sensitive information and then storing it on the blockchain, which is at risk when the data are broadcasted. Haque et al. [9], the authors proposed an architectural framework of a permission blockchain-based vaccination passport for the European Union's General Data Protection Regulations (GDPR). The scope of this regulation is broad, and any organization that collects, transfers, retains, or processes personal information involving all EU member states is subject to the regulation. Then, the double-chain structured blockchain system proposed by Qiu and Zhu [10] combines a public blockchain and a private blockchain to manage and store data information in different processes of vaccine logistics and vaccination. However, the user privacy of this system relies too much on the authorization mechanism of the private blockchain.

1.2. Contributions. In this study, we propose a double-chain framework with the vaccine cold chain logistics system and vaccination record system. We introduce threshold signature technology at the vaccine audit stage of public blockchain to deal with complicity between vaccine manufacturing companies and vaccine approval institutions. Second, it applies the consortium blockchain to record the information of vaccination hospitals to give vaccination to users. Its process ensures the privacy of vaccination hospitals, vaccination users, and vaccination vaccines and reserves the right to reveal and audit the vaccination information records by government departments under special circumstances.

In the issuance and presenting of the vaccine passport, the use of anonymous credential, ring signature, and range proofs ensures that the validity of the vaccine passport is proven without revealing the user's vaccination hospital and identity information during the process.

1.3. Paper Organization. In the subsequent content of this study, we present the entities and the system threat model in the vaccine passport system in Section 2. We show the cryptographic techniques and tools used to build the system protocol in Section 3. Section 4 of this article provides the structural design of the system and the specific protocol design. We give the security analysis and proof of the protocols in this model in Section 5. We give a system evaluation in Section 6, and we finally conclude this article in Section 7.

2. Assumptions and Threat Model

2.1. Entities and Assumptions. Before presenting the system structure, we introduce the entity participants in the system.

(i) International coalition government, $\mathscr{GV}$: it acts as the system's CA to manage the authorization and authentication of each participant. It acts as a trusted third party for threshold signatures in the vaccine approval process. In exceptional cases, it can audit the encrypted information in the consortium blockchain that records vaccinations.

(ii) Hospital, $\mathscr{HOSP}$: it issues a credential for the user's vaccine passport after completion of the vaccination and uploads the information recording the vaccination to the consortium blockchain.

(iii) User: the user receives a vaccine passport after completion of vaccination at the hospital. When it is necessary to prove the legitimacy and validity of the vaccine passport to the vaccine passport checkpoint, zero-knowledge proof protocol is applied to protect their privacy.

(iv) Vaccine manufacturing company: it sends samples of the vaccine to be tested to the vaccine approval institutions in each country for approval. Once the vaccine is approved, the batch is issued a certificate of authorization.

(v) Vaccine approval institutions, $\mathscr{AI}$: each country's approval body tests the submitted vaccine samples according to its own standards. The approved vaccine approval institution signs a threshold signature for the vaccine. The $\mathscr{GV}$ issues a threshold signature certificate to the vaccine lot after (t, n) vaccine approval institutions have been met and approved simultaneously.

(vi) Vaccine passport checkpoint: it verifies the user's identification and proof of the legitimacy and validity of the vaccine passport. It also takes the appropriate vaccination measures and policies for the fulfillment of the conditions of the user's vaccine passport.

(vii) Vaccine transit centers: they act as a transit point for vaccine shipments connecting vaccine companies to the CDC. Information on storage and transport conditions during cold chain logistics is uploaded.

(viii) CDC: it audits the vaccine cold chain logistics process for compliance with biologics-related regulations. If so, the vaccine is held in temporary storage and eventually shipped to the hospital where it is administered.

Considering the specific prerequisite assumptions for the application of the vaccine passport system to realistic scenarios and specific programs, the system provides the following reasonable assumptions.

(i) The authority of the international coalition government is recognized by every country in the world

(ii) Countries strictly adhere to the normal operation of the system

(iii) The number of corrupted institutions in vaccine approval institutions is less than half of the total number

(iv) Authorized hospitals follow the hospital code of conduct and do not conspire with users

(v) Users do not disclose or share their secret keys

2.2. Threat Model. In this study, we do not consider network-level security attacks, physical hardware-level damage, and software vulnerability penetration during the engineering implementation of the protocol. In this study, we only consider cryptographic attacks towards the protocol design.

(i) In the threat model of this study, we assume that $\mathscr{GV}$ and auditor are completely honest. They operate according to the protocol algorithm and do not disclose the privacy parameters generated.

(ii) In the threshold signature phase, adversary is allowed to corrupt up to $t < n/2\mathscr{AI}$ s. $\mathscr{GV}$ does not disclose institutional audit signatures to vaccine manufacturing companies.

(iii) In the vaccination information record uploading consortium blockchain phase, all peers except the auditor and $\mathscr{GV}$ are assumed to be honest-but-curious; they try to break the privacy by passively eavesdropping on the inputs and outputs of the protocol but not actively violating the protocol process.

(iv) In the vaccine passport display phase, vaccine passport checkpoint is assumed to be honest-but-curious; it tries to get the user's private data, but it still follows the protocols.

3. Preliminaries

3.1. Bilinear Pairing. Let $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ a bilinear map where $\mathbb{G}_1$ is a GDH group and $\mathbb{G}_1 \neq \mathbb{G}_2$ in our protocol. $\mathbb{G}_1, \mathbb{G}_2$ are the two multiplicative cyclic groups of prime order q. The bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ has the following three properties:

(i) Bilinear: for all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and $\alpha, \beta \in \mathbb{Z}_q$, it holds that $e(g_1^{\alpha}, g_2^{\beta}) = e(g_1, g_2)^{\alpha\beta}$;

(ii) Computability: there exists an efficient algorithm to calculate $e(g_1, g_2)$, where $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$;

(iii) Nondegenerate: $e(g_1, g_2) \neq 1$ for $\exists g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, where 1 is the unit element in the multiplicative cyclic group.

3.2. q-Strong Diffie–Hellman Assumption. The q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined that for adversary $\mathscr{A}$ on input a $(q+2)$-tuple $(g_1, g_2, g_2^x, g_2^{x^2}, \ldots, g_2^{x^q}) \in\in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$

$$\Pr\left[\begin{array}{c} A = g_1^{1/(x+c)}: \\ (A, c) \longleftarrow \mathscr{A}\left(g_1, g_2, g_2^x, g_2^{x^2}, \ldots, g_2^{x^q}\right) \end{array}\right] \leqslant negl(\lambda). \quad (1)$$

3.3. Threshold Signature Scheme. The (t, n) threshold signature scheme allows any t signers among n signers to generate a signature for a message, but less than t signers participate to generate a valid signature. The threshold signature scheme can build a robust signature system to prevent the unlawful behavior of some signers. The threshold signature scheme consists of the following four algorithms:

(i) ThresholdKeyGen (λ, n, t): for distributed systems, threshold key generation algorithm is a protocol that runs interactively among many participants. With the input security parameters λ, number of users n, and threshold t, it outputs the secret share x_i for each participant, such that $(x_1, \ldots, x_n) \longrightarrow^{(} t, n)$ sk.

(ii) Sign (x_i, m): the signers in the participants output the signature share σ_i based on the input secret share x_i and the message m.

(iii) Reconstruction (σ_i): the resulting signature σ can be generated by a trusted third party based on the signature share σ_i of not less than t signers.

(vi) Verify (pk, m, σ): the verification algorithm inputs the verification public key pk, message m, and resulting signature σ and outputs 1 when the signature is successfully verified; otherwise, it outputs 0.

3.4. Ring Signature Scheme. A ring signature is a digital signature that can be executed by any member of a group of users that each have a pair of keys, so that a message with a ring signature is recognized by someone in a particular group. But, it is computationally infeasible to determine which group member's key is used to generate the signature, which is one of the security properties of ring signatures. All possible signers are formed into a ring. Each possible signer is called a ring member. The ring member that generates the signature is called a signer, and each other ring member is called a nonsigner. The ring signature scheme consists of the following three algorithms:

(i) KeyGen (λ, n): let ring $R = \{R_1, \ldots, R_n\}$. With the input security parameters λ, it outputs each user public-secret key pair $(\mathrm{sk}_i, \mathrm{pk}_i)$. Assume that the signing member is R_s.

(ii) Sign $(\mathrm{sk}_s, m, \{\mathrm{pk}_i\}_{i\in\{1,\ldots,n\}, i\neq s})$: the signer R_s generates a ring signature σ_{ring} on message m with its own secret key sk_s and the public keys $\{\mathrm{pk}_i\}$ of other members.

(iii) Verify $(\{\mathrm{pk}_i\}_{i\in\{1,\ldots,n\}}, m, \sigma_{\mathrm{ring}})$: the verification algorithm is with the input of public keys $\{\mathrm{pk}_i\}_{i\in\{1,\ldots,n\}}$, message m, and ring signature σ_{ring} and outputs 1 when the signature is successfully verified; otherwise, it outputs 0.

3.5. Zero-Knowledge Proof. A zero-knowledge proof is a protocol that the prover P can convince the verifier V that an argument is correct without providing any useful information to the verifier. A zero-knowledge proof is essentially an agreement involving two or more parties, i.e., a series of steps that two or more parties need to take to accomplish a task. The prover convinces the verifier that he or she knows or has a certain message, but the proof process cannot divulge any information about the proven message to the verifier. In our system protocol design, we focus on zero-knowledge proof for NP language $L_R = \{y | \exists \omega \ \ s.t. \ (y, \omega) \in R\}$, where ω is a witness for statement y. A zero-knowledge proof protocol between P and V satisfies the following three properties:

(i) Completeness: if $y \in L_R$, prover P convinces V that his statement is true with probability $1 - \mathrm{negl}(\lambda)$.

(ii) Soundness: if the prover's statement $y \notin L_R$, then any malicious prover P^* convinces an honest verifier of his statement with probability $\mathrm{negl}(\lambda)$.

(iii) Honest verifier zero-knowledge (HVZK): after the proof is executed, the verifier only knows whether the statement of the verifier is true or not, but he does not have access to any other information during the proof. It can also be said that there exists a simulator algorithm Sim that simulates interaction scripts that are nondistinguishable with the real interaction scripts between P and V.

Range proof: range proof is proof that a secret value x, which is encrypted or committed to, lies in a certain interval $[a, b]$. In this study, the secret value x is hidden by Pedersen commitment, such that $C = g^x h^r$. Range proof does not leak any information about the secret value other than the fact that they lie in the interval. The prover needs to provide zero-knowledge proof to the verifier $\mathrm{PK}\{(x, r)\colon C = g^x h^r \wedge x \in [a, b]\}$.

4. Our Proposed System

Before showing the overview of our system model, we present the reasons for choosing the double chain as the basis of the system. The generation of the vaccine passport and the vaccine itself are indivisible. Given the biomedical properties of the vaccine itself, we need a public blockchain to store the production and logistics information of the vaccine. The choice of the consortium blockchain is that vaccination records are information with privacy properties and are required to be privacy protected and regulated. So, it is uncomplicated to achieve the intended effect in a blockchain under authorization.

4.1. Overview. Our system consists of three main phases in the vaccine cold chain logistics phase, as shown in Figure 1.

Step 1. It is for the vaccine manufacturing company to send a batch of vaccine samples that need to be checked to ensure quality to the vaccine approval institutions in each country.

Step 2. It consists of each country's vaccine approval institution passing its review results through a (t, n) threshold (if a total of n vaccine approval institutions are satisfied with the approval of t vaccine approval institutions, then the batch of vaccine is approved). If the batch meets the audit requirements, a certificate is issued for the batch through the threshold signature.

Step 3. It is that the vaccine manufacturing company entrusts the cold chain logistics company with the approved batch of vaccine to send to the target hospital. The sender is the vaccine production company. The receiver is the first vaccine transit center. The transported goods are batches of vaccines. The logistics information is uploaded to the public blockchain after the logistics are completed.

Step 4. It is the uploading of cold chain logistics information between vaccine transfer centers. The sender is the previous vaccine transfer center. The receiver is the next vaccine transfer center. The transported goods are batches of vaccines with the environmental conditions of the temporary storage of vaccines and the signature of the person in charge.

Step 5. It is when the vaccine is delivered at the last logistics transit center; the CDC under whose jurisdiction the target hospital is located audits the entire cold chain logistics storage and transportation for compliance with the logistics requirements for biologics. If the batch of the vaccine cold chain logistics process meets the requirements, the CDC issues a certificate of conformity signature to the batch of vaccine.

Step 6. It is to upload the logistics information between the last vaccine transfer center and the CDC to the public blockchain after the approval of the vaccine cold chain logistics. The sender is the last vaccine transfer center. The receiver is the local CDC, and the transported goods are batches of vaccines with the CDC's certificate for vaccine cold chain logistics.

Step 7. It is to upload the logistics information of the final vaccine delivery from the local CDC to the target hospital to the public blockchain. The sender is the local CDC, and the receiver is the target vaccination hospital. The transported goods are batch of vaccines with a certificate from the CDC for the cold chain logistics of the vaccine and a threshold signature certificate from the vaccine approval institutions. Users are given the right and ability to know the approval results of vaccinations and vaccine cold chain logistics information by viewing the information recorded on the public blockchain before vaccination in hospitals. This helps to achieve openness and transparency of vaccine information to vaccination users.

In the vaccination phase shown in Figure 2, the local hospital completes the uploading of vaccination information to the consortium blockchain while protecting the privacy of the vaccination information.

Step 8. It is after the last injection of the user's vaccine at the local hospital, the hospital creates vaccination information signed by it and sends the vaccination information to the endorser. The sender of the vaccination information is the local hospital. The receiver is the vaccination user. The information transmitted is the details of the vaccine.

Step 9. It is for the endorser to verify the uploaded vaccination information and generate an endorsement signature.

Step 10. It is that the submitting local hospital broadcasts the collected endorsement signatures and the vaccination information itself to the orderers.

Step 11. It is for orderers to broadcast the sorted set of vaccination information to all peers.

Step 12. It is for the committing peer to check if the vaccination information submitted by the orderers has a legitimate certificate issued by the endorser. The committing peer also detects malicious cases where the same vaccination is included in the vaccination information more than once. In this case, the first valid vaccination information will be accepted. Once the uploaded vaccination information is verified by the committing peer, the vaccination information is submitted and the committing peer maintains the state and a copy of the ledger. For the privacy-preserving vaccination information on the consortium blockchain, it is necessary to audit it in case of special circumstances. Auditors have the ability to open the encrypted vaccination information on the consortium blockchain to audit the vaccination details, such as the time of vaccination and vaccine production date.

In the vaccine passport phase in Figure 2.

Step 13. It is where the local hospital opens the vaccination user's commitment to the vaccine production date, vaccine shelf life, vaccine immunity lasting time, and vaccination date. After the hospital confirms that the commitment is correct, a ring signature is generated for the commitment and the international coalition government-issued user identity card. Finally, the ring signature, commitment, and user identity certificate together form the vaccine passport and are sent to the user.

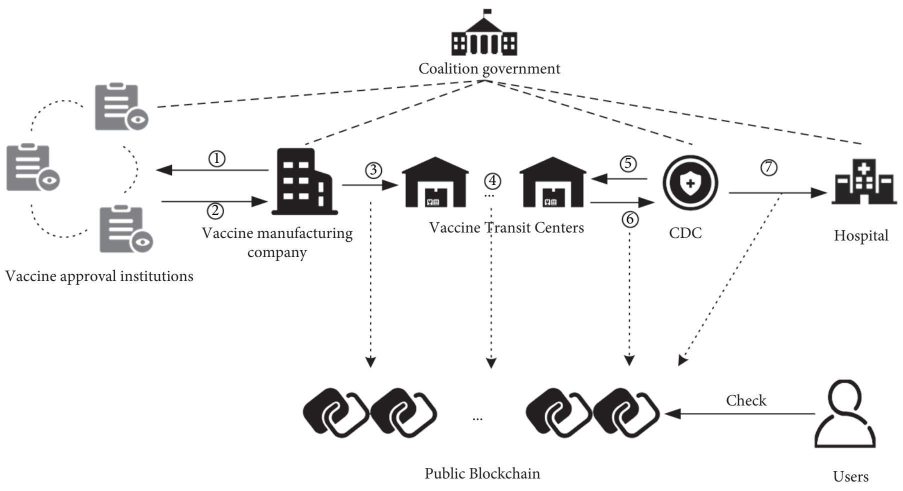


FIGURE 1: Vaccine cold chain logistics phase.

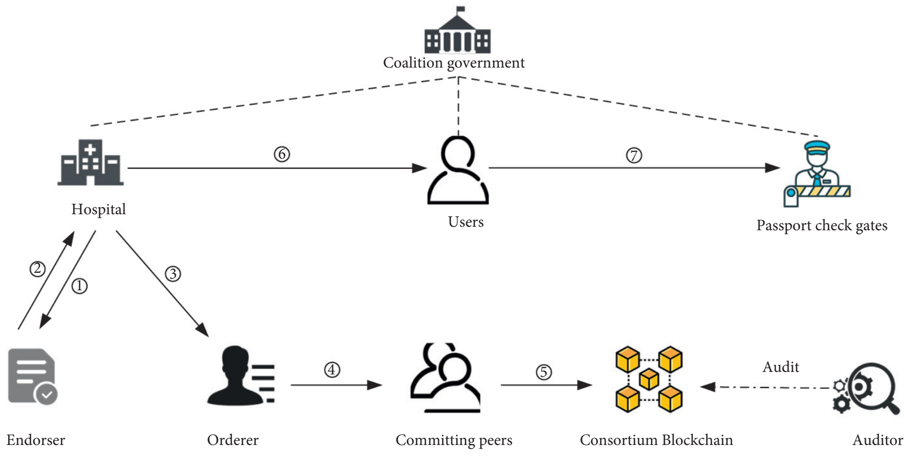


FIGURE 2: Vaccination and vaccine passport phase.

Step 14. It is for the user to first present the vaccine passport to the passport checkpoint. The passport checkpoint verifies the legitimacy of the user's identity and vaccine passport. Next, the user proves the validity of the vaccine passport to the passport checkpoint. This includes the following three items:

(i) The vaccine injected by the user is within the shelf life. If the vaccine injected by the user does not meet this condition, then first, the passport checkpoint needs to report this medical issue to a government authority. This requires a request for an audit of the vaccination information for the batch (including the local vaccination hospital) and a traceability audit of the vaccine batch. Also, the user needs to be reimbursed for the corresponding vaccination.

(ii) The user produces high titers of antibodies to create effective protection. This corresponds to the last date of vaccination plus 14 days [11], which needs to be greater than the current date. If the user's vaccination information does not meet this condition, the passport checkpoint needs to take a quarantine for 14 days before allowing the user to pass.

(iii) The vaccinated user is in the duration of immunization for the vaccine. This is equivalent to the last date of vaccination plus the vaccine immunity lasting time that needs to be less than the current date. If the user's vaccination information does not meet this condition, the passport control point will need to adopt the vaccine again to stimulate an effective antibody prevention strategy.

None of the above proofs will reveal any information about the user's vaccination, including the production date and shelf life of the vaccine.

4.2. Vaccine Cold Chain Logistics. This study adds Boldyreva's [12] threshold signature technique to other blockchain-based vaccine distribution management systems. Vaccine approval institutions in each country that adopt different standards act as participants in the threshold signature. The international coalition government acts as a trusted third party as the group administrator in the threshold signature group. This vaccine approval protocol effectively prevents collusion and corruption between vaccine approval institutions and vaccine manufacturing companies. The vaccine approval institutions approve samples of vaccines to be submitted for review in a distributed structure on a per-share basis. The distributed protocol allows for up to half of the vaccine approval institutions to be malicious. Once the approval of the submitted vaccine is complete, the vaccine manufacturer receives only the results of whether the submitted vaccine batch was approved or not and does not know the respective review opinions of the individual vaccine approval institutions. This prevents the vaccine manufacturing company from influencing the outcome of the approval, thereby, achieving fairness and equity in vaccine approval. Details are outlined as follows.

Setup (1^λ): on input 1^λ, where $\lambda \in \mathbb{N}$ is a security parameter, let $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, a bilinear map, where $\mathbb{G}_1$ is a GDH group and g is the generator of $\mathbb{G}_1$. $\mathbb{G}_2$ and $\mathbb{G}_T$ are the cyclic groups. The participants in our scheme are the set of n vaccine approval institutions $\{\mathscr{A}\mathscr{I}_1, \ldots, \mathscr{A}\mathscr{I}_n\}$. All $\mathscr{A}\mathscr{I}$s are connected by a broadcast channel as well as by secure point-to-point channels including the international coalition government $\mathscr{G}\mathscr{V}$. Let $H\colon \{0,1\}^* \longrightarrow \mathbb{G}_2$ be collision-resistant hash function.

Generating x $(f_i(y), f_i'(y))$: $\mathscr{A}\mathscr{I}_i$ chooses $a_{i0}, \ldots, a_{it} \overset{R}{\longleftarrow} \mathbb{Z}_p$ and $a_{i0}', \ldots, a_{\text{it}}' \overset{R}{\longleftarrow} \mathbb{Z}_p$ to form the polynomials $f_i(y)$ and $f_i'(y)$ of degree t: $f_i(y) = a_{i0} + a_{i1}y + \cdots + a_{\text{it}}y^t$ and $f_i'(y) = a_{i0}' + a_{i1}'y + \cdots + a_{\text{it}}'y^t$. $\mathscr{A}\mathscr{I}_i$ broadcasts commitment to polynomial coefficients $C_{\text{ik}} = g^{a_{\text{ik}}}h^{a_{\text{it}}'} \bmod p$ for $k \in \{0, \ldots, t\}$. $\mathscr{A}\mathscr{I}_i$ computes $s_{\text{ij}} = f_i(j)$ and $s_{\text{ij}}' = f_i'(j) \bmod q$ for $j \in \{1, \ldots, n\}$ and sends s_{ij} and s_{ij} to $\mathscr{A}\mathscr{I}_j$ to verify. Then, each $\mathscr{A}\mathscr{I}_j$ verifies if

$$g^{a_{\text{ik}}}h^{a_{\text{it}}'} = \prod_{k=0}^{t} (C_{\text{ik}})^{j^k}. \quad (2)$$

If the above equation is not satisfied, $\mathscr{A}\mathscr{I}_j$ will broadcast the complaint against $\mathscr{A}\mathscr{I}_i$. According to the conditions satisfied by the distributed key generation protocol DKG for discrete-log based systems of Gennaro et al. [13], each $\mathscr{A}\mathscr{I}_i$ sets his share of the secret as $x_i = \sum_{j\in \text{QUAL}} s_{\text{ij}} \bmod q$. The distributed secret value x equals $x = \sum_{i\in \text{QUAL}} a_{i0} \bmod q$ from the distributed secret polynomial:

$$F(y) = \sum_{i\in QUAL} a_{i0} + \left(\sum_{i\in QUAL} a_{i1}\right)y + \cdots + \left(\sum_{i\in QUAL} a_{\text{it}}\right)y^t. \quad (3)$$

Vaccine approval (x_i): $\mathscr{A}\mathscr{I}_i$ decides whether to approve the batch of vaccine according to the criteria. If $\mathscr{A}\mathscr{I}_i$ approves it, a signature $\sigma_i = H(\text{vaccine})^{\text{xi}}$ and $\text{pk}_i = g^{x_i}$ are generated and sent to $\mathscr{G}\mathscr{V}$. $\mathscr{G}\mathscr{V}$ verifies the signature by $e(g, \sigma_i) = e(\text{pk}_i, H(\text{vaccine}))$. If the verification passes, $\mathscr{A}\mathscr{I}_i$ is assigned to the set APPR.

Threshold signature (σ_i): if the number of $\mathscr{A}\mathscr{I}_i$s in set APPR is greater than t,

$$LB_j(y) = \prod_{k=0,k!=j}^{t} (y - y_k)/(y_j - y_k), \quad (4)$$

is public Lagrange coefficient for the set APPR according to the Lagrange interpolation method [13].

$$\begin{aligned} x &= \sum_{i\in \text{QUAL}} a_{i0} = \sum_{i\in \text{QUAL}} \left(\sum_{j\in \text{APPR}} \text{LB}_j(0) \cdot s_{\text{ij}}\right) \\ &= \sum_{j\in \text{APPR}} \text{LB}_j(0) \cdot x_j. \end{aligned} \quad (5)$$

According to the above equation, the resulting signature is that $\sigma_{\text{vaccine}} = \prod_{i\in \text{APPR}} (\sigma_i^{\text{LB}_i(0)}) = H(\text{vaccine})^x$ and public key is that $\text{pk} = \prod_{i\in \text{APPR}} (\text{pk}_i^{\text{LB}_i(0)}) = g^x$.

User verification $(\sigma_{\text{vaccine}}, \text{pk}, \text{vaccine})$: the user checks that $e(g, \sigma_{\text{vaccine}}) \overset{?}{=} e(\text{pk}, H(\text{vaccine}))$ for the vaccine. The user accepts the signature if $e(g, \sigma_{\text{vaccine}}) = e(\text{pk}, H(\text{vaccine}))$ holds or rejects it otherwise.

Logistics consignment $(\sigma_{\text{vaccine}}, \text{vaccine})$: structure of vaccine includes the following attributes: ID = H(manufacturer, batch number, serial number), manufacturer, batch number, serial number, vaccine certificate σ_{vaccine}, production date x_p, shelf life x_s, and the duration of immunization x_d. The vaccine manufacturing company broadcasts the vaccine properties, the entrusted logistics company, and the certification certificate $\sigma_{vaccine}$ as a package to the public blockchain.

Cold chain logistics transit $(\sigma_{\text{vaccine}}, \text{vaccine}, \sigma_r)$: the responsible person for the cold chain logistics staging area broadcasts to the public blockchain the vaccine, the vaccine storage environment, its signature σ_r, and the logistics destination package.

Distribution of CDC $(\text{public blockchain}, \text{sk}_{\text{CDC}})$: after checking that the cold chain logistics on the public blockchain meets the standards for transporting biologics, the CDC attaches a signature $\sigma_{C\ DC}$ and broadcasts the distribution to the destination vaccination hospital to the public blockchain.

4.3. Vaccination Record. The framework of the vaccination record system is based on Hyperledger Fabric [14], which is a permissioned blockchain. The privacy protections of the identity of the vaccination hospitals and vaccination users in the vaccine record system are referred to the technique of one-time sender and receiver public key in PAChain [15]. The certificate of authority for the long-term public key (representing the identity of the hospital and the user) of the vaccination hospital and the vaccination user uses the BBS + signature [16] issued by the international joint government. However, in the vaccination record system of this study, the identity of the user and hospital is anonymous to the endorsement node. The endorsement of the vaccination record by the endorsing node uses the anonymous credential technique based on the Boneh-Boyen signature [17]. Vaccination information is encrypted with the auditor's public key using ElGamal encryption [18] to ensure that the information is hidden. If necessary, the auditor can reveal the encrypted vaccination information with his or her secret key. Details are outlined as follows.

(param)$\longleftarrow$ **Setup** (1^{λ}): on input 1^{λ}, where $\lambda \in \mathbb{N}$ is a security parameter. Suppose H_1: $\{0,1\}^{*} \longrightarrow \{0,1\}^{128} \in \mathbb{Z}_p$ and H_2: $\mathbb{G}_1 \longrightarrow \mathbb{Z}_p$ are collision-resistant hash functions. It randomly picks generators $g_0, g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_u, g_h \in G_1, \widehat{g}_1, \widehat{g}_2, \widehat{g}_e \in G_2$.

$(\mathrm{sk}_{\mathrm{vc}}, \mathrm{sk}_{\mathrm{re}}, \mathrm{sk}_{\mathrm{sd}}, \mathrm{pk}_{\mathrm{vc}}, \mathrm{pk}_{\mathrm{re}}, \mathrm{pk}_{\mathrm{sd}})\longleftarrow$ **AuditorKeyGen():** auditor picks random secret keys $\mathrm{sk}_{\mathrm{vc}}, \mathrm{sk}_{\mathrm{re}}, \mathrm{sk}_{\mathrm{sd}} \overset{R}{\longleftarrow} \mathbb{Z}_p$ and outputs their public keys $\mathrm{pk}_{\mathrm{vc}} = g_1^{\mathrm{sk}_{\mathrm{vc}}}, \mathrm{pk}_{\mathrm{re}} = g_2^{\mathrm{sk}_{\mathrm{re}}}, \mathrm{pk}_{\mathrm{sd}} = g_3^{\mathrm{sk}_{\mathrm{sd}}}$.

$(\mathrm{sk}_{\mathrm{CA},U}, \mathrm{pk}_{\mathrm{CA},U}, \mathrm{sk}_{\mathrm{CA},H}, \mathrm{pk}_{\mathrm{CA},H})\longleftarrow$ **CAKeyGen():** CA picks random secret keys $\mathrm{sk}_{\mathrm{CA},U}, \mathrm{sk}_{\mathrm{CA},H} \overset{R}{\longleftarrow} \mathbb{Z}_p$ and outputs their public keys $\mathrm{pk}_{\mathrm{CA},U} = \widehat{g}_1^{\mathrm{sk}_{\mathrm{CA},U}}, \mathrm{pk}_{\mathrm{CA},H} = \widehat{g}_2^{\mathrm{sk}_{\mathrm{CA},H}}$.

$(\mathrm{sk}_e, \mathrm{pk}_e)\longleftarrow$ **EndorserKeyGen():** endorser picks random a secret key $\mathrm{sk}_e \overset{R}{\longleftarrow} \mathbb{Z}_p$ and outputs its public key $\mathrm{pk}_e = \widehat{g}_e^{\mathrm{sk}_e}$.

$(\mathrm{sk}_{U,1}, \mathrm{sk}_{U,2}, \mathrm{pk}_{U,1}, \mathrm{pk}_{U,2})\longleftarrow$ **UserKeyGen():** the user randomly picks a pair of long-term secret keys $\mathrm{sk}_{U,1}, \mathrm{sk}_{U,2} \overset{R}{\longleftarrow} \mathbb{Z}_p$ and computes a pair of long-term public keys $\mathrm{pk}_{U,1} = g_4^{\mathrm{sk}_{U,1}}, \mathrm{pk}_{U,2} = g_4^{\mathrm{sk}_{U,2}}$. $\mathscr{HOSP}$ is also a type of user, so it follows the same algorithm to generate $(\mathrm{sk}_{H,1}, \mathrm{sk}_{H,2}, \mathrm{pk}_{H,1}, \mathrm{pk}_{H,2})$.

$(\mathrm{Cert}_{\mathrm{CA},U})\longleftarrow$ **CACertIssue** $(\mathrm{sk}_{\mathrm{CA}}, \mathrm{pk}_{U,1})$: first, the user needs proof to CA: $\mathrm{PoK}\{(\mathrm{sk}_{U,1})\colon \mathrm{pk}_{U,1} = g^{\mathrm{sk}_{U,1}}\}$. After passing CA verification, CA computes $A_{\mathrm{CA},U} = (g_u \cdot \mathrm{pk}_{U,1} \cdot g_5^{s_u})^{1/(\mathrm{sk}_{\mathrm{CA},U}+\omega_u)}$ using randomly selected $s_u, \omega_u \overset{R}{\longleftarrow} \mathbb{Z}_p$ and its own $\mathrm{sk}_{\mathrm{CA},U}$. Then, CA issues a certificate $\mathrm{Cert}_{\mathrm{CA},U} = \{A_{\mathrm{CA},U}, s_u, \omega_u\}$ to the user's $\mathrm{pk}_{U,1}$. $\mathscr{HOSP}$ is also a type of user, so it follows the same algorithm to generate $(\mathrm{Cert}_{\mathrm{CA},H} = \{A_{\mathrm{CA},H} = (g_h \cdot \mathrm{pk}_{H,1} \cdot g_5^{s_h})^{1/(\mathrm{sk}_{\mathrm{CA},H}+\omega_h)}, s_h, \omega_h\})\longleftarrow$ **CACertIssue** $(\mathrm{sk}_{\mathrm{CA},H}, \mathrm{pk}_{H,1})$.

$(E_i, R_i, E, R_v, \pi_{\mathrm{enc}})\longleftarrow$ **VaccInfoEnc**(vaccination, $\mathrm{pk}_{\mathrm{vc}}$)**:** vaccination information includes ID$=$ H (manufacturer, batch number, serial number), vaccine certificate $\sigma_{\mathrm{vaccine}}$, production date x_p, shelf life x_s, the date of vaccination x_v, and the duration of immunization x_d. Let $M = H_1$(vaccination), and it divides 128-bit M into 8 segments of 16-bit messages m_i by $M = \sum_{i=0}^{7} m_i \cdot (2^{16})^i$. It encrypts each m_i into $E_i = g_0^{m_i}\mathrm{pk}_{\mathrm{vc}}^{r_{v,i}}$ and $R_i = g_6^{r_{v,i}}$, where $r_{v,i} \overset{R}{\longleftarrow} \mathbb{Z}_p$. The encryption E on M can be generated by $E = \prod_{i=0}^{7} E_i^{(2^{16})^i} = g_0^{M}\mathrm{pk}_{\mathrm{vc}}^{r_v}$ and $R_v = g_6^{r_v}$, where $r_v = \sum_{i=0}^{7} r_{v,i} \cdot (2^{16})^i$. The user sends E_i to the auditor. Then, it proves in zero-knowledge proof that the knowledge of $(m_i, r_{v,i})$ and (M, r_v): $\mathrm{PoK}\{(\{m_i, r_{v,i}\}_{i\in\{0,\ldots,7\}}, M, r_v) E_i = g_0^{m_i}\mathrm{pk}_{\mathrm{vc}}^{r_{v,i}} \wedge R_i = g_6^{r_{v,i}} \wedge E = g_0^{M}\mathrm{pk}_{\mathrm{vc}}^{r_v} \wedge R_v = g_6^{r_v}\}$.

Details of the zero-knowledge proof is as follows:

(1) The $\mathscr{HOSP}$ randomly picks $a_i, b_i \in \mathbb{Z}_p$ for $i \in \{0, \ldots, 7\}$ and $a, b \in \mathbb{Z}_p$ and then computes commitments: $C_{v,i} = g_0^{a_i}\mathrm{pk}_{\mathrm{vc}}^{b_i}, D_{v,i} = g_6^{b_i}$ and $C_v = g_0^{a}\mathrm{pk}_{\mathrm{vc}}^{b}, D_v = g_6^{b}$.

(2) It computes $c = H(\{E_i, R_i, C_{v,i}, D_{v,i}\}_{i\in\{0,\ldots\}}, E, R_v, C_v, D_v)$ and for $i \in \{0, \ldots 7\}$ computes challenge response: $z_{1,i} = a_i + \mathrm{cm}_i, z_{2,i} = b_i + \mathrm{cr}_{v,i}$, $z_1 = a + \mathrm{cM}$, $z_2 = b + \mathrm{cr}_v$.

(3) Then, it outputs $\pi_{\mathrm{enc}} = \{\{C_{v,i}, D_{v,i}, z_{1,i}, z_{2,i}\}_{i\in\{0,\ldots 7\}}, C_v, D_v, z_1, z_2, c\}$

$(\mathrm{otpk}_U, R_U)\longleftarrow$ **OTpkGen** $(\mathrm{pk}_{U,1}, \mathrm{pk}_{U,2})$: $\mathscr{HOSP}$ randomly picks $r_u \overset{R}{\longleftarrow} \mathbb{Z}_p$ and outputs $(\mathrm{otpk}_U = \mathrm{pk}_{U,1} \cdot g_4^{H_2(pk_{U,2}^{r_u})}, R_U = g_4^{r_u})$. $\mathscr{HOSP}$ uses the same algorithm to generate $(\mathrm{otpk}_H = \mathrm{pk}_{H,1} \cdot g_4^{H_2(pk_{H,2}^{r_h})}, R_H = g_4^{r_h})\longleftarrow$ **OTpk-Gen** $(\mathrm{pk}_{H,1}, \mathrm{pk}_{H,2})$. $\mathscr{HOSP}$ encrypts user's long-term public key $\mathrm{pk}_{U,1}$ and long-term public key $\mathrm{pk}_{H,1}$ of $\mathscr{HOSP}$ to the auditor by picking random $r_{\mathrm{re}}, r_{\mathrm{sd}} \overset{R}{\longleftarrow} \mathbb{Z}_p$ and computing $(E_{\mathrm{re}} = \mathrm{pk}_{U,1} \cdot \mathrm{pk}_{\mathrm{re}}^{r_{\mathrm{re}}}, R_{\mathrm{re}} = g_2^{r_{\mathrm{re}}})$ and $(E_{\mathrm{sd}} = \mathrm{pk}_{H,1} \cdot \mathrm{pk}_{\mathrm{sd}}^{r_{\mathrm{sd}}}, R_{\mathrm{sd}} = g_3^{r_{\mathrm{sd}}})$. Then, $\mathscr{HOSP}$ runs the following proof of knowledge for ensuring:

(i) $\mathrm{pk}_{U,1}$ and $\mathrm{pk}_{H,1}$ are issued a valid certificate of identity by CA.

(ii) otpk_U is generated by $\mathrm{pk}_{U,1}$. otpk_H is generated by $\mathrm{pk}_{H,1}$. otpk_U is the one-time public key identity of the user whose public key is $\mathrm{pk}_{U,1}$. otpk_H is the one-time public key identity of $\mathscr{HOSP}$ whose public key is $\mathrm{pk}_{H,1}$.

(iii) The user's long-term public key $\mathrm{pk}_{U,1}$ and $\mathscr{HOSP}$'s long-term public key $\mathrm{pk}_{U,1}$ are encrypted by the auditor's public key $\mathrm{pk}_{\mathrm{re}}$ and $\mathrm{pk}_{\mathrm{sd}}$.

$\mathscr{HOSP}$ needs to use proof of knowledge to endorser:

$$
\begin{aligned}
&\mathrm{PoK}\Big\{\big(A_{\mathrm{CA},U}, s_u, \omega_u, \mathrm{pk}_{U,1}, r_{\mathrm{re}}, H_2\big(\mathrm{pk}_{U,2}^{r_u}\big)\big)\colon e\big(A_{\mathrm{CA},U}, \widehat{g}_1^{\omega_u} \cdot \mathrm{pk}_{\mathrm{CA}}\big) = e\big(g_u \cdot \mathrm{pk}_{U,1} \cdot g_5^{s_u}, \widehat{g}_1\big) \wedge \mathrm{otpk}_U \\
&= \mathrm{pk}_{U,1} \cdot g_4^{H_2\left(\mathrm{pk}_{U,2}^{r_u}\right)} \wedge R_U = g_4^{r_u} \wedge E_{\mathrm{re}} = \mathrm{pk}_{U,1} \cdot \mathrm{pk}_{\mathrm{re}}^{r_{\mathrm{re}}}\Big\}.
\end{aligned}
\tag{6}
$$

The details of the zero-knowledge proof is as follows:

(1) $\mathscr{HOSP}$ randomly picks $r_a, r_b, r_c, r_d, r_e, r_\alpha, r_\beta \overset{R}{\longleftarrow} \mathbb{Z}_p$ and makes $\theta = A_{\mathrm{CA},U}^{r_a}$. It computes commitments: $C_{U,1} = e((g_u \cdot E_{\mathrm{re}})^{r_e} g_5^{r_\beta} \theta^{-r_c} \mathrm{pk}_{\mathrm{re}}^{-r_d}, \widehat{g}_1)$, $C_{U,2} = g_{\mathrm{re}}^{r_\alpha}$, $C_{U,3} = R_{\mathrm{re}}^{r_e} g_{\mathrm{re}}^{-r_d}$, $C_{U,4} = \mathrm{pk}_{\mathrm{re}}^{r_\alpha} g_4^{-r_b}$.

(2) It computes challenge $c = H(E_{\mathrm{re}}, R_{\mathrm{re}}, \theta, C_{U,1}, C_{U,2}, C_{U,3}, C_{U,4})$ and computes challenge response: $z_b = r_b + c \cdot H_2(\mathrm{pk}_{U,2}^{r_u}), z_c = r_c + c \cdot \omega_u$, $z_d = r_d + c \cdot r_f r_a$, $z_e = r_e + c \cdot r_a$, $z_\alpha = r_\alpha + c \cdot r_{re}$, $z_\beta = r_\beta + c \cdot r_a s_u$.

(3) It outputs $\pi_{\text{re}} = \{\text{otpk}_U, E_{\text{re}}, R_{\text{re}}, c, \theta, z_b, z_c, z_d, z_e, z_\alpha, z_\beta\}$

Likewise, $\mathscr{HOSP}$ proofs the above relationship to the endorser. The proof process π_{sd} is very similar to that of the user, so it will not be explained in detail here.

$(\text{otsk}_H) \longleftarrow$ **OTskGen** $(\text{sk}_{H,1}, \text{sk}_{H,2}, R_H)$: with $\text{sk}_{H,1}$, $\text{sk}_{H,2}$, and R_H, $\mathscr{HOSP}$ calculates $\text{otsk}_H = \text{sk}_{H,1} + H_2(R_H^{sk_{H,2}})$ and lets $\text{otpk}_H = g_4^{\text{otsk}_H}$. At the same time, $\mathscr{HOSP}$ sends R_U to the vaccination user over a secure channel. The user then generates his own one-time secret key $\text{otsk}_U = \text{sk}_{U,1} + H_2(R_U^{\text{sk}_{U,2}})$.

$(0/1) \longleftarrow$ **EndorserVerify** $(\pi_{\text{re}}, \pi_{\text{sd}}, \pi_{\text{enc}})$: the endorser verifies the legitimacy of the vaccination information and the legitimacy of the one-time public key of the sender ($\mathscr{HOSP}$) and the receiver (user).

The details of the zero-knowledge proof is as follows:

(1) First $\mathscr{HOSP}$ needs proof to endorser: PoK $\{(\text{otsk}_H)\colon \text{otpk}_H = g_4^{\text{otsk}_H}\}$.

(2) On input π_{enc}, for $i \in \{0, \dots, 7\}$, endorser computes $c = H(\{E_i, R_i, C_{v,i}, D_{v,i}\}_{i\in\{0,\dots 7\}}, E, R_v, C_v, D_v)$ and checks $C_{v,i} \overset{?}{=} E_i^c g_0^{z_{1,i}} \text{pk}_{vc}^{z_{2,i}}, D_{v,i} \overset{?}{=} R_i^c g_1^{z_{2,i}}$, $C_v \overset{?}{=} E^c g_0^{z_1} \text{pk}_{\text{vc}}^{z_2}, D_v \overset{?}{=} R_v^c g_1^{z_2}$.

It outputs 1 if the above equation holds or 0 otherwise.

(3) On input π_{re}, endorser computes $C_{U,1}' = e((g_u \cdot E_{\text{re}})^{z_e} g_5^{z_\beta} \theta^{-z_c} \text{pk}_{\text{re}}^{-z_d}, \widehat{g}_1) \cdot e(\theta, \text{pk}_{\text{CA}})^c$, $C_{U,2}' = g_{\text{re}}^{z_\alpha} R_{\text{re}}^{-c}$, $C_{U,3}' = R_{\text{re}}^{z_e} g_{\text{re}}^{-z_d}$, $C_{U,4}' = \text{pk}_{\text{re}}^{z_\alpha} g_4^{-z_b} (\text{otpk}_U / E_{\text{re}})^c$.

Then, endorser computes $c' = H(E_{\text{re}}, R_{\text{re}}, \theta, C_{U,1}', C_{U,2}', C_{U,3}', C_{U,4}')$ and checks $c' \overset{?}{=} c$. It outputs 1 if $c' = c$ holds or 0 otherwise.

(4) On input π_{sd}, endorser does same as (3). The initiator of the vaccine record upload operation can only be the hospital. Therefore, at this step, the endorser needs to verify that the initiator of the upload operation has a valid hospital identification credential.

If all four of the above verifications output 1, then $(1) \longleftarrow$ **EndorserVerify** $(\pi_{\text{re}}, \pi_{\text{sd}}, \pi_{\text{enc}})$.

$(\text{Cert}_e) \longleftarrow$ **EndorserCredIssue** $(\text{otpk}_H, E, \text{sk}_e)$: after verifying the legitimacy of the vaccine information commitment and the legitimacy of the one-time public key of $\mathscr{HOSP}$ and the user, the endorser generates a certificate Cert_e by endorsing the vaccination record $(\text{otpk}_H$ and $E)$. The endorser picks some random $l, k \overset{R}{\longleftarrow} \mathbb{Z}_p$ and uses secret key sk_e to compute $\text{Cert}_e = \{A_e = (g_7 \cdot g_8^l \cdot E \cdot \text{otpk}_H)^{1/(\text{sk}_e + k)}, l, k\}$ to $\mathscr{HOSP}$.

$(1/0) \longleftarrow$ **EndorserCredProof** $(\text{Cert}_e, \text{otsk}_H, M, r_v)$: after obtaining the endorser's certificate Cert_e, $\mathscr{HOSP}$ needs zero-knowledge proof to the verifier that the vaccination record has a valid certificate. First, $\mathscr{HOSP}$ computes the tag $T = f^{\text{otsk}_H}$ for detecting double recording. $\mathscr{HOSP}$ needs to use proof of knowledge to verifier:

$$\text{PoK}\left\{(\text{otsk}_U, M, r_v, A_e, l, k)\colon e\left(A_e, \text{pk}_e \cdot \widehat{g}_e^k\right) = e\left(g_7 \cdot g_8^l \cdot g_0^M \cdot \text{pk}_{\text{vc}}^{r_v} \cdot g_4^{\text{otsk}_H}, \widehat{g}_e\right) \wedge T = f^{\text{otsk}_H}\right\}. \tag{7}$$

The details of the zero-knowledge proof are as follows:

(1) $\mathscr{HOSP}$ randomly picks $r_a, r_b, r_c, r_d, r_e, r_s, r_\alpha, r_\beta \overset{R}{\longleftarrow} \mathbb{Z}_p$ and makes $S_1 = A_e \cdot u_1^{r_a}, S_2 = g_8^{r_a}$. It computes commitments: $C_{e,1} = e(u_1^{r_d} \cdot S_1^{-r_e} \cdot g_8^{r_\beta} \cdot g_0^{r_s} \cdot \text{pk}_{\text{vc}}^{r_\alpha} \cdot g_4^{r_b}, \widehat{g}_e) \cdot e(u_1, \text{pk}_e)$, $C_{e,2} = g_8^{r_c}, C_{e,3} = S_2^{r_e} g_8^{-r_d}, C_{e,4} = f^{r_b}$.

(2) It computes challenge $c = H(T, S_1, S_2, C_{e,1}, C_{e,2}, C_{e,3}, C_{e,4})$ and computes challenge response $z_b = r_b + c \cdot \text{otsk}_H, z_c = r_c + c \cdot r_a$, $z_e = r_e + c \cdot k$, $z_d = r_d + c \cdot r_a \cdot k$, $z_\alpha = r_\alpha + c \cdot r_v$, $z_\beta = r_\beta + c \cdot l$, $z_s = r_s + c \cdot M$.

(3) It outputs $\pi_e = \left\{\text{Cert}_e, c, S_1, S_2, z_b, z_c, z_d, z_e, z_s z_\alpha, z_\beta\right\}$

(4) On input π_e and pk_e, verifier computes $C_{e,1}' = e(u_1^{z_d} S_1^{-z_e} g_8^{z_\beta} g_0^{z_s} \text{pk}_{\text{vc}}^{z_\alpha} g_8^{a_b} g_7^c, g) \cdot e(u_1^{z_c} S_1^{-c}, \text{pk}_e)$, $C_{e,2}' = g_8^{z_c} S_2^{-c}$, $C_{e,3}' = S_2^{z_e} g_8^{-z_d}$, $C_{e,4}' = f^{z_b} T^{-c}$.

Then, verifier computes $c' = H(T, S_1, S_2, C_{e,1}', C_{e,2}', C_{e,3}', C_{e,4}')$ and checks $c' \overset{?}{=} c$. It outputs 1 if $c' = c$ holds or 0 otherwise.

$(1/0) \longleftarrow$ **Link** (T_1, T_2): on input, two vaccination records with two tags T_1, T_2. If $T_1 = T_2$, it outputs 1. Otherwise, it outputs 0.

$(1/0) \longleftarrow$ **Audit** $(E_{\text{re}}, R_{\text{re}}, E_{\text{sd}}, R_{\text{sd}}, \{E_i, R_i\}_{i\in\{0,\dots,7\}})$: on input a ciphertext $(E_{\text{re}}, R_{\text{re}})$, $(E_{\text{sd}}, R_{\text{sd}})$ and $\text{sk}_{\text{re}}, \text{sk}_{\text{sd}}$, auditor has the ability to reveal long-term public keys of users and $\mathscr{HOSP}$ by computing $\text{pk}_{U,1} = E_{\text{re}}/R_{\text{re}}^{\text{sk}_{\text{re}}}, \text{pk}_{H,1} = E_{\text{sd}}/R_{\text{sd}}^{\text{sk}_{\text{sd}}}$. On input a ciphertext $\{E_i, R_i\}_{i\in\{0,\dots,7\}}$ and sk_{vc}, auditor has the ability to reveal vaccination information by computing $g^{m_i} = E_i/R_i^{\text{sk}_{\text{vc}}}$. The auditor uses a precomputation table containing $(g^0, g^1, \dots, g^{(2^{16}-1)})$ to find out the message of m_i and reveal vaccination information $M = (m_7 \| \dots \| m_0)$. The auditor uses the secret keys $\text{sk}_{\text{re}}, \text{sk}_{\text{sd}}$ to reveal the long-term public key $\text{pk}_{H,1} = E_{\text{sd}}/R_{\text{sd}}^{\text{sk}_{\text{sd}}}$ of the vaccination hospital and the long-term public key $\text{pk}_{U,1} = E_{\text{re}}/R_{\text{re}}^{\text{sk}_{\text{re}}}$ of the vaccination user.

4.4. Vaccine Passport.

4.4. Vaccine Passport. The signing of the vaccine passport is accomplished by the vaccination hospital. This process uses ring signature [19] to ensure the anonymity of the vaccination hospital when issuing the authorization. During the presentation of the vaccine passport, the vaccination properties are proven using the Bulletproofs scheme [20] in range proofs to guarantee the validity of the vaccine without exposing the vaccine information. Before using Bulletproofs, it uses interactions to transform the relationships of vaccine attributes into relationships suitable for Bulletproofs range proofs [21]. The identity privacy of the owner of the vaccine passport is protected using the same one-time public key technique as that used to protect the identity of the user in the previous vaccination record system.

After the user received the last vaccination at the hospital, the hospital uploads the vaccination record information to the consortium blockchain. The hospital then issues a vaccine passport to the user.

4.4.1. Vaccine Passport Issue

(1) The user commits the date of vaccination x_v, production date x_p, shelf life x_s, and the duration of immunization x_d by selecting $r_v, r_p, r_s, r_d \overset{R}{\longleftarrow} \mathbb{Z}_p$ and generates commitments $C_v = g^{x_v} h^{r_v}, C_p = g^{x_p} h^{r_p}$, $C_s = g^{x_s} h^{r_s}, C_d = g^{x_d} h^{r_d}$. The user sends $C_v, C_p, C_s, C_d, r_v, r_p, r_s, r_d$ to the vaccination hospital. For the user identity certificate $\mathrm{Cert}_{\mathrm{CA},U} = \{A_{\mathrm{CA},U} = (g_u \cdot \mathrm{pk}_{U,1} \cdot g_5^{s_u})^{1/(\mathrm{sk}_{\mathrm{CA}}+\omega_u)}, s_u, \omega_u\}$ issued by the CA, the user randomly selects $r_a \overset{R}{\longleftarrow} \mathbb{Z}_p$ to send $A_{\mathrm{CA},U}^{r_a}$ to $\mathscr{HOSP}$.

(2) $\mathscr{HOSP}$ receives the user information and opens the commitment and checks:

$$\begin{aligned} C_v &\overset{?}{=} \mathrm{Commit}(x_v', r_v),\\ C_p &\overset{?}{=} \mathrm{Commit}(x_p', r_p),\\ C_s &\overset{?}{=} \mathrm{Commit}(x_s', r_s),\\ C_d &\overset{?}{=} \mathrm{Commit}(x_d', r_d). \end{aligned} \tag{8}$$

If one of the equations does not hold, $\mathscr{HOSP}$ refuses to issue a vaccine passport to its user. Otherwise, $\mathscr{HOSP}$ accepts to issue a vaccine passport for the user.

(3) $\mathscr{HOSP}$ generates a ring signature σ_{ring} for the vaccine passport information $(C_v, C_p, C_s, C_d, A_{\mathrm{CA},U}^{r_a})$. First, it lets $m = H(C_v, C_p, C_s, C_d, A_{\mathrm{CA},U}^{r_a})$ and selects $(n-1)$ public keys of other hospitals. Then, it randomly picks seed $\alpha \overset{R}{\longleftarrow} \mathbb{Z}_p$ and $x_i \overset{R}{\longleftarrow} \mathbb{Z}_p$. Suppose that f is a trapdoor one-way function such as RSA. It computes $y_i = f(x_i, \mathrm{pk}_i)$ and $v_{(i_s+1)} = H(m \| \alpha)$ to go along the ring from signer index i_s. It closes the ring by computing $v_{(i_s)} = H(m \| y_{(i_s-1)} \oplus \cdots \oplus H(m \| y_{(i_s+1)} \oplus H(m \| \alpha)))$ and uses secret key sk_H of signing $\mathscr{HOSP}$ to compute $x_{i_s} = f^{-1}(v_{(i_s)} \oplus \alpha)$. $\mathscr{HOSP}$ randomly selects an index i_0 and outputs the ring signature $\sigma_{\mathrm{ring}} = (i_0, v_{i_0}, x_1, \ldots x_n, \mathrm{pk}_1, \ldots, \mathrm{pk}_n)$.

(4) $\mathscr{HOSP}$ outputs vaccine passport $\{C_v, C_p, C_s, C_d, A_{\mathrm{CA},U}^{r_a}, \sigma_{\mathrm{ring}}\}$

4.4.2. Vaccine Passport Proof

(1) User generates new one-time public and secret keys pair by $(\mathrm{otpk}_U', R_U') \longleftarrow$ **OTpkGen** $(\mathrm{pk}_{U,1}, \mathrm{pk}_{U,2})$ and $(\mathrm{otsk}_U') \longleftarrow$ **OTskGen** $(\mathrm{sk}_{U,1}, \mathrm{sk}_{U,2}, R_U')$. The user needs proof to vaccine passport checkpoint:

$$\mathrm{PoK}\left\{(\mathrm{otsk}_U') : \mathrm{otpk}_U' = g^{\mathrm{otsk}_U'}\right\}, \tag{9}$$

$$\begin{aligned} \mathrm{PoK}\Big\{&\left(A_{\mathrm{CA},U}, s_u, \omega_u, \mathrm{pk}_{U,1}, r_{\mathrm{re}}, H_2(\mathrm{pk}_{U,2}^{r_u})\right) : e\left(A_{\mathrm{CA},U}, g_5^{\omega_u} \cdot \mathrm{pk}_{\mathrm{CA}}\right) = e\left(g_u \cdot \mathrm{pk}_{U,1} \cdot g_5^{s_u}, \widehat{g}_1\right) \wedge \mathrm{otpk}_U'\\ &= \mathrm{pk}_{U,1} \cdot g_4^{H_2((\mathrm{pk}_{U,2}^{r_u}))} \wedge R_U' = g_4^{r_u} \wedge E_{\mathrm{re}} = \mathrm{pk}_{U,1} \cdot \mathrm{pk}_{\mathrm{re}}^{r_{\mathrm{re}}}\Big\}. \end{aligned} \tag{10}$$

(2) Vaccine passport checkpoint verifies the legitimacy of the ring signature σ_{ring}. The verification is straightforward; the vaccine passport checkpoint starts at index i_0 with value v_{i_0}. If $v_{i_0} = H(m \| y_{(i_0-1)} \oplus \cdots \oplus H(m \| v_{i_0} \oplus y_{i_0}))$, it verifies that the vaccine passport has the hospital's valid ring signature.

(3) The vaccine injected by the user is within the shelf life. It requires that the inequality $(x_v - x_p - x_s) > 0$ be satisfied.

The user produces high titers of antibodies to create effective protection. This corresponds to the last date of vaccination plus 14 days [11], which needs to be greater than the current date. It requires that the inequality $(x_v + x_d) < t$ be satisfied, where t is the current date.

The vaccinated user is in the duration of immunization for the vaccine. This is equivalent to the last date of vaccination plus the vaccine immunity lasting time needs to be less than the current date. It requires that the inequality $x_v > (t - 14)$ be satisfied.

$$\begin{aligned} \mathrm{PoK}\Big\{&\left(x_p, x_s, x_d, x_v, r_p, r_s, r_d, r_v\right), \left(C_p, C_s, C_d, C_v\right) : C_p = g^{x_p} h^{r_p}, C_s = g^{x_s} h^{r_s}, C_d = g^{x_d} h^{r_d}, C_v = g^{x_v} h^{r_v}\\ &\wedge \left(x_v - x_p - x_s\right) > 0, \left(x_v + x_d\right) < t1, x_v > (t - 14)\Big\}. \end{aligned} \tag{11}$$

(4) After vaccine passport checkpoint returns $g^t, g^{(t-14)}, g^{2^l}$, the above range proof translates to

$$\begin{aligned}&\mathrm{PoK}\Big\{\big(x_p, x_s, x_d, x_v, r_p, r_s, r_d, r_v\big), \big(A_1 = C_v/\big(C_pC_s\big),\\ &A_2 = C_vC_dg^{2^l}/g^t, A_3 = C_v/g^{(t-14)}\big): A_1 = g^{x_v-x_p-x_s}h^{r_v-r_p-r_s}, A_2 = g^{x_v+x_d-t+2^l}h^{r_v+r_d}, A_3 = g^{x_v-t+14}h^{r_v}\wedge\big(x_v - x_p - x_s\big)\\ &\in \big[0, 2^l\big), \big(x_v + x_d - t + 2^l\big) \in \big[0, 2^l\big), (x_v - t + 14) \in \big[0, 2^l\big)\Big\}.\end{aligned} \tag{12}$$

5. Security Analysis

Definition 1. Threshold signature scheme is called secure robust threshold signature scheme if the following two conditions hold:

(i) Unforgeability: for every PPT adversary A, it is allowed to corrupt up to t participants in the threshold system and is given the oracle channel to ask a finite number of messages m_i and threshold signatures σ_i. Eventually, it forges with negligible probability a valid (m, σ), and m is not in the set of previous queries (m_i, σ_i).

(ii) Robustness: for every PPT adversary A, it is allowed to corrupt up to t participants in the threshold system, and threshold signature protocol runs successfully.

Theorem 1. *(t, n)-threshold signature scheme under the GDH group is a secure threshold signature scheme in the random oracle model against an adversary which is allowed to corrupt any $t < n/2$ participants.*

Definition 2 (Soundness). The vaccination information privacy protocol is sound if for all PPT adversary $\mathscr{A}$ with oracle to query polynomial level times $(E_i, R_i, E, R_v)\longleftarrow$ **VaccInfoEnc** (vaccination, pk_{vc}), and then,

$$\Pr\left[\begin{array}{c}1\longleftarrow\textbf{EndorserVerify}\left(\pi_{\mathrm{enc}}\right):\\ \left(\pi_{\mathrm{enc}}\right)\longleftarrow\mathscr{A}\left(\{E_i, R_i, E, R_v\}\right)\end{array}\right]\leqslant \mathrm{negl}(\lambda). \tag{13}$$

Theorem 2. *The vaccination information privacy protocol is sound if DLP is hard, and the protocol provides knowledge of soundness.*

Proof. It rewinds $c' = H\big(\big\{E_i, R_i, C'_{v,i}, D'_{v,i}\big\}_{i\in\{0,\ldots,7\}}, E, R_v, C'_v, D'_v\big)$, where $c \neq c'$ and computes $\big\{z'_{1,i}, z'_{2,i}\big\}_{i\in\{0,\ldots,7\}}, z'_1, z'_2$. It extracts the knowledge of

$$\begin{aligned} m_i &= \big(z'_{1,i} - z_{1,i}\big)/\left(c' - c\right),\\ r_{v,i} &= \big(z'_{2,i} - z_{2,i}\big)/\left(c' - c\right),\\ M &= \left(z'_1 - z_1\right)/\left(c' - c\right),\\ r_v &= \left(z'_2 - z_2\right)/\left(c' - c\right). \end{aligned} \tag{14}$$

□

Definition 3 (Privacy). The vaccination information is private in the protocol if for all PPT adversary $\mathscr{A}$:

$$\left|\Pr\left[\begin{array}{c} b = b':\\ \left(\{\pi_{\mathrm{enc}}\}_{(0),(1)}\right)\longleftarrow\textbf{VaccInfoEnc}\left(\{\mathrm{vaccination}\}_{(0),(1)}\right),\\ b\overset{R}{\longleftarrow}\{0,1\}, b'\longleftarrow\mathscr{A}\left(\pi_{\mathrm{enc}}\}_{(b)}\right)\end{array}\right] - 1/2\right| \leq \mathrm{negl}(\lambda). \tag{15}$$

Theorem 3. *The vaccination information is private in the protocol if DDH is hard in $\mathbb{G}_1$, and the protocol is HVZK.*

Proof. The encryption $(E_i = g_0^{m_i}\mathrm{pk}_{vc}^{r_{v,i}}, R_i = g_6^{r_{v,i}})$ used in this protocol is the ElGamal encryption algorithm. The security of this encryption is based on the DDH assumption. If the DDH assumption is difficult on $\mathbb{G}_1$, the vaccination information of this protocol is private during transmission.

The simulator of this protocol randomly picks $\big\{E_i, R_i, z_{1,i}, z_{2,i}\big\}_{i\in\{0,\ldots,7\}}, c', z_1, z_2\overset{R}{\longleftarrow}$corresponding domain. Then, it computes

$$\begin{aligned} C_{v,i} &= E_i^{c'}g_0^{z_{1,i}}pk_{vc}^{z_{2,i}},\\ D_{v,i} &= R_i^{c'}g_1^{z_{2,i}},\\ C_v &= E^{c'}g_0^{z_1}\mathrm{pk}_{vc}^{z_2},\\ D_v &= R_v^{c'}g_1^{z_2}, \end{aligned} \tag{16}$$

where they are indistinguishable from real protocol interactions. The simulator sets c' as $H\big(\big\{E_i, R_i, C_{v,i}, D_{v,i}\big\}_{i\in\{0,\ldots,7\}}, E, R_v, C_v, D_v\big)$ in the random oracle model. Therefore, this protocol provides zero-knowledge of vaccination information. □

Definition 4 (Soundness). The users (including hospitals and vaccination users) privacy protocol is sound if for all PPT adversary $\mathscr{A}$ with oracle to query polynomial level times $(\mathrm{Cert}_{CA,U}) \longleftarrow$ **CACertIssue** $(\mathrm{pk}_{U,1})$, and then,

(i) The public key of the user (including hospital and vaccination user) is issued a valid certificate $(A_{CA,U}, s_u, \omega_u)$:

$$\Pr\left[\begin{array}{c} 1 \longleftarrow \mathbf{EndorserVerify}(\pi_{re}(\text{or } \pi_{sd})): \\ \left(\mathrm{Cert}_{CA,U}'\right) \longleftarrow \mathscr{A}\left(\mathrm{pk}_{U,1}\right) \\ \text{where}\left(\mathrm{Cert}_{CA,U}', \mathrm{pk}_{U,1}\right) \notin \text{oracle queries}, \\ (\mathrm{otpk}_U, \pi_{re}(\text{or } \pi_{sd})) \longleftarrow \mathbf{OTpkGen}\left(\mathrm{pk}_{U,1}, \mathrm{pk}_{U,2}\right) \end{array}\right] \leqslant \mathrm{negl}(\lambda). \tag{17}$$

(ii) otpk_U is computed from a public key $\mathrm{pk}_{U,1}$ and the public key $\mathrm{pk}_{U,1}$ is encrypted to the auditor:

$$\Pr\left[\begin{array}{c} 1 \longleftarrow \mathbf{EndorserVerify}(\pi_{re}'(\text{or } \pi_{sd}')): \\ \left(\mathrm{Cert}_{CA,U}\right) \longleftarrow \mathbf{CACertIssue}\left(\mathrm{pk}_{U,1}\right), \\ (\mathrm{otpk}_U', \pi_{re}'(\text{or } \pi_{sd}')) \longleftarrow \mathscr{A}\left(\mathrm{pk}_{U,1}', \mathrm{pk}_{U,2}'\right) \\ \text{where } \mathrm{pk}_{U,1}' \neq \mathrm{pk}_{U,1} \end{array}\right] \leqslant \mathrm{negl}(\lambda). \tag{18}$$

Theorem 4. *The users (including hospitals and vaccination users) privacy protocol is sound if the q-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ in the random oracle model, where q is the maximum number of* ***CACertIssue*** *oracle queries, and the protocol provides knowledge of soundness.*

Proof. It rewinds $c' = H(E_{re}, R_{re}, \theta, C_{U,1}', C_{U,2}', C_{U,3}', C_{U,4}')$, where $c \neq c'$, and computes $z_b', z_c', z_d', z_e', z_\alpha', z_\beta'$. It extracts the knowledge of

$$\begin{aligned} r_a &= (z_e' - z_e)/(c' - c), \\ s_u &= \left(z_\beta' - z_\beta\right)/\left(r_a(c' - c)\right) \\ \omega_u &= (z_c' - z_c)/(c' - c), \\ r_{re} &= (z_\alpha' - z_\alpha)/(c' - c), \\ H_2\left(\mathrm{pk}_{U,2}^{r_u}\right) &= (z_b' - z_b)/(c' - c), \\ \mathrm{pk}_{U,1} &= \mathrm{otpk}/g_4^{H_2\left(\mathrm{pk}_{U,2}^{r_u}\right)} \\ A_{CA,U} &= \theta^{1/r_a}. \end{aligned} \tag{19}$$

BBS + signature is unforgeable against adaptively chosen message attack under the q-SDH assumption. □

Definition 5 (Anonymity). The anonymity of users (including hospitals and vaccination users) is enabled in the protocol if for all PPT adversary $\mathscr{A}$,

$$\left|\Pr\left[\begin{array}{c} b = b': \\ \left(\{\mathrm{otpk}_U, R_U\}_{(0),(1)}\right) \longleftarrow \mathbf{OTpkGen}\left(\mathrm{pk}_{U,1}\right), \\ b \overset{R}{\longleftarrow} \{0,1\}, b' \longleftarrow \mathscr{A}\left(\{\mathrm{otpk}_U, R_U\}_{(b)}\right) \end{array}\right] - 1/2\right| \leqslant \mathrm{negl}(\lambda). \tag{20}$$

Theorem 5. *The anonymity of users (including hospitals and vaccination users) is enabled in the protocol if CDH is hard in $\mathbb{G}_1$, and the protocol is HVZK.*

Proof. The encryptions ($E_{\text{re}} = \text{pk}_{U,1} \cdot \text{pk}_{\text{re}}^{r_{\text{re}}}, R_{\text{re}} = g_2^{r_{\text{re}}}$) and ($E_{\text{sd}} = \text{pk}_{H,1} \cdot \text{pk}_{\text{sd}}^{r_{\text{sd}}}, R_{\text{sd}} = g_3^{r_{\text{sd}}}$) used in this protocol are the ElGamal encryption algorithm. The security of this encryption is based on the DDH assumption. The one-time public key ($\text{otpk}_U = \text{pk}_{U,1} \cdot g^{H_2(pk_{U,2}^{r_u})}, R_U = g_4^{r_u}$) and ($\text{otpk}_H = \text{pk}_{H,1} \cdot g^{H_2(pk_{H,2}^{r_h})}, R_H = g_4^{r_h}$) generation algorithm is based on the CDH assumption. If the CDH assumption is difficult on $\mathbb{G}_1$, the anonymity of users (including hospitals and vaccination users) is enabled during transmission.

The simulator of this protocol randomly picks $c', \theta, z_b, z_c, z_d, z_e, z_\alpha, z_\beta \xleftarrow{R}$ corresponding domain. Then, it computes

$$\begin{aligned} C_{U,1} &= e\left(\left(g_u \cdot E_{\text{re}}\right)^{z_e} g_5^{z_\beta} \theta^{-z_c} \text{pk}_{re}^{-z_d}, \widehat{g}_1\right) \cdot e\left(\theta, \text{pk}_{\text{CA}}\right)^{c}, \\ C_{U,2} &= g_{re}^{z_\alpha} R_{\text{re}}^{-c}, \\ C_{U,3} &= R_{\text{re}}^{z_e} g_{\text{re}}^{-z_d}, \\ C_{U,4} &= \text{pk}_{\text{re}}^{z_\alpha} g_4^{-z_b} \left(\text{otpk}_U / E_{\text{re}}\right)^{c}, \end{aligned} \tag{21}$$

where they are indistinguishable from real protocol interactions. The simulator sets c' as $H(E_{\text{re}}, R_{\text{re}}, \theta, C_{U,1}, C_{U,2}, C_{U,3}, C_{U,4})$ in the random oracle model. Therefore, this protocol provides zero-knowledge of CA certificate for the user's long-term public key and the user's long-term public key. □

Definition 6 (Soundness). The vaccination information endorsement protocol is sound if for all PPT adversary $\mathscr{A}$ with oracle to query polynomial level times $(\text{Cert}_e) \longleftarrow$ **EndorserCertIssue** (otpk_H, E), and then, this vaccination information $E = g_0^M \text{pk}_{\text{vc}}^{r_v}$ and otpk_H is issued a valid certificate (A_e, l, k) by the endorsement nodes:

$$\Pr\left[\begin{array}{c} 1 \longleftarrow \textbf{EndorserCredProof}\left(\text{Cert}_e', \text{otpk}_H, E\right) \vee \\ 1 \longleftarrow \textbf{EndorserCredProof}\left(\text{Cert}_e, \text{otpk}_H', E'\right): \\ \left(\text{Cert}_e'\right) \longleftarrow \mathscr{A}\left(\text{otpk}_H, E\right) \vee \\ \left(\text{Cert}_e\right) \longleftarrow \textbf{EndorserCredIssue}\left(\text{otpk}_H, E\right) \\ \text{where}\left(\text{otpk}_H', E'\right) \neq \left(\text{otpk}_H, E\right) \end{array}\right] \leqslant \text{negl}(\lambda). \tag{22}$$

Theorem 6. *The vaccination information endorsement protocol is sound if the q-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ in the random oracle model, where q is the maximum number of* ***EndorserCredIssue*** *oracle queries, and the protocol provides knowledge of soundness.*

Proof. It rewinds $c' = H(T, S_1, S_2, C_{e,1}', C_{e,2}', C_{e,3}', C_{e,4}')$, where $c \neq c'$, and computes $z_b', z_c', z_d', z_e', z_\alpha', z_\beta', z_s'$. It extracts the knowledge of

$$\begin{aligned} r_a &= \left(z_c' - z_c\right) / \left(c' - c\right), \\ l &= \left(z_\beta' - z_\beta\right) / \left(c' - c\right), \\ k &= \left(z_e' - z_e\right) / \left(c' - c\right), \\ M &= \left(z_s' - z_s\right) / \left(c' - c\right), \\ r_v &= \left(z_\alpha' - z_\alpha\right) / \left(c' - c\right), \\ \text{otsk}_H &= \left(z_b' - z_b\right) / \left(c' - c\right), \\ A_e &= S_1 / u_1^{r_a}. \end{aligned} \tag{23}$$

BBS + signature is unforgeable against adaptively chosen message attack under the q-SDH assumption. □

Definition 7 (Privacy). The vaccination information is private in the protocol if for all PPT adversary $\mathscr{A}$,

$$\left| \Pr\left[\begin{array}{c} b = b': \\ \left(\{\text{Cert}_e\}_{(0),(1)}\right) \longleftarrow \textbf{EndorserCredIssue} \\ \left(\{E, \text{otpk}_H\}_{(0),(1)}\right), \\ b \xleftarrow{R} \{0,1\}, b' \longleftarrow \mathscr{A}\left(\{E, \text{otpk}_H\}_{(b)}\right) \end{array}\right] - 1/2 \right| \leqslant \text{negl}(\lambda). \tag{24}$$

Theorem 7. *The vaccination information is private in the protocol if the protocol is HVZK.*

Proof. The simulator of this protocol randomly picks $\left\{c', S_1, S_2, z_b, z_c, z_d, z_e, z_s z_\alpha, z_\beta\right\} \xleftarrow{R}$ corresponding domain. Then, it computes

$$\begin{aligned} C_{e,1} &= e\left(u_1^{z_d} \cdot S_1^{-z_e} \cdot g_8^{z_\beta} \cdot g_0^{z_s} \cdot \text{pk}_{\text{vc}}^{z_\alpha} \cdot g_8^{a_b} g_7^{c'}, g\right) \\ &\quad \cdot e\left(u_1^{z_c} S_1^{-c'}, \text{pk}_e\right), \\ C_{e,2} &= g_8^{z_c} S_2^{-c'}, \\ C_{e,3} &= S_2^{z_e} g_8^{-z_d}, \\ C_{e,4} &= f^{z_b} T^{-c'}, \end{aligned} \tag{25}$$

where they are indistinguishable from real protocol interactions. The simulator sets c' as $H(T, S_1, S_2, C_{e,1}, C_{e,2}, C_{e,3}, C_{e,4})$ in the random oracle model.

Table 1: Comparison of COVID-19 vaccine systems.

	Blockchain structure	Vaccine supply chain	Vaccination record	User privacy	Vaccination hospital privacy	Vaccination privacy	Auditable	Vaccination certificate
[7]	Public blockchain	√	√	√	×	×	√	×
[10]	Public chain and private blockchain	√	√	√	√	√	√	×
[6]	Public blockchain	√	×	×	×	×	√	×
[8]	Private blockchain	×	√	√	×	√	×	√
[5]	Public blockchain	√	×	×	×	×	√	×
[22]	Consortium ethereum-based blockchain	×	√	√	×	√	×	√
[23]	Consortium blockchain	×	√	√	×	√	?	√
[24]	Permissioned blockchain	×	√	√	√	√	√	√
China health code system	No blockchain	×	√	×	×	×	√	√
This study	Public blockchain and consortium blockchain	√	√	√	√	√	√	√

"√" represents that the privacy protection of this attribute of the vaccine system is based on stronger assumptions, such as storing the private data in a private database off-chain or increasing the restriction of database access, thus having a higher probability of privacy leakage. "?" represents that this attribute of the vaccine system is not mentioned from the open-source code or references.

Therefore, this protocol provides zero-knowledge of vaccination information. □

Lemma 1 (Ring lemma). *Ring signature is unforgeable if the DL assumption holds. The anonymity of the ring signature is unconditional.*

Lemma 2. *The Bulletproof has perfect completeness, perfect special honest verifier zero-knowledge, and computational witness extended emulation.*

6. System Analysis

6.1. Security and Privacy. We compare the vaccine system proposed in this study with other solutions proposed in academia and platform systems that have been applied in practice, as given in Table 1. The main aspects of comparison are the blockchain structure, the domain covered by the system, the properties of user privacy protection, and auditability.

In terms of vaccine system structure, the non-blockchain-based vaccine system is represented by the China health code system, a digital vaccine certificate implemented by the Chinese government based on Alipay, a trusted third party. The authentication of the vaccine certificate is done by the verifier through the QR code in the Alipay wallet app. Another blockchain-based vaccine system mainly takes advantage of the immutability and decentralized property of blockchain to create a more credible and secure vaccine system, which is also the trend of vaccine system research. The main types of blockchains in vaccine systems are public blockchains, private blockchains, and consortium blockchains. In this study and [10], a double-chain structure is used. However, under the assumption of global recognition, the consortium blockchain has an advantage over the private blockchain in terms of use coverage.

In terms of privacy protection, we divide user privacy into user identity privacy, vaccination hospital privacy, and privacy of vaccination records. Systems with a single public blockchain structure, for example [5, 6], are not user privacy protected. The blockchain of vaccination records in [7] keeps sensitive information of users out of the blockchain and protects user privacy to some extent. The [8, 10, 22, 23] schemes use private blockchain or consortium blockchain for participant's identity authentication to protect user privacy. Qiu and Zhu [10] stored all the vaccination records in a private blockchain and Alabdulkarim et al. [24] stored the private data on the private database of the authorized specific peer. However, this does not guarantee the leakage of user vaccination privacy by the nodes in the private blockchain, and the storage of vaccination records would be centralized. In the study by Abid et al. [8], the vaccination certificate is issued by the healthcare provider (issuer) with a signature. Therefore, this process can expose the privacy of the user's vaccination hospital. Also, this scheme cannot audit the vaccination information because it uses a private blockchain and a certain degree of information encryption. Both [22, 23] used consortium blockchains structure, but do not have any encryption of user privacy information, so these two schemes guarantee the privacy of user personal information only to some extent. However, the privacy of vaccination hospitals cannot be guaranteed.

6.2. Performance Analysis. The main objective of this study is to propose a framework for a double-chain-based vaccine passport system and to refine the design of the protocol between specific participants. This goal of this study is to provide a systematic solution to the vaccine passport which focuses more on the theoretical aspect. Therefore, only a qualitative analysis of the system's performance is presented here.

The additional performance overhead of the public blockchain-based vaccine cold chain phase is mainly in the approval phase. For each approval process of vaccines sent for review, the approval institutions in each country need to participate in the distributed setting of threshold secret sharing of the value x. For each distributed key generation protocol, it is assumed that there are n approval institutions, and each institution needs to generate 2 random polynomials. At the same time, each approval authority broadcasts the commitment of polynomials to the other $(n-1)$ approval authorities. The communication data volume of the whole broadcast channel is $O(n^2)$.

The permission blockchain framework for the vaccination record phase of this study is based on the Hyperledger Fabric architecture. By referring to the idea of PAChain [15], the privacy of the vaccination records is protected among the endorsers, orderers, and committing peers. This study removes the trust in the endorser compared to PAChain, thus increasing the authentication protocol. Therefore, the system latency in this phase is slightly higher than PAChain.

The performance bottleneck in the passport identification phase is mainly due to the range proof of the vaccine attributes. Benefiting from the efficiency and aggregability of Bulletproofs [20], the proof size of vaccine passports in the presentation phase is $O(\log(\text{mn}))$ for a batch size of n users and the vaccine attribute length of m bits. For the specific case where the vaccine attribute is 64 bits $(m = 64)$, the proof size for a single user is $3 \times 675 = 2025$ bytes; while, the aggregated proof size for 512 users is $3 \times 1253 = 3759$ bytes.

Based on the results of the above system performance analysis, we believe that the vaccine passport system proposed in this study is feasible for development and implementation. In future implementations, sacrificing acceptable system performance loss in exchange for abundant privacy-preserving security properties is to be considered in advance.

7. Conclusion

This study makes improvements to the vaccine approval part of the previous vaccine distribution and management system. The introduction of a threshold signature scheme in distributed vaccine approval institutions has a certain degree of deterrence against collusive corruption between vaccine approval institutions and vaccine manufacturing companies. Second, the privacy protection in the previous double-chain system is optimized. In this study, the privacy protection of vaccination hospitals, vaccine trusts, and vaccination users is added to the audit function, which increases the controllability and auditability of the vaccination record system in practice. Finally, the vaccine passport proposed in this study protects the privacy of the user's vaccination hospital, the vaccine, and the user's identity while proving the validity and legitimacy of the passport to the vaccine passport checkpoint. Moreover, it is possible to differentiate and adopt targeted measures and policies for different conditions of the vaccine passport. Future work in this study lies in weakening the authority of local vaccination hospitals in the system. It can increase the link between the double chains using corresponding cryptographic techniques.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] W. Tan, X. Zhao, X. Ma et al., "A novel coronavirus genome identified in a cluster of pneumonia cases—wuhan, China 2019- 2020," *China CDC weekly*, vol. 2, no. 4, pp. 61-62, 2020.

[2] "Vaccination certification to be digitized by the end of this year for use in Japan," 2021, https://www.nikkei.com/article/DGXZQOUA264980W1A820C2000000/.

[3] L. Byers, "BC launches proof of vaccination to stop spread of covid-19," 2021, https://news.gov.bc.ca/releases/2021HLTH0053-001659.

[4] M. A. Hall and D. M. Studdert, "Vaccine passport" certification—policy and ethical considerations," *New England Journal of Medicine*, 2021.

[5] L. Cui, Z. Xiao, J. Wang et al., "Improving vaccine safety using blockchain," *ACM Transactions on Internet Technology*, vol. 21, no. 2, pp. 1–24, 2021.

[6] C. Antal, T. Cioara, M. Antal, and I. Anghel, "Blockchain platform for covid-19 vaccine supply management," *IEEE Open Journal of the Computer Society*, vol. 2, pp. 164–178, 2021.

[7] B. Yong, J. Shen, X. Liu, F. Li, H. Chen, and Q. Zhou, "An intelligent blockchain-based system for safe vaccine supply and supervision," *International Journal of Information Management*, vol. 52, Article ID 102024, 2020.

[8] A. Abid, S. Cheikhrouhou, S. Kallel, and M. Jmaiel, "Novidchain: blockchain-based privacy-preserving platform for covid-19 test/vaccine certificates," *Software: Practice and Experience*, 2021.

[9] A. B. Haque, B. Naqvi, A. K. M. N. Islam, and S. Hyrynsalmi, "Towards a gdpr-compliant blockchain-based covid vaccination passport," *Applied Sciences*, vol. 11, no. 13, p. 6132, 2021.

[10] Z. Qiu and Y. Zhu, "A novel structure of blockchain applied in vaccine quality control: double-chain structured blockchain system for vaccine anticounterfeiting and traceability," *Journal of Healthcare Engineering*, vol. 2021, Article ID 6660102, 10 pages, 2021.

[11] F.-C. Zhu, Y.-H. Li, X.-H. Guan et al., "Safety, tolerability, and immunogenicity of a recombinant adenovirus type-5 vectored covid-19 vaccine: a dose-escalation, open-label, non-randomised, first-in-human trial," *The Lancet*, vol. 395, no. 10240, pp. 1845–1854, 2020.

[12] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group

signature scheme," in *Proceedings of the International Workshop on Public Key Cryptography*, pp. 31–46, Springer, Miami, FL, USA, 6 January 2003.

[13] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 295–310, Springer, Prague Czech Republic, 2 May 1999.

[14] E. Androulaki, A. Barger, V. Bortnikov et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, pp. 1–15, ACM, Porto Portugal, 23 April 2018.

[15] T. H. Yuen, "Pachain: private, authenticated & auditable consortium blockchain and its implementation," *Future Generation Computer Systems*, vol. 112, pp. 913–929, 2020.

[16] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-taa," in *Proceedings of the International conference on security and cryptography for networks*, pp. 111–125, Springer, Maiori, Italy, 6 September 2006.

[17] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Proceedings of the International conference on the theory and applications of cryptographic techniques*, pp. 56–73, Springer, Interlaken, Switzerland, 2 May 2004.

[18] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[19] E. Bresson, J. Stern, and M. Szydlo, "Threshold ring signatures and applications to ad-hoc groups," in *Proceedings of the Annual International Cryptology Conference*, pp. 465–480, Springer, Santa Barbara, CA, USA, August 2002.

[20] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: short proofs for confidential transactions and more," in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pp. 315–334, IEEE, San Francisco, CA, USA, 20 May 2018.

[21] J. Camenisch, R. Chaabouni, and abhi shelat, "Efficient protocols for set membership and range proofs," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 234–252, Springer, Melbourne, VIC, Australia, 7 December 2008.

[22] M. Eisenstadt, M. Ramachandran, N. Chowdhury, A. Third, and J. Domingue, "Covid-19 antibody test/vaccination certification: there's an app for that," *IEEE Open Journal of Engineering in Medicine and Biology*, vol. 1, pp. 148–155, 2020.

[23] "A simple and secure certificate of covid-19 immunity," 2020, https://www.immupass.org/.

[24] Y. Alabdulkarim, A. Alameer, M. Almukaynizi, and A. Almaslukh, "Spin: a blockchain-based framework for sharing covid-19 pandemic information across nations," *Applied Sciences*, vol. 11, no. 18, p. 8767, 2021.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 1443978, 17 pages
https://doi.org/10.1155/2022/1443978

Research Article

IoT-DeepSense: Behavioral Security Detection of IoT Devices Based on Firmware Virtualization and Deep Learning

Jin Wang,[1] **Chang Liu**,[1] **Jiangpei Xu**,[1] **Juan Wang**,[2] **Shirong Hao**,[2] **Wenzhe Yi**,[2] **and Jing Zhong**[2]

[1]*Electric Power Research Institute of State Grid Hubei Electric Power Company, Wuhan 430072, Hubei, China*
[2]*School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei, China*

Correspondence should be addressed to Juan Wang; jwang@whu.edu.cn

Received 15 October 2021; Revised 30 January 2022; Accepted 4 February 2022; Published 18 March 2022

Academic Editor: Weizhi Meng

Recently, IoT devices have become the targets of large-scale cyberattacks, and their security issues have been increasingly serious. However, due to the limited memory and battery power of IoT devices, it is hardly possible to install traditional security software, such as antivirus software for security defense. Meanwhile, network-based traffic detection is difficult to obtain the internal behavior states and conduct in-depth security analysis because more and more IoT devices use encrypted traffic. Therefore, how to obtain complex security behaviors and states inside IoT devices and perform security detection and defense is an issue that needs to be solved urgently. Aiming at this issue, we propose IoT-DeepSense, a behavioral security detection system of IoT devices based on firmware virtualization and deep learning. IoT-DeepSense constructs the real operating environment of the IoT device system to capture the fine-grained system behaviors and then leverages an LSTM-based IoT system behavior abnormality detection approach to effectively extract the hidden features of the system's behavior sequence and enforce the security detection of the abnormal behavior of the IoT devices. The design and implementation of IoT-DeepSense are carried out on an independent Internet of things behavior detection server, without modifying the limited resources of IoT devices, and have strong scalability. The evaluation results show that IoT-DeepSense achieves a high behavioral detection rate of 92%, with negligible impact on the performance of IoT devices.

1. Introduction

In recent years, the Internet of things (IoT) has developed rapidly and has been continuously expanded and applied in the fields of smart homes, smart cities, industrial systems, and smart medical products [1]. Statistics show that the number of IoT devices will reach about 50 billion [2] by 2020. These interconnected IoT devices also bring a lot of smart applications, services, and data, resulting in more emerging data-centric businesses [3].

With the rapid growth of the Internet of things applications and devices, attacks against the Internet of Things are becoming more and more serious. For example, remote attackers attack and damage patients' implantable medical devices [4], which may not only cause huge economic losses to individuals but also endanger the safety of patients. Also, with the widespread use of IoT devices in other key areas, attackers may jeopardize public network security. For example, in 2016, a distributed denial-of-service (DDoS) [5] attack against domain name system provider Dyn Company resulted in the inaccessibility of multiple websites such as GitHub and Twitter. This attack is performed through a botnet composed of a large number of IoT devices, including IP cameras, gateways, and even baby monitors. Furthermore, in April 2021, the Mozi botnet that targets IoT devices had controlled approximately 438,000 nodes for DDoS attacks, data exfiltration, and remote command execution. Each compromised node in the botnet is instructed to find a new victim IoT device for infection [6]. Therefore, it is foreseeable that the botnets will continue to expand rapidly and cause more serious damage.

However, most enterprises and users lack privacy and security awareness, and only focus on the realization of the core functions of the product, while ignoring potential security issues. Unless the user initiates a firmware update, IoT device vendors usually do not update and patch their device security vulnerabilities. At the same time, due to limited power consumption and resources, IoT devices are usually unable to install and run traditional security software, such as antivirus software and IDS, resulting in vulnerabilities in IoT devices (e.g., default passwords and unpatched errors) that cannot be eliminated in a long time [7]. Therefore, under the situation where the number of IoT devices is increasing rapidly and security problems are constantly emerging, it is of great significance how to conduct security detection on the behavior of the IoT devices to realize the IoT security defense.

At present, the research on the behavior security of IoT devices mainly focuses on behavior detection schemes based on network traffic. Network traffic-based detection can only detect the security issues of IoT devices from the network layer. It is difficult to obtain the internal behavior status of the system and conduct an in-depth security analysis. Besides, more and more IoT devices currently use encrypted traffic, which also makes traffic-based security detection more difficult.

The firmware virtualization technology simulates the operating environment of the embedded system based on the firmware image, realizes large-scale and automated dynamic analysis of the embedded firmware binary file, and then mines firmware vulnerabilities to implement the embedded firmware security analysis. The firmware virtualization technology can simulate the real operating environment of the firmware, which helps to obtain the operating data and security status of the internal system layer of the device and thus realize the behavior detection and defense at the system level of the IoT device. Therefore, we propose a method to perform system behavior detection on IoT devices using firmware virtualization and combining it with deep learning. Given the difficulty in obtaining complex security behaviors and states inside IoT devices, the system operating environment of IoT devices is simulated based on firmware virtualization technology and then the complex security behaviors and states inside IoT devices are collected. Aiming at the problem that the internal behavior information structure of IoT devices is complex and difficult to detect, a deep learning-based IoT behavior security detection scheme is proposed. In response to the demand for risk mitigation of abnormal behaviors of the Internet of things, the attack stage and security risks are analyzed and determined, and the risk mitigation policies are presented.

In particular, our main contributions can be summarized as follows:

(i) We propose a fine-grained dynamic system behavior capture mechanism to collect the complex security behaviors and states inside the IoT devices in real time.

(ii) We propose an LSTM-based IoT system behavior abnormality detection model. The abnormal behavior detection model can effectively extract the hidden features of the system's behavior sequence and well express the internal dependencies, to successfully implement the security detection of the abnormal behavior of the IoT devices.

(iii) We design and implement IoT-DeepSense, an IoT device behavior security detection system that combines firmware virtualization and deep learning. The evaluation results show that the system can achieve a high behavioral detection rate of 92%, with negligible impact on the performance of IoT devices.

The rest of the study is organized as follows. Section 2 introduces the background, and Section 3 describes our system overview and the design details. The implementation and evaluation of our system are described in Section 4. We present related work in Section 5. Finally, we discuss limitations and future work in Section 6 and conclude this study in Section 7.

2. Background

In this section, we describe the background of firmware virtualization and recurrent neural networks.

2.1. Firmware Virtualization. Recently, IoT device risk vulnerabilities are increasing day by day, often with very serious security consequences. However, due to the limited resources of IoT devices, it is a problem to install security monitoring software on IoT devices to obtain the security states of IoT physical devices and defend them. Network function virtualization (NFV) can detect and defend the security states of IoT devices by constructing virtual security functions. However, network security functions such as virtual IDS and virtual firewall only obtain limited and simple device security status by analyzing network traffic. The firmware virtualization technology can build the operating environment of the IoT device system based on the IoT firmware virtualization, to obtain the complex and comprehensive internal behaviors and states of the IoT device.

Since most IoT devices are not X86/X64 architectures, they cannot be emulated using VMware-like software like standard desktop or server operating systems. However, mainstream virtualization software (e.g., QEMU) can already support firmware virtualization of IoT devices with MIPS or ARM architecture, so that IoT device firmware (including operating systems and applications) can run in a virtualized environment.

QEMU [8] is a fast processor emulator based on dynamic binary conversion. Unlike traditional simulators that interpret the target program by instruction, QEMU converts multiple basic blocks at once. More importantly, QEMU caches the converted blocks and uses block linking to link them together. This keeps the executive program in the code cache most of the time, thereby minimizing the overhead of conversion.

For firmware emulation, in addition to instruction conversion, address space conversion is also very important. QEMU has two execution modes: system mode and user mode. The implementation of address space conversion is different in different execution modes. In system mode, QEMU uses a software memory management unit (MMU) to handle memory access. The software MMU maps the client's memory address (GVA) to the host memory address (HVA). This mapping process is transparent to the guest operating system, so the guest operating system can set the GVA to client physical address (GPA) mapping and handle page faults through the page table interface. QEMU adds GVA-to-GPA conversion logic for each memory access. To speed up the conversion, QEMU uses an address translation cache (TLB) to cache the conversion result. Moreover, to avoid invalid code cache and block linking whenever the address translation changes, all converted blocks are indexed using GPA, and block linking is only performed when two basic blocks are within the same physical page. The mapping from GPA to HVA is done using linear mapping (HVA = GPA + offset). Contrary to system model simulation, in user mode simulation, the host virtual address (HVA) is equal to the client virtual address (GVA) plus a constant offset. Therefore, the conversion in user mode simulation is much faster than the conversion in system mode simulation.

There are already several frameworks that support the use of QEMU to implement an IoT virtualized environment. Avatar achieves this goal by building a hybrid execution environment that includes a processor emulator (QEMU) and actual hardware. Avatar uses an emulator to execute and analyze instructions while passing I/O operations to physical hardware, but Avatar must obtain the physical hardware of the device under test and must manually identify and interact with the debug port on the device interface, reducing the practicability of this architecture. FIRMADYNE added hardware support for IoT firmware in the system mode QEMU. FIRMADYNE supports the popular ARM and MIPS architectures among IoT manufacturers. To obtain hardware support, FIRMADYNE implements a complete simulation system by modifying the kernel and drivers to handle the abnormality of the Internet of things due to a lack of actual hardware. Due to FIRMADYNE's full-system emulation, this study chooses to use FIRMADYNE to virtualize IoT devices with firmware to obtain the real operating environment of IoT devices.

2.2. RNN. Artificial neural network (ANN) [9] was inspired by the biological learning system and loosely modeled its basic functions. The biological learning system is a complex network of interconnected neurons [10]. The most common type of standard neural network is a feedforward neural network. Here, the collection of neurons is organized hierarchically: an input layer, an output layer, and at least one intermediate hidden layer. Feedforward neural networks are limited to static classification tasks. Therefore, they are limited to providing a static mapping between input and output.

To model the time prediction task, we need a dynamic classifier. We need to extend the feedforward neural network to dynamic classification. To obtain this property, we need feedback on the signal of the previous time steps back to the network. These networks with recursive connections for processing sequence data are called recurrent neural networks (RNNs) [11]. The typical structure of RNN includes three layers, namely the input layer, the output layer, and the hidden layer. Because the gradient disappears or the gradient explodes, the RNN is limited to learning about ten-time steps. When we deal with short-term dependencies, recurrent neural networks can work well.

Long short-term memory recurrent neural network (LSTM-RNN) [12] solves the problem of gradient disappearance and gradient explosion during long sequence training. The LSTM network is biologically reasonable to a certain extent and can learn more than 1,000-time steps, depending on the complexity of the network being constructed. In short, LSTM can perform better than ordinary RNN in longer sequences. LSTM has achieved considerable success and has been widely used in many problems, such as natural language processing and speech recognition.

The basic LSTM neural network is composed of an input layer, a hidden LSTM layer, and an output layer. However, this architecture can be extended to deeper LSTMs, where multiple hidden LSTM layers are stacked on top of each other [13]. This is done by taking the output of each LSTM cell and using them as the input of the cell at the corresponding location in the next LSTM layer.

The LSTM [14] network uses a memory unit to replace the basic unit in the hidden layer of the RNN. There are three gates in the memory unit of the LSTM network, including the input gate, the output gate, and the forget gate. Since IoT device behavior information is collected and recorded in chronological order and has temporal characteristics, LSTM can be applied to IoT device behavior analysis scenarios.

3. System Design

3.1. System Overview. Due to the limited computing and storage resources of IoT devices, it is difficult to install monitoring software on real IoT devices to obtain the internal behavior state of the device, thereby performing security state detection and defense. To solve this problem, we have proposed IoT-DeepSense, a behavioral security detection architecture and system for IoT devices based on firmware virtualization and deep learning. Using the firmware virtualization technology, we simulate the real operating environment for each real Internet of things device and then collect the complex security behaviors inside the Internet of Things devices and generate device behavior logs. Through abnormal detection and analysis of the device behavior logs, we can realize the behavioral security detection and defense of Internet of Things devices.

The overall architecture of IoT-DeepSense is shown in Figure 1. The architecture includes a device behavior collection module, an abnormal behavior detection module, and an abnormal behavior analysis and risk mitigation module. The device behavior collection module simulates the real operating environment of the IoT device based on the firmware virtualization technology, then collects the fine-grained complex security behaviors inside the IoT device, and generates device behavior logs. The abnormal behavior detection module performs abnormal behavior

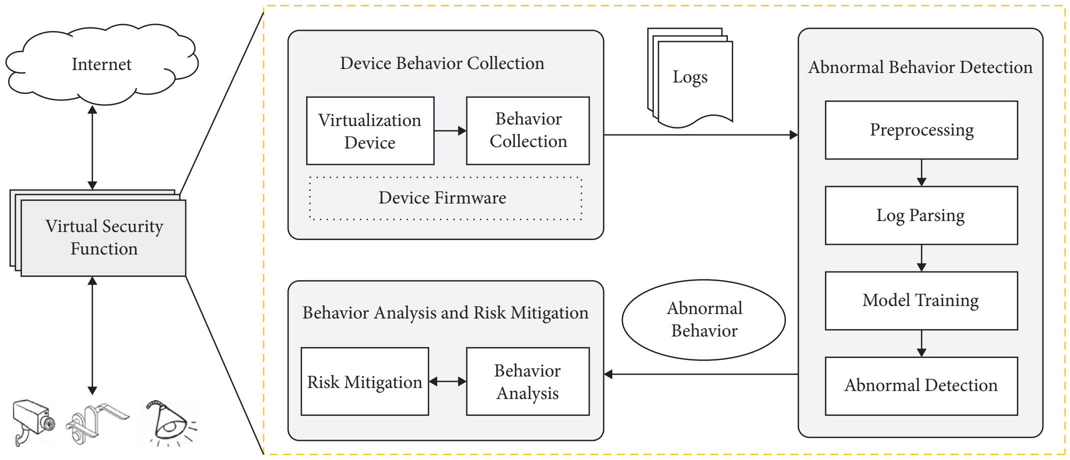


FIGURE 1: System architecture of IoT-DeepSense.

detection on the device behavior log based on deep learning technology and sends the detected abnormal behavior to the behavior analysis and risk mitigation module. The behavior analysis and risk mitigation module analyze the abnormal behavior and propose risk mitigation policies.

3.1.1. Device Behavior Collection Module. This module contains two sub-modules, the virtualization device module and the behavior collection module. (1) Virtualization device module: the virtualization device module uses HTTP or FTP to download the firmware image of the IoT device from the IoT device supplier support website and then uses the firmware virtualization technology to generate the IoT virtual device based on the firmware image, thereby simulating the real operating environment of IoT devices. (2) Behavior collection module: the behavior collection module obtains the complex security behavior and states inside the IoT virtual device in real time through the device log behavior collection script, including a timestamp, process number, process name, system call name, and system call input value and return value (if any), environment variables, and other information. Then, the module records the system behavior in the device behavior log file and sends the device behavior log to the abnormal behavior detection module.

3.1.2. Abnormal Behavior Detection Module. This module performs abnormal behavior detection on device behavior logs based on deep learning. This module includes preprocessing module, log parsing module, model training module, and abnormal detection module. (1) Preprocessing module: the preprocessing module is responsible for data cleaning and the deduplication operations of device behavior logs. (2) Log parsing module: the log parsing module is responsible for converting unstructured device behavior logs into structured device behavior logs, then further extracting meaningful information, and finally generating system behavior event sequences. (3) Model training module: the model training models the system behavior event sequence based on the long- and short-term memory (LSTM) model. (4) Abnormal detection module: for the newly collected unknown system behavior, it will be converted into a sequence of system behavior events through a preprocessing and log parsing module and then predicted using a deep learning model.

3.1.3. Behavior Analysis and Risk Mitigation Module. The module analyzes abnormal behavior to obtain fine-grained abnormal behavior information, such as abnormal file operations and abnormal executable files. The risk mitigation module implements risk mitigation policies including device isolation, traffic filtering, and user notification, which effectively mitigate the security risks brought by abnormal behaviors.

3.2. Dynamic Behavior Collection. Based on firmware virtualization, we have designed a scheme for capturing the dynamic behavior of IoT devices as shown in Figure 2. The purpose of the solution is to collect the system behavior generated during the operation of the IoT system, and hence, we virtualize an IoT virtual device for each IoT real device and simulate the real operating environment of the IoT device, and then, we track and record the security behavior and states generated in the simulated IoT operating environment. The collected behavior log file contains the system behavior in chronological order: each line includes a timestamp, process number, process name, system call name, system call input value and return value (if any), and other information such as environment variables. The chronologically recorded system behavior information allows us to construct various feature vectors to characterize the behavior of IoT devices with different accuracy.

3.2.1. Virtualization Device Based on Firmware. The system virtualizes an IoT virtual device for each IoT real device and then simulates the real operating environment of the IoT device. We collect complex system behaviors and states inside IoT devices in the real operating environment of the Internet of things and generate system behavior logs from the system behaviors and states. The virtualization device is

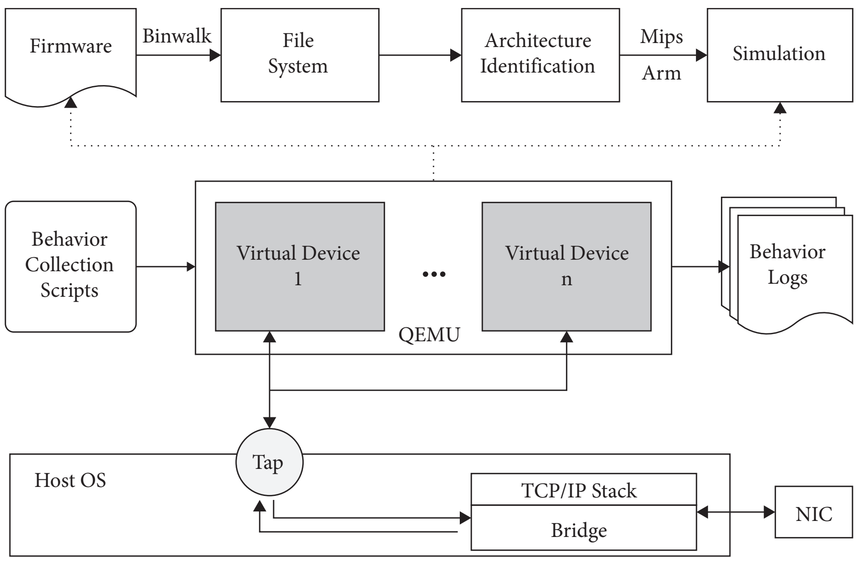


Figure 2: Dynamic behavior capture of IoT devices.

based on FIRMADYNE. FIRMADYNE is an automated dynamic analysis system for Linux-based IoT firmware. In QEMU system mode, hardware support for IoT firmware is added to implement full-system simulation to provide support for popular ARM and MIPS architectures in IoT device firmware. The process of virtualized equipment is mainly divided into four steps.

(1) Download the firmware image. A Web crawler is used to download the firmware image of the target device from the IoT provider website. For dynamic websites that are difficult to automatically crawl, the vendor's FTP website is used. For the collected files, we use Binwalk to check the format of non-firmware files for file filtering. Binwalk is a well-known firmware decompression tool that can extract various data from binary through pattern matching.

(2) Extract the file system. A custom file extraction program is built based on the API of the firmware extraction tool Binwalk, which is used to extract the kernel and root file system. Then, the extracted file system and kernel are compressed into TAR packages and stored for standardization and normalization operations.

(3) Device architecture identification. After extracting the root file system from the firmware image, the system recognizes the architecture and endianness of the target device. Currently, firmware virtualization supports emulating ARM little-endian platforms, MIPS big-endian platforms, and little-endian platforms. For each firmware image, the QEMU system emulator uses the modified kernel that matches its corresponding architecture to guide the file system extracted in the second step.

(4) Simulation. In the simulation phase, the modified kernel tracks and records system calls to the file system, network, and other related kernel subsystems to infer the device system and network configuration. After collecting the information, this information is fed back to our simulation framework to develop a more accurate QEMU configuration for the system. Finally, the system configures the matching network environment to communicate with the simulation firmware.

When configuring the IoT virtual device network, the network TAP device on the host is first instantiated, which is associated with one of the analog network interfaces (such as eth0) in the firmware. For firmware mirroring that uses VLANs, we assign the corresponding VLAN ID to the TAP interface to successfully communicate with the simulated network service. Next, the TAP interface is configured with an IP address that is on the same subnet as the IP address assigned by the firmware to the emulated interface.

This network configuration limits the network functions of IoT virtual devices and prevents them from accessing external networks. Although it can successfully implement automatic dynamic analysis and vulnerability detection of embedded firmware, it cannot make the IoT virtual device perform monitoring tasks. To solve this problem, we add bridges and routes to the IoT virtual device network configuration, so that the IoT virtual device can access the external network. First, a virtual bridge is instantiated on the host. An interface of the virtual bridge is connected to the host's network card (ens33). The host network card serves as one end of the bridge and connects to the external network. The other interface of the bridge is connected to the TAP device. The TAP device serves as the interface at the other end of the bridge and connects to the network interface of the virtual device of the Internet of things, thereby realizing the connection of the two networks at both ends of the bridge. The IP address of the IoT virtual device is modified to correspond to the IP of the real IoT device, and a default route is added.

3.2.2. System Behavior Collection. Virtual IoT devices based on firmware virtualization simulation are all based on Linux.

The system call is an interface provided by the Linux kernel. In IoT devices, the function of system calls mainly includes network communication, creation of new processes, I/O operations, and file operations. Malicious IoT devices and trusted IoT devices have different system behaviors, such as malicious devices requesting more permission to frequently access sensitive resources or frequent I/O operations, resulting in different system call frequencies. In addition, because the system behavior of an IoT device usually involves multiple system calls in sequence, a single system call cannot be considered independent. The system call sequence reveals the dynamic behavior of the application, and different system call sequences reflect different behaviors. Therefore, the process information and the frequency, sequence, and parameters of system calls during the operation of the IoT virtual device can all characterize its dynamic behavior. The following introduces the four types of dynamic characteristics of process information and system call frequency, sequence, and parameters in detail.

(1) Process Information. When the Internet of things device is running, the process information inside the system will reflect the system-related behavior information. For example, when the IoT device becomes the Bot host controlled by the Mirai botnet, the IoT device will have an additional malicious process during its operation (e.g., mirai.mips, where mips indicates that the IoT device is based on a MIPS architecture). The process communicates with the C&C server and operates according to malicious commands issued by the server. Therefore, whether the IoT device is attacked by malware can be detected through the process information. This shows that process information can be used as an important feature of behavior detection.

(2) System Call Frequency. The system call frequency represents the number of occurrences of each type of system call when the Internet of things device runs within a specific time. The frequency of system calls that occur during the execution of the Internet of things system carries information about related behaviors. When an IoT device is maliciously attacked, the device may use some specific system calls more frequently than normal. For example, under DDoS attacks, system calls related to I/O operations made by IoT devices may be much more frequent than normal. This indicates that the increased frequency of system calls may be a sign of malicious behavior.

(3) System Call Sequence. The system call sequence describes the local time relationship between system calls within a limited time range. System calls are fine-grained operating system information. The system call sequence reflects the execution path of the Internet of things device during operation. Different system call sequences reflect different behaviors. Because the functions of IoT devices are relatively fixed and single, the system call sequence has a certain regularity. For example, Table 1 shows the system call sequence information generated by the execution flow of motion [15]. Motion is a highly configurable program for monitoring video signals from multiple types of cameras. The main loop consists of a series of blocks. Each cycle begins with the camera capturing image frames. When motion is detected, the current motion frame is saved to the file system. Then, the application also independently saves snapshots at regular intervals (e.g., every 5 seconds). The two modules, save motion frame and save snapshot, use the same program flow to save the image to a file, thereby generating the same system call sequence as shown in the table.

After that, some predefined functions (e.g., waiting for the camera client to connect) may trigger external programs. This main loop repeats at the specified frame rate (e.g., 3 frames per second). Depending on the function, some blocks may not be executed in every cycle. When the system is maliciously attacked, it will destroy the original regular system call sequence. Therefore, the system call sequence can be used to extract features for malicious behavior detection.

(4) System Call Parameters. Some abnormal behaviors of IoT devices are not directly reflected in the execution path of the device, but in the form of abnormal parameters. For example, the attacker forged a small piece of code to make the system call sequence the same as the normal system call sequence [16], but by modifying the file path parameters, the device information was leaked. As shown in Table 2, the current motion frame is leaked to the desired location in the file system, while the generated system call sequence still looks legitimate. We only modify the location of the saved file (FilePath = "/path/to/Ideal location for attackers"), which makes the motion frame data leaked. Since the code blocks we added to use the same program routines as the "Save Motion Frame" and "Save Snapshot" code blocks, the system call sequence patterns generated by these three code blocks are the same. At the same time, the cross-block conversion does not generate a new pattern. If only one legal block is executed (save motion frames), the resulting sequence is still legal, because the inserted block (leaked motion frame) looks like another unexecuted block (save snapshot). The only way the system call sequence can detect such malicious execution is to learn the time relationship between two legal blocks through a pattern long enough. However, since the image size may vary greatly, this detection method is highly unlikely. Therefore, we need to obtain the specific location of the read-write file and the file name and other information from the system call parameters, to accurately detect abnormal behavior of the Internet of things.

Besides the above four types of dynamic characteristics including process information and system call frequency, sequence, and parameters, other hidden behavior features of IoT systems need to be mined to achieve more accurate IoT behavior security detection. The method for obtaining fine-grained system behavior information executed by the IoT virtual device in firmware is as follows.

When IoT virtual devices use QEMU for software simulation, to realize full software simulation, a custom pre-built kernel for ARM and MIPS architecture is used to replace the kernel extracted from the firmware image. During the custom pre-built kernel compilation process, an analysis module was added to the custom Linux kernel

Table 1: System call sequence.

Time	Function	System call sequence
T1	Get time	gettimeofday–gettimeofday
T2	Get frame	(ioctl)–rt sigprocmask–ioctl–ioctl–rt sigprocmask
T3	Get sport frame	open–fstat64–mmap2–write– ... –write–close–munmap–clone–write
T4	Save snapshot	open–fstat64–mmap2–write– ... –write–close–munmap–clone–write unlink–symlink
T5	Wait for connection	Select (accept–ioctl–write)–(write–munmap–close)–(mmap2–gettimeofday)
T6	Frame rate control	gettimeofday–(nanosleep)

Table 2: Image leakage under the same system call sequence.

Time	Function	System call sequence
T1	Save sport frame	open–fstat64–mmap2–write– ... –write–close–munmap–clone–write
T2	Leak sport frame	open–fstat64–mmap2–write– ... –write–close–munmap–clone–write
T3	Save snapshot	open–fstat64–mmap2–write– ... –write–close–munmap–clone–write

module, which uses the kernel dynamic probe (kprobes) framework to track multiple system calls, thereby helping to debug and simulate the original IoT system environment. Operations such as assigning MAC addresses, creating bridges, restarting the system, and executing programs are all monitored by the firmware virtualization framework to properly configure the simulated network environment.

Since the modified kernel can intercept system calls to the file system, network, and other related kernel subsystems, we can obtain the system call information of all processes executed in the firmware by setting the relevant mask of the kernel system call parameters.

By setting the corresponding bits for these system calls, each time the system is called, the system records the information about the system calls in the system behavior log. We can set the corresponding bit in the firmadyne.syscall parameter at startup to transfer information so that the IoT virtual device outputs the system behavior during the system operation and records it in the system behavior log file. The IoT dynamic behavior capture solution saves the collected IoT system behavior records into behavior log files, which contain chronological system behavior. Each line entry is a system behavior, including a timestamp, process number, process name, system call name, system call input, return value (if any), environment variables, and other information. Recording system behavior information in chronological order enables us to construct various complex features, characterize various behaviors of IoT devices with different precisions, and provide a basis for implementing abnormal behavior detection.

3.3. Abnormal Behavior Detection

3.3. Abnormal Behavior Detection. The fine-grained behavior logs of IoT devices record the system behavior of IoT devices when they are running. Therefore, behavior logs are one of the most valuable data sources for anomaly detection. In addition, due to the complexity and a huge number of behavior logs, manual detection of abnormal behavior becomes infeasible. The keyword matching method based on explicit keywords (e.g., "Error") and the regular expression method based on structural features can only detect a single abnormal behavior log, and it is difficult to detect most behavior log anomalies. These anomalies can only be inferred based on their behavior log sequence, which contains multiple behavior logs that violate conventional rules. Therefore, we need an automatic anomaly detection method based on the system behavior log sequence.

As shown in Figure 3, we propose a deep learning-based abnormal behavior detection scheme for real-time abnormal detection of the behavior logs of the IoT system in this study. The system behavior collected during the normal operation of the Internet of things system has a certain periodicity and stability. The solution is based on the system behavior logs collected by the dynamic behavior capture module during the normal operation of the IoT device, and the deep learning method is used to learn the effective behavior characteristics of the IoT system from a large number of fine-grained system behaviors, thereby realizing the abnormal behavior of the IoT real-time detection without any modification to the existing infrastructure.

The abnormal behavior detection scheme based on deep learning mainly includes two stages, the training stage and the detection stage. During the training phase, the system preprocesses the IoT behavior log files to obtain the normal system behavior execution flow. Then, through the log parsing module, the unstructured behavior log is parsed into a sequence of system behavior events. The system behavior event sequence is used as the input of the abnormal behavior detection model, which learns the complex features of the system behavior event sequence and then constructs the abnormal behavior detection model. In the real-time detection phase, the system generates an IoT system behavior log at that moment and then generates a new system behavior sequence through preprocessing. Through log analysis, the system behavior sequence is parsed into a system behavior event sequence. Finally, the trained abnormal behavior detection model is used to detect whether the system behavior is normal. If it is detected as abnormal behavior, the system behavior will be sent to the abnormal behavior analysis module for further analysis.

3.3.1. Preprocessing and Log Parsing. To collect the data set, we simulated the normal usage of IoT devices. Our data

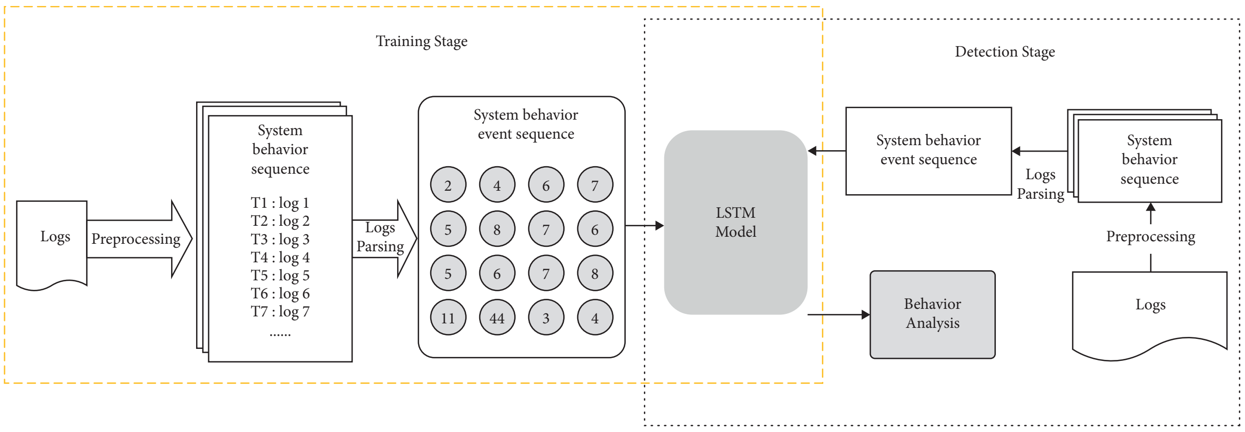


Figure 3: Anomaly detection scheme based on deep learning.

collection method is as follows: firstly, the IoT device is started and allowed to perform any initial configuration or firmware update. Secondly, when the IoT device is in a stable state, we interact with the device through smart applications. We also provide some idle time for the device to communicate without user intervention. According to the activity of the device, we captured tens of thousands of system behaviors from each device and recorded them in the system behavior log file.

For the characteristics of the system behavior log collected based on firmware virtualization in this study, we preprepared it from the following aspects to improve the efficiency of log analysis.

(1) System behavior cleaning. In the system behavior log, many system behaviors only contain constant strings, and there are no internal parameters. The repeated occurrence of these system behaviors will result in a large number of duplicate messages in the system behavior log. Many system behavior log messages will appear repeatedly at the same time. We delete these types of system behaviors to reduce data redundancy, thereby greatly improving the efficiency of subsequent log parsing.
(2) Delete information related to firmware simulation. During the simulation process of the IoT virtual device, system behavior information related to the device simulation process will appear in the behavior log. This information can reflect the system behavior of the device during the simulation, but it cannot reflect the system behavior of the device during operation. We delete the system behavior information related to the equipment simulation process to reduce the impact on the analysis of system operation behavior.

In general, the behavior log records the fine-grained system behavior of the IoT device, and the current security status information of the IoT device can be obtained by analyzing the behavior log. System behavior log files are unstructured text, which increases the difficulty of analyzing system behavior. To realize automated system behavior analysis, the system behavior log must be analyzed first, to parse the original system behavior log data into a system behavior event sequence. Therefore, we first structure the unstructured system behavior log into several parts (e.g., date, time, and content), then extract meaningful information from these parts, and finally generate a sequence of system behavior events.

To achieve the goal of automated log analysis, academia and industry have proposed many data-driven methods, including frequent pattern mining (LogCluster [17]), iterative partitioning (IPLoM [18]), layering clustering (LKE [19]), longest common subsequence calculation (Spell [20]), and parse tree (Drain [21]). These log analysis methods can automatically generate common event templates based on log data. Log parsing should make full use of the inherent structure and characteristics of log messages to obtain good parsing accuracy, instead of directly applying standard algorithms such as clustering and frequent pattern mining [22]. Among several parsing methods, Drain uses a fixed-depth tree structure to represent log messages and effectively extract event templates. This method uses the characteristics of logs and performs well in many log parsing situations.

Due to the complex structure of the behavior log and rich event templates, the above methods still cannot accurately parse the behavior log, and considering the accuracy, robustness, and efficiency of several log analysis methods, we further improved Drain to implement the system behavior log analysis. The following describes the analysis steps in detail.

During log parsing, an unstructured system behavior log file containing free text log messages is used as input. The unstructured system behavior logs we collected consist of a constant part and a variable part. The constant section displays the event template of the log message and remains unchanged for each log event. The variable section shows the parameters of the system during dynamic operation, which may change between different events. For the characteristics of the behavior log format, a regular expression function is written to split the log message. We structure the behavior log message into headers with Time, Syscall, PID, and Content.

When the log message is parsed, a structured system behavior log and a system behavior event template with summary event counts are output. The content of structured system behavior in the structured system behavior log file is shown in Table 3. System behavior is structured to include event ID, PID information, system calls, event templates, event template-related event behavior list, and other system behavior content. The content of the system behavior event template file includes the event ID, the content of the event template, and the number of occurrences of the event template in the entire system behavior log file.

To distinguish each system behavior, we generate an identifier EventID for each system behavior. We use UTF8 to encode the important content of the structured system behavior (PID name, system call, event template, and event parameter list) and then use MD5 to encrypt the encoded content into a string of 8-bit length, and finally, for each encrypted character strings are mapped to one-to-one decimal numbers as EventID of each system behavior.

The system processes the structured system behavior log file to output the sequence of corresponding system behavior events for each PID. As shown in Table 4, the sequence of system behavior events executed by the process with PID 623 is 4, 5, 6, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15. Only three system behavior event sequences are illustrated in the table, and all system behaviors in the system behavior log are similarly operated. Finally, the system behavior event sequence is processed, PID-related information is deleted, and the system behavior event sequence is generated as shown in Table 5. The system behavior event sequence represents a collection of system behavior events executed by an IoT system program, and the system uses it as an input of an anomaly detection model.

Table 3: Example of structured system behavior.

EventID	22
Syscall	Sys socket
PID	372
PIDName	snmpd
EventTemplate	Family
ParameterList	[“2”,”1”,”0”]

Table 4: PID and system behavior event sequence.

PID	Event sequence
623	4,5,6,6,7,8,9,10,11,12,13,14,15
647	3,2,5,6,2,2,2,4
649	1,2,3,2,6,6,6,8,9,6,2,4,5,6,3

Table 5: System behavior event sequence.

4 5 6 6 7 8 9 10 11 12 13 14 15
3 2 5 6 2 2 2 4
1 2 3 2 6 6 6 8 9 6 2 4 5 6 3

3.3.2. Model Training. The log parsing process generates a collection of system behaviors representing the execution of IoT system programs. Once the system behavior log entries are parsed into a sequence of system behavior events, the sequence of system behavior events will reflect a specific execution order execution path. Each data in this data set are mapped to each system behavior one by one.

The collection $S = \{s_1, s_2, s_3, s_4,\ldots,s_n\}$ is used to represent the system behavior sequence data set, and s_i represents the system behavior at position i in the current system behavior sequence. Obviously, s_i is one of the n possible system behaviors in the set S and is strongly dependent on the latest system behavior collection that appeared before s_i. Therefore, we model anomaly detection in the system behavior sequence as a multi-class behavior classification problem, where each different system behavior defines a class. We trained the anomaly detection model as a multi-class classifier in the latest context. The input is the historical record of recent system behavior, and the output is the probability distribution of n system behaviors in the system behavior set S, indicating the probability that the next system behavior in the sequence is $s_i \in S(i = 1, 2,\ldots, n)$.

Suppose t is the sequence ID of the next system behavior to be displayed. The input used for classification is the window w of the h most recent system behaviors, i.e., $w = \{s_{t-h}, s_{t-h+1}, \ldots, s_{t-2}, s_{t-1}\}$, where each s_i in S is the system behavior contained in the system behavior set S. Due to the repeatability of the system behavior, the same system behavior may appear several times in the window w.

The training phase relies on the system behavior generated by the normal execution of the IoT device system. For each system behavior log sequence of length h in the training data, the system update uses $s_i \in S(i = 1,2,\ldots, n)$ as the probability distribution of the next system behavior. For example, suppose that a small amount of system behavior logs generated by normal execution are parsed into a series of system behavior events: $\{s_5, s_8, s_{11}, s_2, s_{26}, s_6, s_3\}$. Assuming that the window size h is 4, the input sequence and output label pairs of the training model will be $\{s_5, s_8, s_{11}, s_2 \longrightarrow s_{26}\}$, $\{s_8, s_{11}, s_2, s_{26} \longrightarrow s_6\}$, and $\{s_{11}, s_2, s_{26}, s_6 \longrightarrow s_3\}$.

The output of the training phase is the conditional probability distribution $P(s_t = s_i \mid < s_{t-h+1},\ldots,s_{t-2},s_{t-1}>)$ (for each system behavior, $s_i \in S(i = 1,2,\ldots,n)$). The detection stage uses this model to predict and compare the predicted system behavior with the actual system behavior.

Inspired by recent research, we found that the abnormal behavior detection of the Internet of things in this study is naturally applicable to the LSTM-RNN model. The system behaviors generated by IoT devices can be organized in chronological order. Our anomaly detection model based on LSTM is shown in Figure 4, where the purpose and parameter settings of each layer are as follows.

(1) Input layer: after preprocessing and log analysis, the normal system behavior will be converted into a system behavior event sequence. We will use the parsed normal system behavior event sequence as the input layer data. Then, we group h consecutive system behaviors to form a system behavior window. Using the system behavior window, we can model the temporal relationship of the behavior of adjacent systems.

(2) LSTM layer: the input layer inputs system behavior to the LSTM layer. In each step, system behavior is

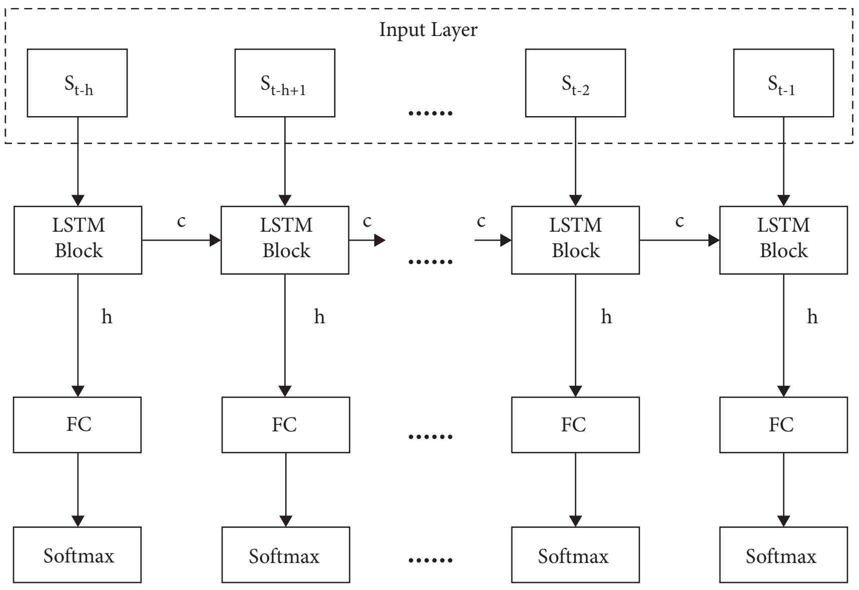


Figure 4: Behavior detection model based on LSTM.

assigned to the LSTM unit. In a single-layer LSTM, the output of the LSTM cell includes the cell state and the hidden state, and the hidden state and the cell state of the LSTM cell are transmitted to the next LSTM cell. Taking 2-layer LSTM as an example, the hidden state of each LSTM cell is also transferred to the stacked lower LSTM cell as its input.

(3) FC layer: we have placed a hidden full-connection (FC) layer between the LSTM layer and the softmax layer, whose size is equal to the number of system behavior categories.

(4) Softmax layer: we output the fully connected layer to the softmax layer for standardization. The output of the softmax layer is a probability distribution for each system behavior, and the probability distribution indicates how likely each system behavior becomes the next system behavior.

The loss function in training is categorical cross-entropy. The optimizer is Adam. In addition, the size of the epoch is 50. After training 50 epochs, the neural network converges.

3.3.3. Abnormal Behavior Detection. Our LSTM-based abnormal behavior detection model can learn the comprehensive and complex associations and patterns contained in a series of system behaviors generated by the execution path of the IoT virtual device system. In the process of abnormal behavior detection, we assume that the IoT virtual device is trusted, the data and operating environment in the IoT virtual device are ensured to be trusted and safe, and the system behavior collected based on the IoT virtual device is also trusted, and the attacker cannot carry out the attack and change the information of the system. For example, Intel SGX [23] technology can be used to provide data storage, data transmission, and run-time security protection for IoT virtual devices.

In summary, our system can detect attacks that cause abnormal behavior in the system's behavior sequence, which can lead to abnormal behavior in the system behavior log. For example, a password blasting attack may perform Telnet or SSH login operations for password verification multiple times, which is reflected in the system behavior log. The category system behavior will increase dramatically; for example, malware will download malware on the Internet of things device and gain control of IoT devices, which is reflected in the malware process in the device system. The malware process will appear to delete files, change network configuration, and other malicious operations, resulting in abnormal system behavior sequences. For example, some attacks will leave traces in the system behavior log. The attack may cause the IoT system to stop working, so the corresponding system behavior sequence ends early or abnormal system behavior occurs.

To realize the real-time abnormality detection of the system behavior of the IoT device, the system will collect the system behavior regularly (in seconds). Then, preprocessing, log analysis, and other processes are carried out, and finally, the system behavior set at this stage is obtained. Behavior $S_i \in S(i = 1,2,\ldots, n)$ is normal or abnormal, $w = \{s_{t-h}, s_{t-h+1}, \ldots, s_{t-2}, s_{t-1}\}$is sent to the LSTM-based abnormal behavior detection model as its input, and finally the model outputs the probability distribution of each system behavior: $P(s_t = s_i|w) = \{s_1{:}p_1, s_2{:}p_2, \ldots, s_n{:}p_n\}$.

This probability distribution describes the probability that each system behavior from the system behavior set S becomes the next system behavior when based on normal IoT system behavior. In fact, S_i may be multiple system behaviors. For example, if the camera and the server are in communication, the system behavior S_i of the camera may be "send video information to the server" or "receive configuration information of the server." In different behavioral contexts, both are normal system behaviors.

The abnormal behavior detection model based on LSTM can learn information about the behavior of multiple systems that can be the next system behavior in the model training phase. In the detection phase, we sort them according to the probability of the possible system behavior Si, and if the value of system behavior is among the first *p* candidate values, they are regarded as normal values. Otherwise, the system behavior is marked as abnormal execution, and then, the abnormal behavior is provided to the abnormal behavior analysis and risk mitigation module for subsequent analysis and other steps.

3.4. Behavior Analysis and Risk Mitigation. IoT attacks are generally divided into multiple stages, and attacks at different stages will cause different device abnormal behavior. When the behavior analysis module obtains the abnormal behavior, it first obtains fine-grained abnormal behavior content from the structured system behavior log file according to the behavior event ID. According to the content of abnormal behavior, different risk mitigation policies are performed.

We propose a general risk mitigation method that is very flexible in different situations. The key idea is that we design a virtual security function for each real IoT device. This virtual security function performs device monitoring tasks, performs abnormal behavior detection, and manages the communication between the IoT real device and the Internet. This module is also responsible for generating and enforcing restricted network access for connected devices. Based on the analysis results of abnormal behavior, it blocks malicious access in real time to reduce the risk of abnormal behavior of the Internet of things. At the same time, the virtual security function also has an isolation function, which controls the access of IoT devices to other IoT devices in the intranet, to prevent one device from being compromised and endangering other devices in the same network segment. This can potentially prevent vulnerable IoT devices from being maliciously triggered by accidental access traffic while ensuring that the system can promptly warn administrators of risks. We propose the following risk mitigation policy, which aims to maintain as many IoT device functions as possible while minimizing security risks.

3.4.1. Network Isolation. The goal of network isolation is to prevent IoT devices with abnormal system behavior from communicating with other devices. To this end, the virtual security function divides the user's network into two virtual networks: an untrusted network and a trusted network. When the IoT device is running, the behavior security detection system will detect the behavior of the IoT device system in real time. The required network isolation level is determined based on the abnormal detection results. IoT devices with normal system behavior have been placed in a trusted network. IoT devices that have detected abnormal behavior are placed in an untrusted network and are strictly isolated from other devices.

According to the different risks caused by abnormal device behavior, we have designed three different isolation levels for IoT devices. (1) Strict isolation level: the IoT device is not allowed to communicate with other IoT devices, and the device does not have Internet access rights. (2) Limited isolation level: this allows devices to communicate with other devices in untrusted network coverage and a limited set of remote targets on the Internet (e.g., cloud services from vendors). (3) Trusted isolation level: this allows the device to communicate with other devices in the trusted network coverage and unrestricted Internet access. Network isolation at the device level granularity ensures that when threatened, any vulnerable device cannot infect other devices in the trusted network. The virtual security function can intercept the traffic in the network and ensure that it is filtered according to the required isolation level.

3.4.2. Flow Filtering. The goal of traffic filtering is to prevent attackers from communicating with vulnerable devices or exposing data. Traffic filtering is performed by virtual security functions and can target specific protocols or endpoints to minimize the impact of the functions of the affected devices.

3.4.3. User Notification. In some cases, network isolation and traffic filtering are insufficient to provide adequate protection. For example, if the IoT device has been attacked by malware, the malware prevents the device from automatically restarting the device to delete the malware in memory. In this case, the effective measure to protect the security of IoT devices is to manually restart or delete the devices at risk. Therefore, the purpose of the user notification is to help the user identify the problematic device, then warn the user that there is a device that cannot overcome the security vulnerability, and ensure that the user restarts or deletes it.

4. Implementation and Evaluation

4.1. Implementation. We evaluate the performance of IoT-DeepSense using the physical machine with an Intel i7-6700 CPU. The machine has 8 GB of memory, 500 G of hard disk, and runs Ubuntu 16.04.

We build a firmware virtualization environment based on FIRMADYNE. We have successfully conducted device simulations and tested multiple firmware versions of NETHEAR WNAP320, WNDAP350, WNDAP360, and WNAP210 series devices. The behavior detection model of IoT-DeepSense is implemented based on PyTorch.

4.2. Data Set. We simulate a normal and several attacked IoT environment, capture the system behaviors separately, and then use the collected system behaviors as the data set of the following experiment. The types of attacks we simulate mainly include IoT malware (Mirai and BrickerBot), DDoS (TFN), and password cracking (Hydra). Mirai is a classic DDoS attack, and BrickerBot is a permanent denial-of-service attack. TFN is an open-source DDoS tool that can perform various DDoS attacks, such as ICMP flooding, SYN

Table 6: System behaviors in our data set.

Type	Normal	Mirai	BrickerBot	Hydra	TFN	Total
Number of system behaviors	42600	4830	4800	2722	1756	56708
Data size	2.9 M	350 K	347 K	200 K	128 K	3.9 M

flooding, UDP flooding, and Smurf attacks. Hydra is an open-source tool for brute force password cracking that provides options for attacking login names of various protocols. Table 6 shows the type and the corresponding number of system behaviors in our data set.

4.3. Functional Evaluation

4.3.1. Behavior Collection. To test the effectiveness of the IoT behavior capture solution, we successfully performed device simulation based on multiple firmware versions of NETGEAR WNAP320, WNDAP350, WNDAP360, and WNAP210 series devices. According to the activity of the device, we collect the system behavior generated by each IoT virtual device running normally for 10 days. Then, we conduct an IoT attack on each device and collect the abnormal system behavior generated by each IoT virtual device under different types of attacks. The system behavior of each IoT virtual device is recorded in the corresponding system behavior log file.

Table 6 shows the system behavior information of the IoT virtual device captured by the behavior capture scheme. We successfully captured the system behavior of the device in a normal scenario with a data size of 2.9 M and a number of 42,600. In addition, we successfully captured different abnormal system behavior in different attack scenarios. It is found that the information contained in the fine-grained system behavior logs is very rich. For example, in the Mirai attack scenario, the system behavior log details that BusyBox executes a file transfer instruction to download the malware (mirai.mips) to the IoT device, then set file permissions on the malware, and finally run the malware. After the malware is successfully executed on the IoT device, the mirai.mips process is generated, so the mirai.mips process and its corresponding process operations appear in the system behavior log. In the scenario of the BrickerBot attack, the system behavior logs recorded in detail the BusyBox process executing a series of commands to destroy the device.

It can be seen from this that our system behavior capture scheme successfully captures the system's fine-grained behavior, and these fine-grained system behaviors can reflect the state of the system. The system behavior collected by the system behavior capture scheme is used as the data set of the abnormal behavior detection scheme. The data set contains the normal system behavior of each device of about 40,000 and the abnormal system behavior of about 15,000. The training set of the model includes all normal system behavior (approximately 24,000), and the validation set and test set contain normal system behavior (approximately 8,000) and abnormal system behavior (approximately 75,00). In addition to some basic system behavior characteristics (e.g., the number of system behaviors, number of system calls, PID, and parameters), there are some hidden characteristics (e.g., the timing characteristics of system behavior) in the system behavior data set. To learn this rich feature information, we model the system behavior based on deep learning to perform abnormal detection.

4.3.2. Abnormal Behavior Detection. Abnormal behavior detection is a binary classification problem. We label the classification results as true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP is the abnormal system behavior accurately determined by the abnormal behavior detection model. TN is accurately determined normal system behavior. If the method determines a certain system behavior as abnormal, but the system behavior is normal, we mark the result as FP. If this method determines a certain system behavior as normal, but the system behavior is abnormal, we mark the result as FN.

In addition to the number of FP and FN, the ability of the classification method is usually evaluated by standard indicators such as precision, recall, and F1 score.

Precision represents the percentage of true abnormal behavior among all detected abnormal behaviors. The calculation method is as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{1}$$

Recall represents the percentage of abnormal behavior in the detected system behavior:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2}$$

The $F1$ score value is the harmonic average of precision and recall:

$$F1 = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \tag{3}$$

(1) Effect of the Value of h. We first test the window value h of different sizes, that is, the effect of system behavior sequence length on the classification results. We record the classification performance of IoT behavior anomaly detection classification models under different window sizes. The test results are shown in Figure 5(a). When the window size is 6, the anomaly detection classification model has a good classification effect compared with the models with other window sizes, with a high recall rate of 93.2%, a precision of 90.9%, and an $F1$ score of 92.03%. When the length of the system behavior sequence is too short, the sequence cannot cover all the characteristics of malicious behavior and normal behavior. When the system behavior sequence is very long, the deep learning network cannot store more information.

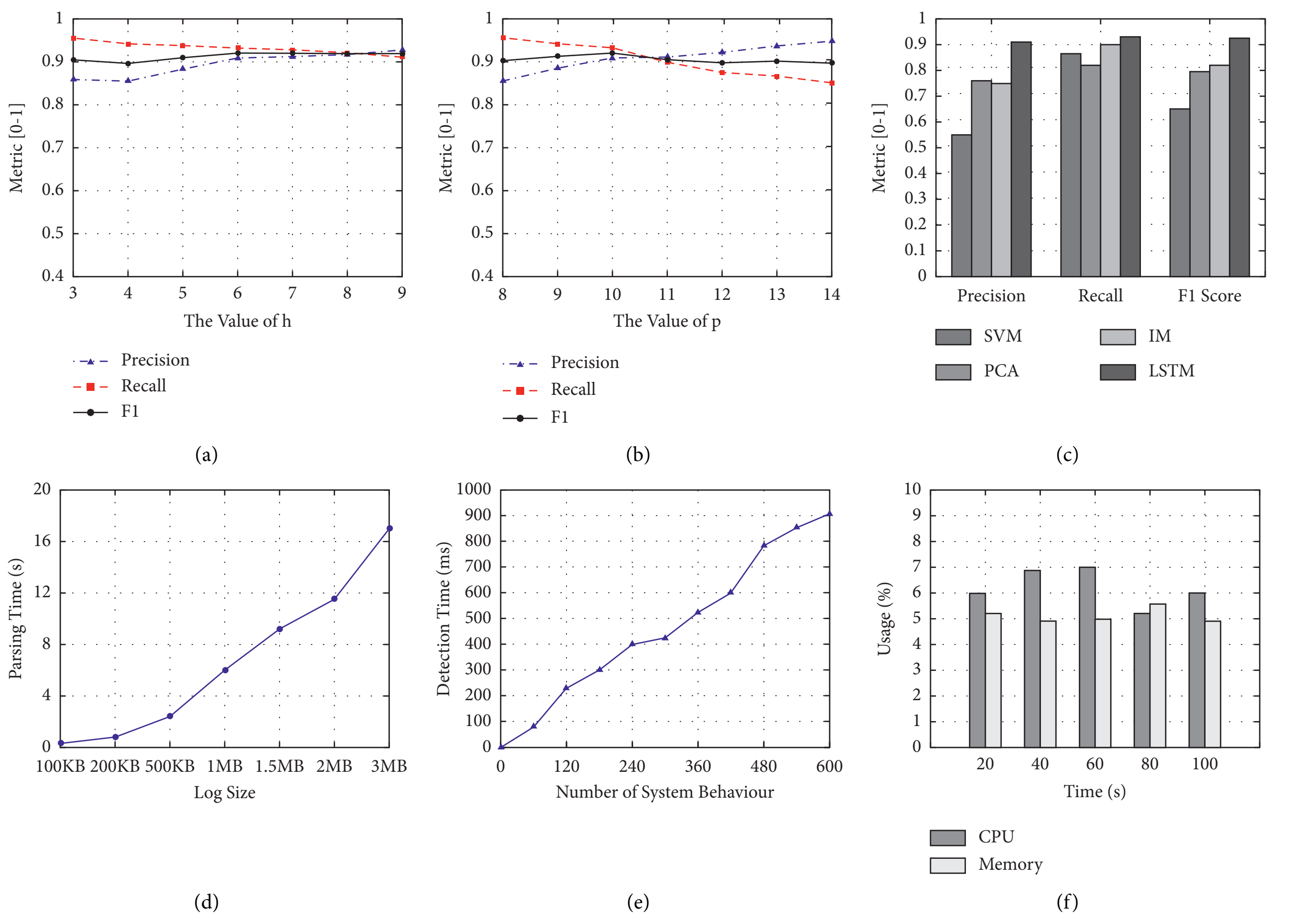


Figure 5: Performance test result. (a) Effect of the value of h. (b) Effect of the value of p. (c) Comparison with other algorithms. (d) Time cost of log parsing. (e) Time cost of behavior detection. (f) The consumption of CPU and memory.

(2) Effect of the Value of p. In the process of abnormal behavior detection, the next system behavior after the predicted window w is sorted according to the probability from large to small. If the system behavior is among the first p system behaviors, it is regarded as a normal value. Otherwise, the system behavior is regarded as abnormal behavior.

It is worth noting that many factors need to be considered when setting the candidate value p. If the candidate value p is set too large, some abnormal behavior information will be ignored, which may bring hidden dangers to the security of future IoT devices. If the candidate p is set too small, the normal system behavior may be judged as abnormal behavior information, which will interfere with abnormal detection. To test the influence of the candidate value p, we recorded the model anomaly detection results in the case of different candidate values p. The result is shown in Figure 5(b). When $p = 10$, the overall model has a great classification effect with a precision of 91.0%, F1 score of 90.5%, and recall rate of 90.1%, so we choose the value 10 as the size of the candidate value p.

(3) Comparison with Other Algorithms. Through the above analysis of the influence of the window size h and the influence of the candidate value p, we determined that we chose $h = 6$ and $p = 10$ as the parameters of our abnormal behavior detection model. In addition, we also determined that when the hidden layer is 3 layers, compared with other hidden layers, LSTM has the best performance. We compare LSTM with some representative log-based anomaly detection works, using support vector machine (SVM), principal component analysis (PCA), and invariant mining (IM). All these algorithms can be used to implement log-based anomaly detection and perform well. Figure 5(c) shows the comparison results of these four algorithms in terms of precision, recall, and $F1$ score. The results show that when the LSTM algorithm performs system behavior classification, the F1 value reaches 92%, which has a good classification effect.

This is because LSTM considers the sequence information of the system behavior sequence, and the other algorithms cannot capture this important information. The abnormal system behavior data set may contain a small amount of normal system behavior, which will affect the overall detection effect of the model, but in general, the abnormal behavior detection model can play a better behavior detection effect.

4.4. Performance

4.4.1. Behavior Log Parsing Performance. To measure the processing efficiency of behavior log parsing, we recorded the running time required to complete the entire parsing process. We parse system behavior log files of different sizes and record the parsing time. The result is shown in Figure 5(d), the system behavior resolution time increases as the size of the system behavior log increases. There are about 1,500 system behavior records in the average 100 kB system behavior log. The average 100 kB file, that is, 1,500 system behavior parsing, takes 0.5 seconds. It can be seen that the parsing time of the system behavior log is basically negligible.

In the training phase of the abnormal behavior detection model, a large amount of normal system behavior needs to be collected for training, so a large amount of normal system behavior needs to be analyzed. In our experiments, we collected system behavior logs generated by each device running normally for 1 hour for training. The system behavior log size is about 3 MB, the system behavior is about 40,000, and it takes about 17 s of parsing time. The time consumed by this process is completed before the model training. When the model training is successful, real-time detection of system behavior will not require such a large number of system behavior logs to be parsed. Therefore, the performance of parsing behavior logs is basically negligible.

4.4.2. Behavior Detection Performance. After the system behavior log is prepared and parsed, we perform model training on the system behavior event sequence, the window size is 6, the predetermined value p is 10, and the number of hidden layers is 3 to train the model. The time consumption of model training is about 3,101 s. After the deep learning-based abnormal behavior detection model is trained, the system performs real-time detection on the IoT system behavior based on the model. The efficiency of detection is an important indicator of the detection effect, which reflects the time spent in detecting abnormal system behavior. Since the Internet of things is a system that requires timeliness, we do not want the abnormal behavior detection system to cause excessive time consumption in the detection process, thereby affecting the experience of using IoT devices.

To test the time consumption of the system in the process of detecting abnormal behavior, we recorded the time it took to detect different amounts of system behavior. Figure 5(e) shows the time used for abnormal behavior detection under different system behaviors. The results show that the time of abnormal behavior detection increases with the increase in the number of system behaviors, but the average detection time per 100 system behaviors is about 150 ms, which is within the acceptable range.

4.4.3. CPU and Memory Resource Consumption. We build virtual security functions for each device on the IoT behavior detection server. This virtual security function needs to perform behavior capture, anomaly detection, risk mitigation, and other functions during operation, so it needs to occupy a certain amount of CPU and memory resources for calculation. Figure 5(f) shows the amount of CPU and memory resources used during the abnormal behavior detection process of the virtual security function running stably for 100 seconds. The results in the figure show that the virtual security function takes up an average of 6.2% of CPU resources and an average of 5.1% of memory resources, which are within the acceptable range.

5. Related Work

5.1. IoT Behavior Security. Although the Internet of things has great potential, it also faces many security challenges [24]. Unfortunately, the security and privacy risks of IoT devices have not received enough attention. Due to a large number of IoT devices and a lack of defense capabilities, they are very attractive targets [25]. For example, IoT devices are used as robots to launch DDoS or spam, smart meters are attacked to reduce utility bills, and handheld scanners are hacked to enter logistic companies.

Most previous research on IoT security and privacy has focused on the use of firewalls [26], intrusion detection [27], access control strategies [28–30], and software patches [31] to protect the IoT infrastructure from attacks. These measures cannot guarantee the behavior security of IoT devices. For example, firewall rules can hardly guarantee that the door is locked when the user is not at home. In addition, the analysis of IoT devices and environments focuses on ensuring the security of IoT applications by analyzing source code. For example, some systems [32] infer the context of the application through run-time prompts to enforce permissions based on that context or require the user to authorize through the interface [33], while other systems apply static models to check for violations. Unfortunately, the current dynamic method is not enough to identify violations, while the static method [34] lacks accuracy and only implements limited strategies.

At present, there has been a lot of research on the behavior security of the Internet of things. HoMonit [35] is used for detecting abnormal behavior on the smart home platform, using side-channel information leakage in the wireless communication channel (packet size and packet interval) to infer the type of communication events between the smart device and the hub and then compare the inferred sequence of events with the expected program logic to identify inappropriate behavior without any modification to the existing infrastructure. IoTMon [36] is an IoT device physical interaction control system that can discover any possible physical interaction and generate all potential interaction chains across applications in the IoT environment. IoTMon also evaluates and mitigates the security risks of each discovered interaction chain between applications based on its physical influence. FlowFence [37] uses the taint tracking technology to track the information flow of sensitive data in the Internet of things applications, to ensure that the authorized users can use the data safely and legally to prevent the leakage of sensitive information. MCshield [38] is a DDoS defense framework. This framework deploys multiple smart filters at the edge of the attack source/target network to filter malicious traffic by learning DDoS behavior.

5.2. Firmware Virtualization. Avatar [39] is a framework that supports dynamic security analysis of embedded system firmware. The framework performs dynamic analysis by running firmware on actual hardware. Although this method is very accurate, it has significant problems. Firstly, Avatar must obtain the physical hardware of the device under test, which puts a huge financial burden on developers. Secondly, Avatar needs to manually identify and interact with the debug port on the device interface, which reduces the scalability of this technology, especially for consumer devices that may not support hardware debugging. How to run the firmware in a virtualized environment and analyze the security of IoT devices is an important task. To solve this problem, FIRMADYNE [40] proposed a complete system simulation that relies on software, so as to realize the large-scale and automated dynamic analysis of embedded firmware binaries. FIRMADYNE solves the inherent challenges of dynamic analysis of embedded systems, such as the presence of hardware-specific peripherals, the use of non-volatile memory, and the creation of dynamically generated files. Costin [41] provides an extensible automated firmware dynamic analysis framework, using pure software simulation to find vulnerabilities in embedded devices. The framework can test the firmware Web application security issues through simulation. FIRM-AFL [42] is the first high-throughput grey box fuzzer for IoT firmware. It proposes a new technology for enhanced process simulation to solve the performance bottleneck caused by QEMU system mode simulation. P2IM [43] presents an abstract model for the I/O behaviors of the processor-peripheral interfaces to enable peripheral-oblivious emulation of MCU devices. uEMU [44] builds a general model for each peripheral to learn how to correctly emulate firmware execution at individual peripheral access points. It takes the image as input and symbolically executes it by representing unknown peripheral registers as symbols. During symbolic execution, it infers the rules to respond to unknown peripheral accesses.

However, the existing research work on the behavior of IoT devices focuses on the security detection of the network behavior and application behavior of IoT devices, and there is no comprehensive consideration of the behavior of IoT devices at the system level. Due to limited resources, it is difficult for IoT devices to obtain monitoring system-level behavior by installing monitoring software. Firmware virtualization technology can solve this problem. By simulating the real operating environment of the firmware, the operating data and security status of the system layer inside the device can be obtained, to perform system-wide security status detection and defense and provide a new idea for solving the problem of behavior security of the Internet of things devices. However, the current work in this area is mainly focused on the dynamic analysis and vulnerability mining of IoT devices, failing to consider the behavioral security detection of IoT devices. Therefore, we propose a behavior security detection system for IoT devices based on firmware virtualization and deep learning in this study.

5.3. Log-Based Anomaly Detection. Anomaly detection plays an important role in the management of modern large distributed systems. Logs that record system run-time information are widely used for anomaly detection. Lang et al. [45] used the SVM algorithm, a commonly used supervised classification method, to build a failure prediction model based on event logs. However, supervised methods require lots of manual efforts to construct data and labels, so unsupervised methods are more practical. Xu et al. [46] proposed an unsupervised method for detecting anomalous sequences of events using PCA, but PCA is data-sensitive so the detection accuracy of PCA will vary on different data sets. Lou et al. [47] converted log sequences into event count vectors and then used IM to mine invariants within vectors. The mined invariants would reflect the normal workflow of the detected system. Then, they used these invariants to detect anomalies in system logs. If the invariant relationship of the log session did not hold, it would be judged as an abnormal session.

6. Discussion and Future Work

This study designs and implements an IoT device behavior safety detection system based on firmware virtualization and deep learning. It implements IoT device behavior security detection based on fine-grained real-time dynamic system behavior, analyzes abnormal behavior, and proposes corresponding risk mitigation strategies. However, there are limitations in the design of this article, and further research and improvement are needed in future work.

(1) Due to the wide variety of IoT devices, firmware virtualization technology only supports firmware images that emulate fixed architectures (ARM and MIPS) and fixed systems (Linux). Even if the firmware image meets the appealing architecture and system conditions, there may be problems with incomplete image files and image encryption, resulting in a considerable number of IoT devices that cannot successfully simulate virtual IoT devices based on the device firmware and thus cannot obtain the IoT device's fine-grained dynamic behavior. Hence, firmware virtualization in a more in-depth way should be studied to realize a more general simulation method of the real operating environment of IoT devices, to better simulate IoT virtual devices.

(2) There are many types of IoT attack methods, and some attack types can achieve security detection and defense through security tools such as intrusion detection systems (e.g., Snort) and firewalls. System behavior can be used as a supplement to network behavior, thereby improving the framework for the secure detection of IoT device behavior. We expect to be combined with software-defined security in the future to generate multiple virtual security functions for each type of IoT device or each IoT device. The virtual security function can be used to detect and defend against a variety of IoT attacks against the network layer, or it can be based on firmware virtualization to simulate a real system environment to detect and defend against IoT system-level attacks.

(3) For the training process of abnormal behavior model based on deep learning, the training data set comes from the system behavior collected when the IoT device is initially connected to the network to maintain stable operation. However, in the actual environment, the system behavior of IoT devices may change due to reasons such as firmware updates. This change leads to the incompleteness of the previously trained model. In future work, we will further improve model accuracy and system performance.

7. Conclusion

In this study, we design and implement IoT-DeepSense, an IoT device behavior security detection system based on firmware virtualization and deep learning. Based on firmware virtualization technology, we build the real operating environment of the IoT system to capture the fine-grained system behaviors, then conduct security detection of the IoT device behaviors based on deep learning, analyze the abnormal behavior, and mitigate the risk according to the detection results. The implementation of the entire system is carried out on a separate IoT behavior detection server, which does not require modification of IoT devices with limited resources and is highly scalable. The evaluation results show that our abnormal behavior detection model has a good effect with an F1 score of 92%.

Data Availability

To test the effectiveness of the IoT behavior capture solution, we successfully performed device simulation based on multiple firmware versions of NETGEAR WNAP320, WNDAP350, WNDAP360, and WNAP210 series devices. According to the activity of the device, we collect the system behavior generated by each IoT virtual device running normally for 10 days. Then, we conduct an IoT attack on each device and collect the abnormal system behavior generated by each IoT virtual device under different types of attacks. The system behavior of each IoT virtual device is recorded in the corresponding system behavior log file.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61872430 and in part by the Hubei Key Research and Development Program under Grant 2020BAA003 and the National Basic Research Program of China (973 Program) under Grant 2014CB340600.

References

[1] I. Lee and K. Lee, "The internet of things (iot): applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[2] G. Davis, "2020:Life with 50 billion connected devices," in *Proceedings of the 2018 IEEE International Conference on Consumer Electronics (ICCE)*, p. 1, Las Vegas, NV, USA, 2018.

[3] M. M. Hossain, M. Fotouhi, and R. Hasan, "Towards an analysis of security issues, challenges, and open problems in the internet of things," in *Proceedings of the 2015 IEEE World Congress on Services*, pp. 21–28, IEEE, New York, NY, USA, July 2015.

[4] B. Edge, "Hacking the human heart," 2016, http://bigthink.com/future-crimes/hacking-the-human-heart.

[5] Wikipedia, "Dyn cyberattack," 2016, https://securityintelligence.com/posts/internetof-threats-iot-botnets-network-attacks/.

[6] D. McMillen, "Internet of threats: iot botnets drive surge in network attacks," 2021, https://www.euronews.com/2016/10/22/what-we-know-about-the-dyn-cyber-attack.

[7] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of iot new features on security and privacy: new threats, existing solutions, and challenges yet to be solved," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1606–1616, 2018.

[8] F. Bellard, "Qemu, a fast and portable dynamic translator," *USENIX Annual Technical Conference, FREENIX Track*, vol. 41, p. 46, 2005.

[9] M. van Gerven and S. Bohte, "Editorial: artificial neural networks as models of neural information processing," *Frontiers in Computational Neuroscience*, vol. 11, p. 114, 2017.

[10] R. C. O'Reilly and M. J. Frank, "Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia," *Neural Computation*, vol. 18, no. 2, pp. 283–328, 2006.

[11] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of the Eleventh Annual Conference of the International Speech Communication Association*, Chiba, Japan, September 2010.

[12] J. A. Pérez-Ortiz, F. A. Gers, D. Eck, and J. Schmidhuber, "Kalman filters improve lstm network performance in problems unsolvable by traditional recurrent nets," *Neural Networks*, vol. 16, no. 2, pp. 241–250, 2003.

[13] A. Graves, A. Mohamed, and G. 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, IEEE, USA, May 2013.

[14] K. Greff, R. k. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: a search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[15] Motion, "Motion," 2019, https://motion-project.github.io/.

[16] M. Yoon, S. Mohan, J. Choi, M. Christodorescu, and L. Sha, "Learning execution contexts from system call distribution for anomaly detection in smart embedded system," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pp. 191–196, Pittsburgh, USA, April 2017.

[17] R. Vaarandi and M. Pihelgas, "Logcluster-a data clustering and pattern mining algorithm for event logs," in *Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM)*, pp. 1–7, IEEE, Barcelona, Spain, November 2015.

[18] A. A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "Clustering event logs using iterative partitioning," in *Proceedings of the 15th ACM SIGKDD International Conference*

on Knowledge Discovery and Data Mining, pp. 1255–1264, Paris, France, June 2009.

[19] Q. Fu, J. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, pp. 149–158, IEEE, Washington, DC, USA, 2009.

[20] M. Du and F. Li, "Spell: streaming parsing of system event logs," in *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 859–864, IEEE, Barcelona, Spain, December 2016.

[21] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: an online log parsing approach with fixed depth tree," in *Proceedings of the 2017 IEEE International Conference on Web Services (ICWS)*, pp. 33–40, IEEE, Honolulu, HI, USA, June 2017.

[22] J. Zhu, S. He, J. Liu et al., "Tools and benchmarks for automated log parsing," in *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 121–130, IEEE, Montreal, QC, Canada, May 2019.

[23] V. Costan and S. Devadas, "Intel sgx explained," *IACR Cryptology ePrint Archive*, vol. 2016, no. 86, 118 pages, 2016.

[24] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: rethinking network security for the internet-of-things," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, pp. 1–7, Philadelphia, PA, USA, November 2015.

[25] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[26] S. Kubler, K. Främling, and A. Buda, "A standardized approach to deal with firewall and mobility policies in the iot," *Pervasive and Mobile Computing*, vol. 20, pp. 100–114, 2015.

[27] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[28] W. He, M. Golla, R. Padhi et al., "Rethinking access control and authentication for the home internet of things (iot)," in *Proceedings of the 27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 255–272, Baltimore, MD, USA, August 2018.

[29] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 2020, Article ID 3022797, 1 page, 2020.

[30] C. Wang, D. Wang, G. Xu, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy for industry 4.0," *Science China Information Sciences*, vol. 65, no. 1, pp. 1–15, 2022.

[31] O. Leiba, Y. Yitzchak, R. Bitton, A. Nadler, and A. Shabtai, "Incentivized delivery network of iot software updates based on trustless proof-of-distribution," in *Proceedings of the 2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 29–39, IEEE, London, UK, April 2018.

[32] Y. J. Jia, Q. A. Chen, S. Wang et al., "Contexlot: towards providing contextual integrity to appified iot platforms," *NDSS*, vol. 2, no. 2, p. 2, 2017.

[33] Y. Tian, N. Zhang, Y.-H. Lin et al., "Smartauth: user-centered authorization for the internet of things," in *Proceedings of the 26th {USENIX} Security Symposium ({USENIX} Security 17)*, pp. 361–378, Baltimore, MD, USA, 2017.

[34] Z. B. Celik, P. McDaniel, and G. Tan, "Soteria: automated iot safety and security analysis," in *Proceedings of the 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, pp. 147–158, 2018.

[35] 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, ON, Canada, October 2018.

[36] W. Ding and H. Hu, "On the safety of iot device physical interaction control," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 832–846, Toronto, ON, Canada, October 2018.

[37] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: practical data protection for emerging iot application frameworks," in *Proceedings of the 25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 531–548, Austin, TX, USA, August 2016.

[38] N. Dao, T. V. Phan, J. Kim, T. Bauschert, S. Cho, and D. Do, "Securing heterogeneous iot with intelligent ddos attack behavior learning," *IEEE Systems Journal*, pp. 1–10, 2017.

[39] J. Zaddach, L. Bruno, A. Francillon, and D. Balzarotti, "Avatar: a framework to support dynamic security analysis of embedded systems' firmwares," *NDSS*, vol. 14, pp. 1–16, 2014.

[40] D. D. Chen, M. Woo, D. Brumley, and M. Egele, "Towards automated dynamic analysis for linux-based embedded firmware," *NDSS*, vol. 16, pp. 1–16, 2016.

[41] A. Costin, A. Zarras, and A. Francillon, "Automated dynamic firmware analysis at scale: a case study on embedded web interfaces," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 437–448, Xi'an, China, June 2016.

[42] Y. Zheng, A. Davanian, H. Yin, C. Song, H. Zhu, and L. Sun, "Firm-afl: high-throughput greybox fuzzing of iot firmware via augmented process emulation," in *Proceedings of the 28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 1099–1114, Santa Clara, CA, USA, August 2019.

[43] B. Feng, A. Mera, and L. Lu, "P2im: scalable and hardware-independent firmware testing via automatic peripheral interface modeling," in *Proceedings of the 29th USENIX Security Symposium (USENIX Security 20)*, pp. 1237–1254, USENIX Association, Santa Clara, CA, USA, September 2020.

[44] W. Zhou, L. Guan, P. Liu, and Y. Zhang, "Automatic firmware emulation through invalidity-guided knowledge inference," in *Proceedings of the 30th USENIX Security Symposium (USENIX Security 21)*, USENIX Association, Santa Clara, CA, USA, November 2021.

[45] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in ibm bluegene/l event logs," in *Proceedings of the Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pp. 583–588, IEEE, Omaha, NE, USA, 2007.

[46] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, pp. 117–132, Big Sky, MT, USA, October 2009.

[47] J. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Ming invariants from console logs for system problem detection," in *Proceedings of the USENIX Annual Technical Conference*, pp. 1–14, Boston, MA, USA, June 2010.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 1862888, 19 pages
https://doi.org/10.1155/2022/1862888

Review Article

Machine Learning Techniques for Spam Detection in Email and IoT Platforms: Analysis and Research Challenges

Naeem Ahmed[1], **Rashid Amin**[1], **Hamza Aldabbas**[2], **Deepika Koundal**[3], **Bader Alouffi**[4] **and Tariq Shah**[1]

[1]*Department of Computer Science, University of Engineering and Technology, Taxila, Pakistan*
[2]*Prince Abdullah Bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University, Al-Salt, Jordan*
[3]*Department of Systemics, School of Computer Science, University of Petroleum & Energy Studies, Dehradun, India*
[4]*Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia*

Correspondence should be addressed to Rashid Amin; rashid4nw@gmail.com

Received 21 September 2021; Revised 1 November 2021; Accepted 30 November 2021; Published 3 February 2022

Academic Editor: Wenjia Li

Nowaday, emails are used in almost every field, from business to education. Emails have two subcategories, i.e., ham and spam. Email spam, also called junk emails or unwanted emails, is a type of email that can be used to harm any user by wasting his/her time, computing resources, and stealing valuable information. The ratio of spam emails is increasing rapidly day by day. Spam detection and filtration are significant and enormous problems for email and IoT service providers nowadays. Among all the techniques developed for detecting and preventing spam, filtering email is one of the most essential and prominent approaches. Several machine learning and deep learning techniques have been used for this purpose, i.e., Naïve Bayes, decision trees, neural networks, and random forest. This paper surveys the machine learning techniques used for spam filtering techniques used in email and IoT platforms by classifying them into suitable categories. A comprehensive comparison of these techniques is also made based on accuracy, precision, recall, etc. In the end, comprehensive insights and future research directions are also discussed.

1. Introduction

In the era of information technology, information sharing has become very easy and fast. Many platforms are available for users to share information anywhere across the world. Among all information sharing mediums, email is the simplest, cheapest, and the most rapid method of information sharing worldwide. But, due to their simplicity, emails are vulnerable to different kinds of attacks, and the most common and dangerous one is spam [1]. No one wants to receive emails not related to their interest because they waste receivers' time and resources. Besides, these emails can have malicious content hidden in the form of attachments or URLs that may lead to the host system's security breaches [2]. Spam is any irrelevant and unwanted message or email sent by the attacker to a significant number of recipients by using emails or any other medium of information sharing [3]. So, it requires an immense demand for the security of the email system. Spam emails may carry viruses, rats, and Trojans. Attackers mostly use this technique for luring users towards online services. They may send spam emails that contain attachments with the multiple-file extension, packed URLs that lead the user to malicious and spamming websites and end up with some sort of data or financial fraud and identify theft [4, 5]. Many email providers allow their users to make keywords base rules that automatically filter emails. Still, this approach is not very useful because it is difficult, and users do not want to customize their emails, due to which spammers attack their email accounts.

In the last few decades, Internet of things (IoT) has become a part of modern life and is growing rapidly. IoT has become an essential component of smart cities. There are a lot of IoT-based social media platforms and applications. Due to the emergence of IoT, spamming problems are increasing at a high rate. The researchers proposed various spam detection methods to detect and filter spam and spammers. Mainly, the existing spam detection methods are divided into two types: behaviour pattern-based approaches and semantic pattern-based approaches. These approaches have their limitations and drawbacks. There has been significant growth in spam emails, along with the rise of the Internet and communication around the globe [6]. Spams are generated from any location of the world with the Internet's help by hiding the attacker's identity. There are a plenty of antispam tools and techniques, but the spam rate is still very high. The most dangerous spams are malicious emails containing links to malicious websites that can harm the victim's data. Spam emails can also slow down the server response by filling up the memory or capacity of servers. To accurately detect spam emails and avoid the rising email spam issues, every organization carefully evaluates the available tools to tackle spam in their environment. Some famous mechanisms to identify and analyze the incoming emails for spam detection are Whitelist/Blacklist [7], mail header analysis, keyword checking, etc.

Social networking experts estimate that 40% of social network accounts are used for spam [8]. The spammers use popular social networking tools to target specific segments, review pages, or fan pages to send hidden links in the text to pornographic or other product sites designed to sell something from fraudulent accounts. The noxious emails that are sent to the same kind of individuals or associations share regular highlights. By investigating these highlights, one can improve the detection of these types of emails. By utilizing artificial ntelligence (AI) [9], we can classify emails into spam and nonspam emails. This solution is possible by using feature extraction from the messages' headers, subject, and body. After extracting this data based on their nature, we can group them into spam or ham. Today, learning-based classifiers [10] are commonly used for spam detection. In learning-based classification, the detection process assumes that spam emails have a specific set of features that differentiate them from legitimate emails [11]. Many factors increase the complexity of the identification process of spam in learning-based models. These factors include spam subjectivity, idea drift, language problems, overhead processing, and text latency.

One example of learning-based models is extreme learning machine (ELM). This is a modern machine learning model for the feedforward neural networks containing only one hidden layer [12]. It eliminates slow training speed and overfitting problems when compared with traditional neural networks. In ELM, it requires only one cycle of iteration. Because of better generalization potential, robustness, and controllability, this algorithm specifically is now used in many fields. In this paper, we consider different machine learning algorithms for spam detection. Our contributions are delineated as follows:

(i) The study discusses various machine learning-based spam filters, their architecture, along with their pros and cons. We also discussed the basic features of spam email.

(ii) Some exciting research gaps were found in the spam detection and filtering domain by conducting a comprehensive survey of the proposed techniques and spam's nature.

(iii) Open research problems and future research directions are discussed to enhance email security and filtration of spam emails by using machine learning methods.

(iv) Several challenges currently faced by spam filtering models and the effects of those challenges on the models' efficiency are discussed in this study.

(v) A comprehensive comparison of machine learning techniques and concepts that help understand machine learning's role in spam detection is provided.

(vi) The study categorizes different spam detection methods according to machine learning techniques to better understand concepts jointly.

(vii) Various future spam detection and filtration directions are discussed that could be explored to detect spam better and add more security to email platforms.

The rest of the paper is organized into nine sections. Section 2 discusses the comparison of previous surveys that were done on email spam detection. Section 3 discusses the basics of email spam and its effects on the community. Section 4 focuses on basic methods used for spam filtration. Section 5 elaborates on the machine learning background, while Section 6 provides an overview of machine learning algorithms used for spam filtration. This section also reviews various papers and proposed machine learning techniques for spam filtration and detection. Section 7 presents the open issues and research gaps, while Section 8 discusses challenges of spam detection systems. At the end, Section 9 concludes and presents the future directions of email spam detection and filtration. Table 1 presents the list of acronyms used in this article with corresponding definitions.

2. Comparison with Previous Surveys

Email spam is nothing more than fake or unwanted bulk mails sent via any account or an automated system. Spam emails are increasing day by day, and it has become a common problem over the last decade. Email IDs receiving spam emails are typically collected through spambots (a computerized application that crawls email addresses across the Internet). The applications of machine learning have been playing a vital role in the detection of spam emails. It has various models and techniques that researchers are using to develop novel spam detection and filtering models [13]. Kaur and Verma [14] present a survey on email spam detection using a supervised approach with feature selection. They discuss the knowledge discovery process for spam detection systems. They also elaborate various techniques and tools proposed for spam detection. The choice of features based on N-Gram is also addressed in this survey. N-Gram [15, 16] is a predictive-based algorithm used to

Table 1: A-list of acronyms used in this article with corresponding definitions.

Acronym	Description
KNN	K-nearest neighbors
NN	Neural networks
SVM	Support vector machine
MLP	Multilayer perceptron neural network
ECML	European conference of machine learning
AI	Artificial intelligence
CART	Classification and regression tree
TF/IDF	Term frequency/inverse document frequency
PSO	Particle swarm optimization
DTM	Document term frequency
BOG	Bag of words
ML	Machine learning
NB	Naïve Bayes
NB tree	Naïve Bayes tree
LAD tree	Logistic analysis of data tree
REP tree	Reduced error pruning tree
UCI	University of California Irvine repository of machine learning databases
XML	Extensible markable language
ID3	Iterative dichotomizer 3
SOM	Self-organizing maps
DBSCAN	Density-based spatial clustering of applications with noise
ELM	Extreme learning machines
AD tree	Alternating decision tree

predict the probability of the next word occurrence after finding $N-1$ terms in a sentence or text corpus. N-Gram uses probability-based techniques for the next word prediction. They compare various machine learning (multilayer perceptron neural network support vector machine, Naïve Bayes) and nonmachine learning (Signatures, Blacklist and Whitelist, and mail header checking) approaches for email spam detection.

Saleh et al. [17] present a survey on intelligent spam email detection. They discuss various security risks of emails, especially spam emails, the scope of spam analysis, and different machine learning and nonmachine learning techniques for spam detection and filtering. They conclude that there is high adoption of supervised learning [18] algorithms for email spam detection. They state that the high usage of supervised learning is the accuracy and consistency of supervised techniques. They also discussed multialgorithm frameworks and found that multialgorithm frameworks are more efficient than a single algorithm. They found that nearly all research work that uses the content of emails for the identification spam, particularly phishing emails, depends on word-based classification or clustering systems.

Blanzieri and Bryl [2, 19] describe a list of learning-based email spam filtering approaches. In this paper, they addressed the spam problems and provided a review of learning-based spam filtering. They explain various features of spam emails. In this study, effects of spam emails on different domains were discussed. Various economic and ethical issues of spam are also discussed in this study. The antispam approach that is common and learning-based filtering is well developed. The commonly used filters are based on different classification techniques applied to various components of email messages. This study suggests that the Naïve Bayes classifier holds a particular position amongst multiple learning algorithms used for spam filtering. With splendid pace and simplicity, it gives high precision results.

Bhuiyan et al. [20] present a review of current email spam filtering approaches. They summarize multiple spam filtering approaches and sum up the accuracy on various parameters of different proposed systems by analyzing numerous processes. They discuss that all the existing methods are efficient for filtering spam emails. Some have successful results, and others are attempting to incorporate other ways to boost their accuracy performance. Although they are all successful, they still have some issues in spam filtering methods, which is the primary concern for researchers. They are trying to create a next-generation spam filtering mechanism to understand large numbers of multimedia data and filter spam emails. They conclude that most email spam filtering is done by utilizing Naïve Bayes and the SVM algorithm. To test the spam filtration models, these models can be trained on different datasets, such as "ECML" and UCI dataset [21].

Ferrag et al. [13] presented a review of deep learning algorithms of intrusion detection systems and spam detection datasets. They discussed various detection systems based on deep learning models and evaluated the effectiveness of those models. They examined 35 well-known cyber dataset by dividing them into seven categories. These categories include Internet traffic-based, network traffic-based, Interanet traffic-based, electrical network-based, virtual private network-based, andriod apps-based, IoT traffic-based, and Internet connected device-based datasets. They conclude that deep learning models can perform better than traditional machine learning and lexicon models for intrusion and spam detection.

Vyas et al. [22] present a review on supervised machine learning strategies for filtering spam emails. They concluded that the Naïve Bayes method provides faster results and decent precision over all other methods (except SVM and ID3) from all the techniques discussed. SVM and ID3 offer greater precision than Naïve Bayes but take much longer time to construct a system. There is a trade-off between timing and precision. They conclude that selecting the learning algorithm heavily depends on the situation and the required accuracy and time. They state that all parts of the email should be considered in the future to create a more robust spam filtering framework.

This survey paper discusses three main types of machine learning that can be used for spam filtering. We review various papers, the proposed techniques, and discuss challenges to spam detection and filtration systems. This article also focuses on the advantages and disadvantages of the proposed techniques for spam detection and filtration that is never reviewed in the past.

3. Spam Messages

The email spam definition is ambiguous since everybody has their views on it. At present, email spam is getting the attention of everyone. Email spam ordinarily includes particular spontaneous messages sent in mass by individuals you do not know. The term spam is obtained from the Monty Python sketch [23], in which the Hormel canned meat item has numerous tedious emphases. While the term spam was purportedly first utilized in 1978 to allude to unwanted email, it increased rapidly in the mid-1990s, as we get to turn out to be progressively typical outside scholastic and research circles [24]. A notable model is the development expense trick in which a client receives an email with an offer that should bring about a prize. In the era of technology, the dodger/spammer shows a story where the unfortunate casualty needs forthright financial help so that the fraudster can gain a lot bigger total of cash, which they would then share. The fraudster will either earn a profit or avoid communication when the unfortunate victim completes the installment.

3.1. Spam Filtering Methods in Email and IoT Platforms. The number of spam emails is rapidly increasing in marketing, chain communications, stock market tips, politics, and education [24]. Currently, various companies develop different techniques and algorithms for efficient spam detection and filtering. We address some filtering strategies in this section to understand the filtering process.

3.1.1. The Standard Spam Filtering Method. Standard spam filtering is a filtering system that implements a set of rules and works with that set of protocols as a classifier. Figure 1 illustrates a standard method for filtering spam. In the first

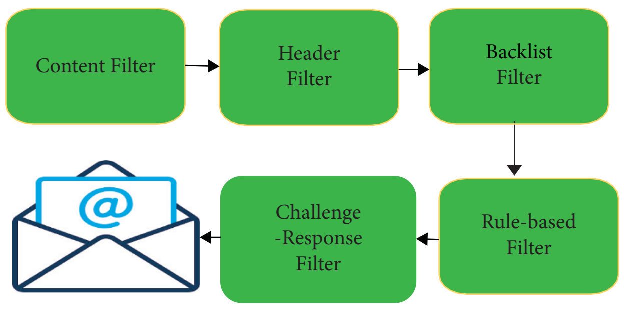


FIGURE 1: Standard spam filtering.

step, content filters are implemented and use artificial intelligence techniques to figure out the spam [25]. The email header filter, which extracts the header information from the email, is implemented in the second step. After that, backlist filters are applied to the emails to clinch the emails coming from the backlist file to avoid spam emails. After this stage, rule-based filters are implemented, recognizing the sender using the subject line and user-defined parameters. Eventually, allowance and task filters are used by implementing a method that allows the account holder to send the mail [26].

3.1.2. The Client Side Spam Filtering. A client is a person who can use the Internet or email network to send or receive an email [27]. Spam detection at the client point offers different rules and mechanisms to ensure secure communications transmission between people and organizations. For transmission of data, a client should deploy multiple existing frameworks on his/her system. Such systems connect with client mail agents and filter the client's mailbox by compositing, accepting, and managing the incoming emails [28, 29].

3.1.3. Enterprise Level Spam Filtering. Email spam detection at the enterprise level is a technique in which various filtering frameworks are installed on the server, dealing with the mail transfer agent and classifying the collected emails into one spam or ham [30]. This system client uses the system consistently and effectively on a network with an enterprise filtering technique to filter the emails. Existing methods of spam detection use the rule of ranking the email. A ranking function is specified in this principle, and a score is generated against every post. The junk mail or ham message is given specific scores or ranks [31]. Since spammers use different approaches, all tasks are regularly modified by implementing a list-based technique to block the messages automatically. Figure 2 is reproduced from Bhuiyan et al. [20]. Figure 2 shows the architecture of the client and enterprise level spam filtering process.

3.1.4. Case-Based Spam Filtering. One of the well-known and conventional machine learning methods for spam detection is the case-based or sample-based spam filtering system [32]. A typical case base filtering structure is illustrated in Figure 3. There are many phases to this type of

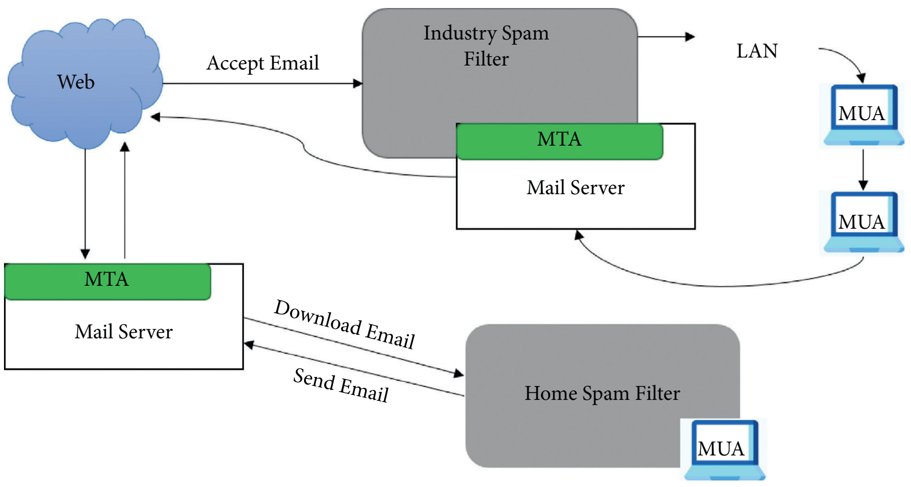


Figure 2: Client based and enterprise level spam filtering [20].

filtering with the aid of the collection method; it collects data (mails) during the first step. After that, the major transition continues with the preprocessing steps through the client graphical user interface, outlining abstraction, and choice of email data classification, testing the entire process using vector expression and classifying the data into two classes: spam and legitimate email.

Finally, the machine learning technique is extended to training sets and test sets to determine whether this is an email. The final decision is made through two steps: self-observation and classifier's result, deciding whether the email is spam or legitimate [32, 33].

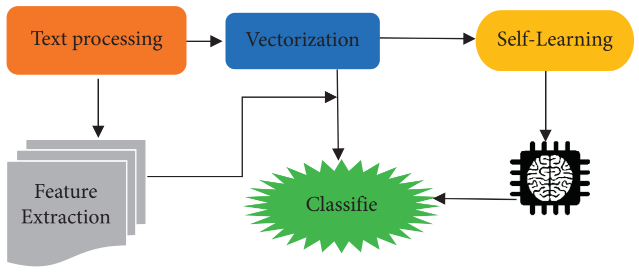


Figure 3: Case-based spam filtering.

4. Internet of Things and Its Attacks (IoT)

The Internet of things (IoT) means a system of interrelated, Internet-connected objects that collect and transfer data over a wireless network without the intervention of humans. IoT enables the integration and implementation of real-world objects regardless of location. In such a scenario, privacy and security techniques are highly critical and challenging in network management and monitoring performance. To solve security problems, such as intrusions, phishing attacks, DoS attacks, spamming, and malware in IoT applications must protect privacy. Ios systems, including objects and networks, are vulnerable to network and physical attacks and privacy failures. The main types of IoT attacks are illustrated in Figure 4.

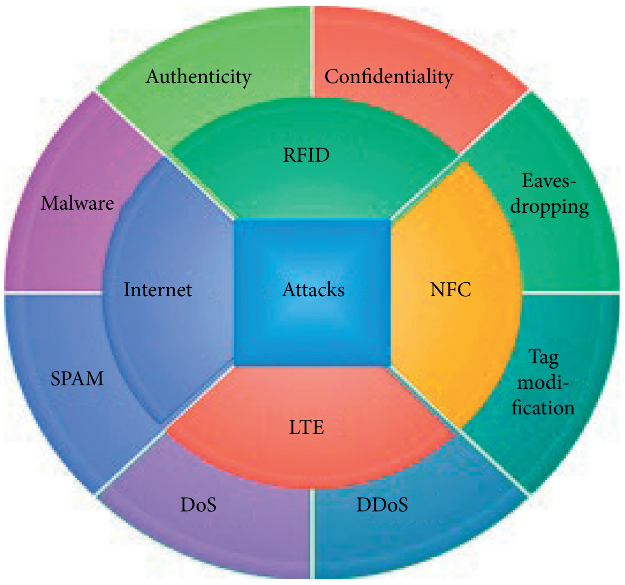


Figure 4: IoT attacks.

The various attacks of IoT systems are listed as follows.

(a) *Self-Promotion Attack*. In this type of attack, the compromised node tries to get importance over the other nodes of the IoT environment for the particular recommendation.

(b) *Bad Mouthing Attack*. In this attack, the compromised node forgave a wrong recommendation; it may execute the trust of the trusted node. It decreased the services of the trusted node.

(c) *Ballot Stuffing Attack*. In this challenge of the IoT environment, the compromised node enhances the other compromised nodes. It is a chance for the compromised node to provide the services. It is also known as the collision recommendation attack.

(d) *Opportunistic Service Attack*. In this type of attack, the compromised node collaborates with the other malicious nodes to build the bad mouthing and ballot stuffing attack.

(e) *On-Off Attack*. In this type of attack, the compromised node provides inadequate services, which means that the compromised node randomly performs a bad service.
(f) *Node Tempering*. The attacker changes the malicious node and gets specific information such as a security key.
(g) *Malicious Node Attack*. The attacker physically adds the malicious node among nodes.
(h) *Man in the Middle Attack*. The attacker secretly intercepts the communication between two nodes over the Internet in this type of attack. The attacker gets the main information by eavesdropping.
(i) *Sybil Attack*. The compromised node steals the recognition of good nodes and acts as a suitable node.

According to a study from Nozomi Networks, in the first half of 2020, there were increasing attacks and threats on Operational Technology (OT) and the IoT networks. Figure 5 shows the number of attacks in IoT devices in respective years.

Machine learning techniques can be used for the prevention and detection of these attacks with high performance. Various research studies have been carried out to detect and prevent the above issues discussed in Section 5.

5. Machine Learning

Machine learning [34] is one of the most important and valuable applications of artificial intelligence (AI), which gives computer systems the ability of automatically learning and enhancing their functionality without explicit programming [34]. The primary purpose of machine learning algorithms is to build automated tools to access and use the data for training. The learning process starts with learning labeled data, also called training dataset. It can be a real-life experience, review, example, or feedback to recognize trends in the data to make better future decisions based on the user's input. The main objective of machine learning models is to learn automatically without any intervention from humans. Machine learning consists of three major kinds, used for numerous tasks.

For the last decade, researchers have been trying to make email communication better than today. Spam filtering of emails [35] is one of the most critical ways of protecting email networks. Many research articles have been published using various machine learning approaches to identify and process spam emails, but there are still some research gaps. Junk mail is one of the central, attractive research fields for filling the gaps [36]. For this reason, many spam classification studies have already been carried out using several methods to make email communication more trustworthy and valuable for users. That is why, this paper is presented to make a summarized version of different existing machine learning models and approaches that are being used for email spam detection. This paper also evaluates the most common machine learning approaches like KNN, SVM, random forest, and Naïve Bayes.

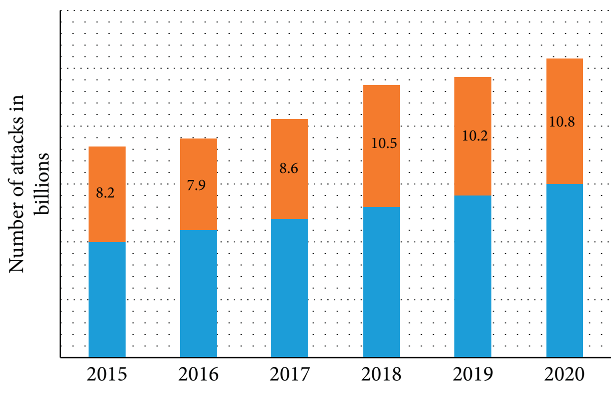


Figure 5: Number of attacks on IoT devices.

5.1. *Machine Learning-Based Spam Filtering Methods.*

Machine learning facilitates the processing of vast quantities of data. Though it typically provides faster and more accurate results to detect unwanted content, it can also require extra time and resources to train its models for a high level of performance. Integrating machine learning with AI and cognitive computing [37] can make handling massive amounts of data even more powerful. Figure 6 demonstrates various kinds of machine learning.

5.1.1. *Supervised Machine Learning.*

Supervised machine learning algorithms [18] are machine learning models that need labeled data. Initially, labeled training data is provided to these models for training, and after training models predict future events. In other words, these models begin with the analysis of an existing training dataset, and they generate a method to make predictions of success values. Upon proper training, the system can provide [38] the prediction on any new data related to the user's data at the training time. Furthermore, the learning algorithm accurately compares the output to the expected output and identifies errors to modify the model.

Supervised learning uses labeled data for training, and then it can predict the new data. This type of learning can be used in solving various problems, i.e., advertisement popularity, spam classification, face recognition, and object classification. The process of supervised learning is illustrated in Figure 7.

Some most commonly used supervised learning techniques are discussed as follows.

5.1.2. *Decision Tree Classifier.*

Decision tree classifier is a machine learning algorithm [39], which has been widely used since the last decade for classification. This algorithm applies a simple method of solving any problem of classification. A decision tree classifier is a collection of well-defined questions about test record attributes. Each time we get an answer, a follow up question is raised until a decision is not made on the record [40]. Tree-based decision algorithms define models that are constructed iteratively or recurrently based on the data provided. The decision tree-based algorithms goal is used to predict a target variable's

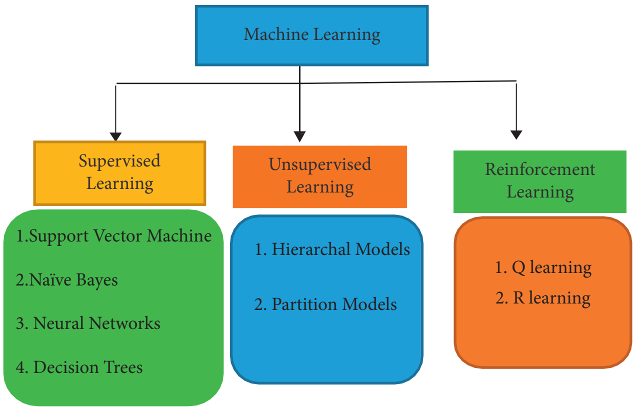


Figure 6: Types of machine learning.

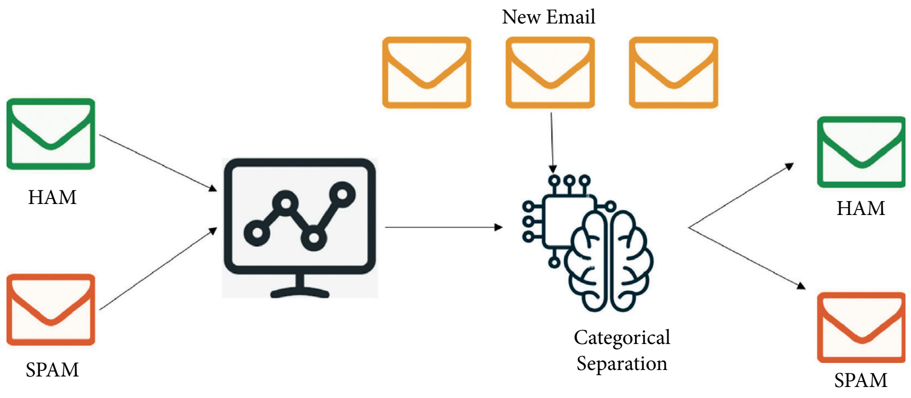


Figure 7: Process of supervised learning.

value on a given set of input values. This algorithm uses a tree structure to solve classification and regression problems [41]. Figure 8 shows the basic structure of the decision tree.

Some of the decision tree algorithms are the following:

(i) Random forest
(ii) Classification and regression tree (CART)
(iii) C4.5 and C5.0
(iv) Chi-square.

The following section deliberates some proposed email spam detection and prevention techniques by using decision tree algorithms.

DeBarr and Wechsler [42] discuss a spam filtering technique using random forest algorithms to classify spam emails and active learning to refine the classification [43]. They used the data of email messages from RFC 822 (Internet) [44] and divided each email into two sections. Then, they find term frequency and inverse document frequency of all features of each email (TF/IDF). For the training dataset, they select a set of emails with clustering to label the data. After considering the cluster prototype mails for training, they experiment with supervised machine learning algorithms: random forest, Naïve Bayes, support vector machine, and KNN [45]. The research results show that the algorithm “random forest” classifies data more efficiently with an accuracy of 95.2%.

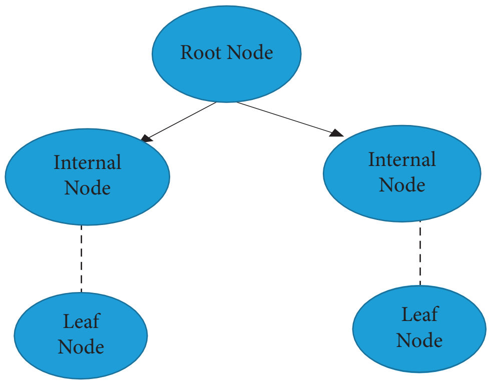


Figure 8: Structure of decision tree.

Takhmiri and Haroonabadi [46] present a different technique to detect spams using a fuzzy decision tree and the Naïve Bayes algorithm. They use the baking voting algorithm to extract patterns of spam behaviour. They do this because obvious characteristics do not exist in the real world. The cross-linking degree for explaining or describing characters is rational and neutral. Decision trees use fuzzy Mamdani rules for the classification of spam and ham email. Then, Naïve Bayes classifier [47] is used by them on the dataset. Finally, the baking method is used by dividing votes into smaller sections. This solution gives them an optimized weight that can be implemented on obtained percentages that achieve a higher accuracy level. The dataset used in this study contains 1000 emails, from which 350 (35%) were spam and 650 (65%) were ham.

Verma and Sofat [48] used supervised machine learning algorithm ID3 [49] to render the decision trees of the problem and the hidden Markov model [50] to measure the probabilities of events that could occur as a combination to classify the emails as junk mail or ham. The proposed model initially marks all emails as spam or legitimate by measuring each e-mail's total likelihood with the aid of subsequently classified email terms. After that, it makes the decision trees of emails one by one. The Enron dataset [51] is used in this study that contains 5172 emails. From all 5172 emails, 2086 were spam, while 2086 were legitimate emails. Their model can categorize the emails as spam and ham by using the feature set obtained by the Enron dataset. They got an 11% error by using the sklearn library's fitness function in the proposed model. Their model got 89% of accuracy results on the given dataset.

Li et al. [52] proposed an email-classification technique for IoT systems based on supervised machine learning. They use a multiview technique that focuses on the collection of richer information for classification. A double view dataset is created with internal and external feature sets. The proposed approach can be used in both labeled and unlabeled data and was evaluated on two datasets with a real network environment. The results of this study indicate that the multiview model can achieve more accuracy than simple email classification. In the end, the multiview model is compared with various existing models.

A spam filtering approach based on different decision tree algorithms is presented by Subasi et al. [40] to compare the accuracy and find the best one for their dataset. They implement classification and regression tree (CART), C4.5, REP tree, LAD tree, NBT, random forest, and rotation forest algorithm on the dataset to classify emails. Their results show that the proposed modified random forest model got the highest accuracy than other decision tree methods for publicly available datasets.

5.1.3. Support Vector Machine (SVM).

5.1.3. Support Vector Machine (SVM). The support vector machine (SVM) is an essential and valuable machine learning model [53]. SVM is a formally defined discriminative supervised learning classifier that takes labeled examples for training and gives a hyperplane as output, classifying new data [54]. A set of objects belonging to various class memberships are separated by decision planes. Figure 9 shows the classification concept of linear support vector machines. In the figure, some circles and stars are called objects. These objects can belong to any of two classes, i.e., the class of stars or dots. The isolated lines determine the choice of objects between green and brown objects. On the lower side of the plane, the objects are brown stars, and on the upper side of the plane all objects are green dots showing that two unique objects are classified into two different classes. If a new object black circle is given to the model, it will classify that circle into one of the classes according to the training examples provided in the training phase.

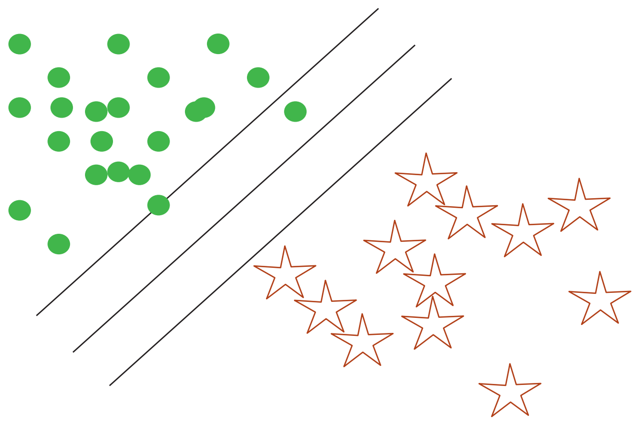


Figure 9: Support vector machine classification.

Banday and Jan [55] present research in which they define the procedure of statistical spam filters. They design those filters using Naïve Bayes, KNN, support vector machines (SVM), and regression trees [56]. They use all these supervised machine learning algorithms and evaluate the results based on precision, recall, and accuracy. Using these machine learning techniques, they found that classification and regression trees (CART) [57] and Naïve Bayes classifiers are the most effective algorithms for the dataset. This approach estimates that, during spam filtering, calculations of false positive are costlier than a false negative.

Zheng et al. [12, 58] present a procedure for detecting spammers and spam messages in any social network. Today, everyone uses social media, and many social media users spend a considerable amount of time communicating with their loved ones. The spammers take advantage of various social media networks and users' posts to send malicious content, advertisements, information, etc., into the social media user's profiles. So, this paper discusses how to detect those posts or malicious content on social media platforms. Their study uses the Sina Weibo social network [59] and machine learning algorithm support vector machine (SVM) for the detection of spammers. The dataset that was used in this study was 16 million messages that were collected from several users. They used 18 features as a feature vector set. The clients of the networks were divided into two categories, legitimate users and spammers. 80% of data was used for the model's training, while 20% was used for testing. For better accuracy, they used 1 : 2 between spammers and nonspammers of the training dataset. With this ratio, the proposed model gives an accuracy level of 99.5% for classifying spammers and nonspammers [60].

A novel fitness framework based on IoT-enabled blockchain technology and machine learning techniques is presented by Jamil et al. [10]. Their proposed model is composed of two modules. The first one is a blockchain-based network used for the security of sensing devices and an intelligent contract-enabled relationship and an inference engine that uncovers hidden insights and usable information from IoT and user device data. The improved smart contract gives users a useful application that allows real-time monitoring, more control, and quick access to several devices distributed across various domains. The inference engine module attempts to uncover underlying patterns and usable information from IoT environment data, assisting in effective decision-making and providing convenient services. Their proposed model can be used to improve system throughput and resource usage, according to their findings. The proposed system in this article may be used in various fields, including healthcare and smart businesses.

Olatunji [61] developed a spam filtering tool using support vector machine and extreme learning machine algorithms. He used the standard dataset for the development of the spam detection model. SVM got an accuracy of 94.06% in his work, and the extreme learning machine (ELM) model got a 93.04% accuracy level, suggesting just 1.1% performance improvement that SVM achieved over ELM. He indicated that SVM's improvement over ELM accuracy is marginal. It implies that, in situations where detection time is critical, as in real-time systems, the ELM spam detector should be given preference over SVM spam detection. Although SVM got a higher accuracy level in his research, it takes more time for training than the ELM system. Tretyakov [62] also discussed various machine learning techniques for email spam filtering. This paper compared the precision results between false positives and precision results after eliminating false positives. They show the result after eliminating false positives, which were more accurate and reliable than before.

5.1.4. Naïve Bayes Classifier (NB). The Naïve Bayes classifier [47] is based on the Bayes theorem. It assumes that the predictors are independent, which means that knowing the value of one attribute impacts any other attribute's value. Naïve Bayes classifiers are easy to build because they do not require any iterative process and they perform very efficiently on large datasets with a handsome level of accuracy. Despite its simplicity, Naïve Bayes is known to have often outperformed other classification methods in various problems.

Rusland et al. [63] present research on email spam filtering and perform the analysis using a machine learning algorithm Naïve Bayes. They used two datasets evaluated on the value of accuracy, F-measure, precision, and recall. As we know, Naïve Bayes uses probability for classification, and the probability is counting the frequency and combination of values in a dataset. This research uses three steps for the filtration of emails, i.e., preprocessing, feature selection, and, at last, it implements the features by using the Naïve Bayes classifier. The preprocessing step removes all conjunction words, articles, and stop words from the email body. Then, they used the WEKA tool [64] and made two datasets called spam data and spam base dataset. The average accuracy was 89.59% using two datasets, while the spam data got 91.13% accuracy. The spam base dataset got an accuracy of 82.54%. The average precision results for spam data were 83%, while, for spam base, the precision result was 88%. They claimed that the Naïve Bayes classifier performs better on spam base data as compared with spam data.

Arif et al. [11] presented an article on machine learning-based spam detection techniques for IoT devices. They used five ML models and analyzed their results using various performance metrics. A large number of input features were used for the training of proposed models. Each model calculates a spam score based on the input attributes. This score represents the trustworthiness of an IoT device based on a variety of factors. The suggested approach is validated using the REFIT smart home dataset. They claim that their proposed system can detect spam better than currently used spam detection systems. Their work can be utilized in smart homes and other places where intelligent devices are used.

Kumar et al. [14] discussed email spam detection using various ML algorithms. Their article explores ML methods and how to implement them on datasets. The optimal algorithm for email spam detection with the highest precision and accuracy is identified from various ML algorithms. They concluded that the Multinomial Naïve Bayes algorithm produces the best results, but it has limitations due to class-conditional independence, which causes the machine to misclassify some inputs. Ensemble models come after Multinomial Naïve Bayes with the best and reliable results in this study. The proposed system in this study can only detect spam from the body of emails.

Singh and Batra [65] proposed a semisupervised machine learning technique for spam detection in social IoT platforms. They used an ensemble-based framework that is consists of four classifiers. The architecture is based on the use of probabilistic data structures (PDS) such as Quotient Filter (QF) to query the database of URLs, spam users, databases of spam keywords, and Locality Sensitive Hashing (LSH) for similarity search. The proposed model minimizes, so it decides by an adaptive weighted voting approach based on each classifier's output. The hybrid sampling technique minimizes the computational efforts, which sample the data according to each classifier. This study indicates that the proposed model can be used for spam detection on large datasets. The proposed model's efficiency was evaluated by comparing PDS with standard data models and the typical evaluation metrics, including accuracy, recall, and F-score.

5.1.5. Artificial Neural Networks. An artificial neural network (ANN) is a computational model based on the functional aspects of biological neural networks, also known as the neural network (NN) [66]. Many sets of neurons are joined in a neural network, and information is interpreted using a computational approach connection. In most situations, an ANN is an adaptive system, which changes its structure depending on external or internal information flowing through the network during the learning phase. Current neural networks are nonlinear approaches to statistical data processing. These are commonly used when there are complex relationships between inputs and outputs or unusual performance patterns [6]. Figure 10 shows the basic structure of the neural network.

The following section elaborates some proposed email spam detection and prevention techniques by using neural networks.

Xu et al. [67] present a method for the detection of spam in online social networks. Their work focuses on the

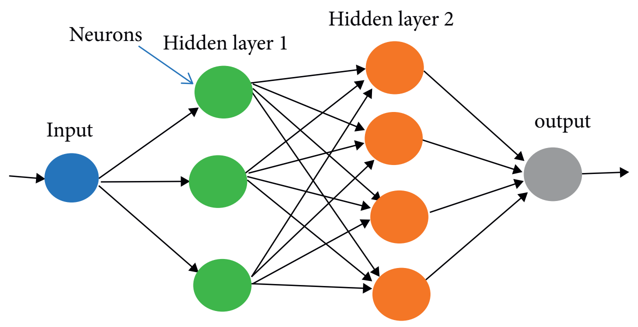


Figure 10: Basic structure of neural network.

combination of spam messages in one social network to another social network. By using Twitter, they gathered 1937 spam and 10943 ham tweets for processing. They also used 1338 spam posts and 9285 ham posts. In TSD, 75.6% of tweets contained URL links for spam tweets, while 24.4% contained different words. Out of 10942 ham tweets, 62.9% contained URL links and words, while 37.1% had only words. For the spam posts of FSD, 32.8% of posts consist of different web links, and the remaining 67.2% of spam posts contain only words [68]. Of 9285 ham posts, 95.1% have web links, and the other 4.9% consist of words. They used the top twenty feature words from Facebook spam data and Twitter spam data. They divide the TSD and FSD into two sets, i.e., training dataset and testing dataset. These datasets were used to train various machine learning classifiers like Naïve Bayes, random forest, logistic regression random tree, and Bayes Net. After analyzing the accuracy of different classifiers, they combine the spam dataset of Facebook into the training dataset of Twitter and the spam dataset of Twitter into the training dataset of Facebook. Then, they used the combined dataset for the training and testing of classifiers. In the end, they compare the results of classifiers on the above-mentioned social networks after measuring the precision, accuracy, recall, and F-1 measure. They found that the accuracy of combined datasets was higher than that of other datasets [68, 69].

Guo et al. [70] proposed a spammer detection technique using a collaborative neural network in IoT applications. They present a novel spam detection mechanism called Cospam for IoT applications. At first, the user and contents of speech at different timestamps are viewed as feature sequences. In the second step, a collaborative neural network model is used. The collaborative model consists of three models: (1) Bi-AE model, (2) GCN model, and (3) LSTM model. These models are used for the identification of the nature of the user. In the end, a series of experiments were conducted for the evaluation of the proposed technique. The proposed model was able to obtain 5% more accuracy than existing spammer detection approaches. Cospam consumes more time than existing techniques because of a large number of parameters.

Makkar and Kumar [71] proposed a deep learning model for web spam detection in an IoT environment. Their system enhances the cognitive ability of search engines for the detection of web spam. This model removes spam pages with the help of a web page rank score calculated by a search engine. Their framework uses the extensive features of deep learning. The first time in which the LSTM model was used to detect spam is used for many problems like weather forecasting. In this study, the proposed model is compared with ten different machine learning models. The WEB-SPAM-UK 2007 standard dataset is used in this study. The preprocessing of the dataset is done by a novel technique called "Split by Oversampling and Train by Underfitting." The accuracy of the proposed model was 95.25%. After the optimization of the system, the proposed model got an accuracy of 96.96%.

Zavvar et al. [72] present a paper on spam detection by considering combined particle swarm optimization and neural networks to select features. They also used SVM for classifying and separating spam. They compared the proposed approach with other approaches such as a self-organizing map and k-means data grouping based on the region under curve parameters. This article uses the UCI base dataset to evaluate spam classification and provide a PSO-ANN and ANFIS algorithm-based approach for spam detection. Seventy percent of data was used for training, and 30 percent was used for testing the models. RMSE, NRMSE, and STD principles were analyzed and got 0.08733, 0.0185, and 0.08742 results in the testing phase. The results show that the proposed method has good accuracy and performance for detecting spam emails. Table 2 summarizes supervised machine learning techniques presented for spam detection.

5.1.6. Discussions and Learned Lessons. Supervised machine learning techniques, i.e., decision trees, random forests, support vector machines, and artificial neural networks, can be used for email spam detection or filtering. Support vector machines classify different objects by using the idea of the hyperplane. Objects are classified into two classes. If a new object is given to the model, it will be classified into one of both classes. Zavvar et al. [12], Garavand et al. [72], and Idris et al. present different techniques for spam detection using the support vector machine (SVM) model. They got a good accuracy level on different spam datasets. Olatunji et al. [73] used the support vector machine and extreme learning machine algorithms on the standard dataset and got 94.06% accuracy using the support vector machine. In their system, extreme learning machines perform better than SVM but take more time, so a time-consuming ELM performs better than SVM. Zheng et al. got the highest accuracy level using Weibo social network dataset. They use two types of features, i.e., content base and user behavior base, to classify spammers and nonspammers. Naïve Bayes classification is another supervised machine learning technique, which predicts some events based on its naïve theorem. Naïve Bayes classifiers are quite simple, and they do not use an iterative process; they perform very efficiently on large datasets with a handsome level of accuracy. Hijawi et al. [41] use the Naïve Bayes network for the detection of spam. They did not get outstanding results using the spam assassin dataset as their accuracy level was only 89%. Another technique which is widely used in the last decade is decision

Table 2: Comparison of supervised techniques for spam filtering.

Authors	Algorithm	Dataset	Accuracy	Advantages	Limitations
DeBarr and Wechsler [42]	Random forest	Custom collection	95.2%	They got good accuracy with multiple trees	The dataset that they used was not a standard dataset
Rusland et al. [63]	Modified Naïve Bayes with selective features	Spam base and spam data	88% on spam base 83% on spam data	Selective features are taken that consume less time	They got less accuracy, and their model was not much intelligent
Halu zu et al. [67]	Bayes Net, SVM, and NB	Twitter and Facebook dataset	90% using SVM	They used the combined dataset for the training and testing of classifiers	Multiple algorithms and a combined dataset system take more training time
Hijawi et al. [41]	(MLP), Naïve Bayes, random forest, and decision tree	Spam assassin	99.3% using random forest	They use a list of most common spam features that improve the spam detection rate	They use a significant corpus of 6050 emails, but they use a small number of features extracted from the corpus
Banday and Jan [55]	Naïve Bayes, K-nearest neighbor, SVM, and additive regression tree	Real-life dataset	96.69% using SVM	They make a spam filter based on 8000 real-life spam emails	Their model is not so effective as spammers continuously change the characteristic that they used for making spam filter
Verma and Sofat [48]	ID3 algorithm hidden Markov	Enron dataset	89%	They use a preclassified dataset that uses less time in processing	Their model got an 11% loss that is not too good for spam filters
Subasi et al. [40]	CART, C4.5, REP tree, LAD tree, and NBT	UCI dataset	95.1%	They used 10-fold cross-validation that helps in better evaluation	Less number of features used
Zheng et al. [12]	SVM	Weibo social network data	99.5%	They use both user content and behavior features for detecting spammers	Feature extraction is based on statistical analysis and manual selection
Garavand et al. [72]	SVM, deep learning, and particle swarm optimization	Standard datasets from UCI 70% education data	93% using the support vector machine	They use deep learning models for feature extraction	The neural networks take massive time for training for the extraction of features
Olatunji et al. [5]	ELM and SVM classifier	Enron dataset	94.06 using SVM	They got a high accuracy level as compared to previous studies on the same dataset	For SVM, it takes more time than ELM to gain the accuracy level claimed in the paper
Jamil et al. [10]	SVM, KNN, DT, and LR	Health fitness data	92.1 using SVM	Smart contract-enabled blockchain technique is used that makes the system more secure	Interoperability of proposed model with IoT framework is not evaluated
Arif et al. [11]	XGBoost, bagged model, and generalized linear model with stepwise feature selection	Smart home dataset	91.8 using generalized linear model with stepwise feature selection	PCA was applied that enhances the accuracy of the system	Climatic and surrounding features of IoT devices are not considered

tree. These decision algorithms define models that are constructed iteratively or recurrently based on the data provided. The decision tree-based algorithms goal is to predict a target variable's value on given set of input variables. Subasi et al. [40] used different decision tree-based algorithms for spam detection on the UCI machine learning platform dataset. They used 10-fold cross-validation for the evaluation of decision tree classifiers. They use open-source Weka tools for the development of the model. DeBarr and Wechsler [42] used a tree-based random forest algorithm for email spam detection and active learning for refining the classification. They used the data of email messages from RFC 822 (Internet) and got the highest accuracy level of 95.2% by using the dataset's custom collection of emails. In all supervised machine learning techniques, Zheng et al. [12] got the highest accuracy level among all researchers using the support vector machine (SVM) technique for email spam detection.

5.2. Unsupervised Machine Learning. Unsupervised machine learning algorithms are used when we do not have labeled data [74]. Unsupervised learning explores how programs can explain a hidden structure by inferring a feature from unlabeled data [75]. The machine does not evaluate the appropriate output but examines the data and can draw inferences from datasets to explain hidden constructs from unlabeled data. Unsupervised learning works on unlabeled data and makes clusters of the data based on the features of that data. This type of learning can be used for various problems like Recommender Systems, identifying Buying Habits, Grouping User Logs, dimensionality reduction, etc. The process of unsupervised learning is illustrated in Figure 11.

Clustering is the main application of unsupervised learning that has two main types. Different clustering techniques are discussed as follows.

5.2.1. Hierarchical Clustering. Hierarchical clustering identifies clusters with a hierarchy achieved either by iteratively combining smaller clusters into a more significant cluster or by splitting a more massive cluster into smaller clusters. This cluster hierarchy, generated through a clustering algorithm, is called a dendrogram [76]. A dendrogram is one way of representing the hierarchical clusters. The user can understand different clusters based on the level at which the dendrogram is defined. It uses a similarity scale representing the distance between the clusters grouped from the massive cluster. A dendrogram is a visual representation of hierarchical clustering that is illustrated in Figure 12.

5.2.2. Partitional Clustering. A partitional clustering divides a single set of data objects into nonoverlapping subsets (clusters) so that each data object is in only one subset [77]. Partitional clustering algorithms make different partitions of data and then evaluate the required results based on some criteria. Figure 13 illustrates the basic structure of partitional clustering algorithms. In Figure 13, partitions (A, B, and C) are created based on some characteristics. Partitional clustering breaks down a dataset into a collection of clusters of disjoints. The partitioning technique forms different partitions of data by using the formula K (N/K); each partition represents a cluster based on a set of N points in the data, that is, by fulfilling the following conditions:

(1) Each class contains one point or more
(2) Each point comes as part of exactly one group

Let us discuss some work on filtering email spam using unsupervised machine learning techniques.

Sharma and Rastogi [78] propose a strategy using unsupervised techniques. They performed various experiments on email spam datasets. After data gathering, they use the k-means clustering model for the clustering of emails. They use various distance measures for this purpose. The study's findings show that the proposed model performs well and cluster spam and ham emails are efficient.

Tan et al. [79] developed a reliable model for spam detection. First, they present a Sybil defense-based automated spam detection scheme called SD2, which considerably outperforms current techniques by considering the social network relationship. They further developed an unsupervised spam detection system called UNIK to address increased spam attacks effectively. Instead of directly detecting spammers, UNIK operates by intentionally eliminating nonspammers from the network. They used the social graph as well as the user-link graph for the detection of the spammer. UNIK's fundamental basis is that spammers actively change their patterns to avoid detection, while nonspammers are not expected to do so. Therefore, we have a reasonably nonvolatile pattern. When tested on a broad network platform, UNIK has a similar performance as SD2 and substantially beats SD2 as spam attack rates go up. They evaluate several known spam activities in the social network platform by the identification of UNIK. Their proposed system, UNIK, can be used for email spam classification. The result shows that various spammer clusters exhibit different characteristics, suggesting the instability of spamming and UNIK's ability of automatically extracting junk mail signatures.

Ahmed [80] used an improved digest algorithm with DBSCAN clustering to classify spam emails. They create a different digest (parts) of emails before clustering. Their proposed model has two key steps. When the system receives emails, it first enters the digest generation phase, where an improved digest algorithm processes it, and the output is the set of digests of each email. These digests are then given to the clustering algorithm, i.e., DBSCAN, in the next phase. In

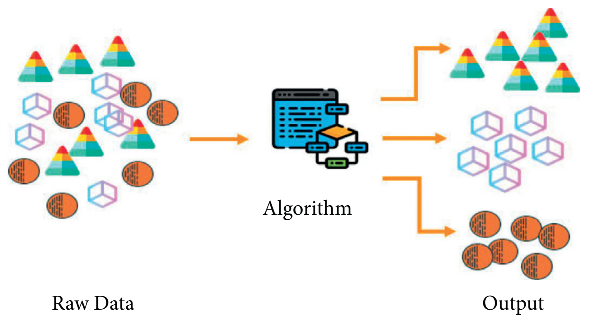


Figure 11: Process of unsupervised learning.

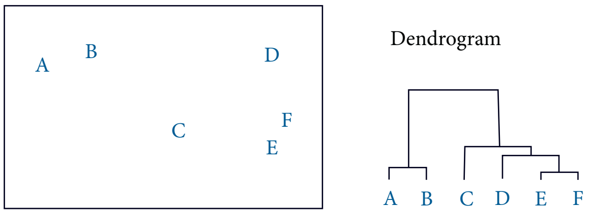


Figure 12: Structure of dendrogram.

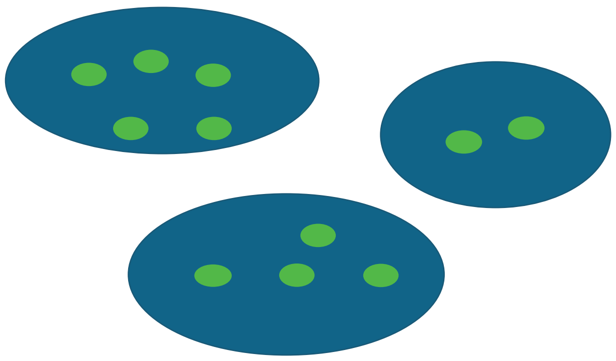


Figure 13: Partitioned clustering structure.

the clustering phase, similar emails are classified in the clustering process in a cluster of spam mails based on similarities among their digests, where mails that do not look like any other digest are considered noise and not clustered. Such emails that are not clustered are standard (ham) emails.

Using unsupervised artificial neural networks (ANNs), Cabrera-León et al. [81] propose a hybrid antispam filter. Their method contains two main steps. The first step is preprocessing of content, and the second one is actual processing. Each step is based on various models of computation. These models are "programmed and neural (using Kohonen SOM) [55]. This proposed system used the Enron dataset for ham or legitimate emails, while for spam emails they used two distinct sources. The first phase preprocessing was done based on thirteen (13) thematic features found in spam and ham emails. The terms frequency (TF) and inverse term frequency (IDF) were used in their system for the sake of feature extraction. Their results were the same as those of other researchers for the same dataset since they use distinct machine learning techniques and attributes. They evaluated their system with various datasets, defined by interdependent origins, ages, users, and forms like image spam samples. Their system got an accuracy level between 75% and 96%. They show that model performance degradation can vary by variations, in datasets, especially in dates. This phenomenon is known as "topic drift." Generally, it affects all classifiers, but it more affects those classifiers that use offline learning. The same case is with adversarial machine learning problems like spam filtering. Their method is robust to phrase obfuscation, which is commonly used in spam content. It was also independent of the need to use lemmatization or stemming.

Sasaki and Shinnou [82] introduce a new approach for spam detection using the vector-space model of content clustering. Their system automatically calculates disjoint clusters using a spherical k-means technique for all spam and nonspam emails. It collects centroid vectors of clusters for the extraction of vector definition. Each centroid is labeled with spam and nonspam to measure several spam emails in the clusters. The system measures the cosine similarity between the current mail vector and the centroid vector as a new email arrives. Eventually, the new mail is assigned the label of the most appropriate cluster. They obtain several kinds of spam and nonspam email topics by using the proposed approach and effectively identifying the spam emails. They introduce the spam detection framework in this paper and demonstrate the research outcomes utilizing the series of Ling-spam datasets. They got 98.06% accuracy with their model.

Narisawa et al. [83] suggest an unsupervised approach for detecting spam documents from several documents relying on string equivalence. They provide three metrics to quantify a string's alienation, which means how distinct they are inside the documents from other substrings. In their proposed model, a document labeled as spam includes a substring with a significant alien degree in an equivalence class. The proposed approach was unsupervised, independently of language, and scalable. Japanese web forum data were used for computational experiments to show the proposed approach's performance on real data. Table 3 presents comparison of unsupervised learning techniques used for spam filtering.

5.2.3. Discussion and Learned Lessons. Several unsupervised machine learning models are being used for email spam detection and filtering. Hierarchical clustering and partitioning clustering are commonly used clustering techniques. Ahmed [80] used DBSCAN clustering and an improved digest algorithm to classify emails. He used the spam assassin dataset for the development of his model. This approach significantly enhances filtering accuracy by 30 percent against the newly proposed algorithms and increases spam detection tolerance against increased spammer's obfuscation effort while maintaining successful email detection at a comparable level of older filtering methods.

Sharma and Rastogi [78] used a machine learning algorithm (k-mean clustering) with local concentration-based content extraction for spam detection and got a handsome accuracy level. Cabrera-León et al. [81] used an artificial neural network that contains two necessary steps. In the first step, they do preprocessing and then in the second step they process cleaned data for computing the results. These steps are based on distinct models of computation. Its accuracy was 95%. Narisawa et al. [83] introduced an unsupervised approach to identify a spam document from a collection of documents based on string equivalence. This solution was a language-independent and scalable method for spam detection. It was tested on the Japanese web forum. Among all the researchers, Sharma Rastogi [78] and Ahmed et al. got the highest accuracy level using DBSCAN and K-mean algorithm, respectively, for the email spam detection. Ahmed [80] used spam assassin dataset for the implementation of his model.

5.3. Reinforcement Machine Learning. Reinforcement learning is another type of machine learning which works on reward taken from its environment. It takes suitable actions to make or get the maximum reward in a given situation [84]. Many machines and software employ it to find the optimal path to take in a specific situation.

The main difference between supervised and reinforcement learning is that supervised learning needs training data with correct labels. Simultaneously, there is no correct label in reinforcement learning, but the agent decides what to do to perform the given task. The agent is bound to learn from its experience if there is no training dataset [85]. Figure 14 illustrates the simple reinforcement learning process in which an agent passes an action to the environment. The environment sends back the reward of action and state to the agent. Let us discuss some research work done on email spam detection using reinforcement learning.

Chiu et al. [86] propose an alliance-based approach to classify, identify, and exchange relevant information on spam email contents. Their spam filter consisted of a rough set theory, a machine learning classifier (XCS), and a genetic algorithm. They used several metrics to evaluate the model results. From their paper, two main conclusions can be

Table 3: Comparison of unsupervised learning techniques used for spam filtering.

Authors	Algorithm used	Dataset	Accuracy (%)	Advantages	Disadvantages
Ahmed [80]	Improved digest and DBSCAN	Spam assassin	96.7	The proposed model divides email into fixed-length strings before clustering, which gives better accuracy	The speed of the proposed model depends upon the length of strings
Sharma and Rastogi [78]	K-means clustering	UCI dataset	92.76	It is discretized using supervised attribute filters and also used 10-fold cross-validation	While comparing multiple algorithms, results take a handsome amount of time
Cabrera-León et al. [81]	Unsupervised artificial neural networks	Enron email	95	The system is robust to word obfuscation, used in spam, independently of the use of stemming or lemmatization	Bad false negative and false positive rate are around 11 and 4%, respectively
Sasaki and Shinnou [82]	Spherical k-means algorithm	Ling-spam	96.04	The model uses various contents of spam emails	Updating spam contents and relevance feedback is not in the proposed model
Narisawa et al. [83]	Equivalence relations of strings	Japanese web forums	95	The model was scalable and language-independent	As the model uses N-Gram of documents, so results depend on the value of "*n*"
Tan et al. [79]	UNIK and SD2	Social network sites data	93	It is highly robust to an increased level of spam attacks	The proposed system cannot handle short URLs

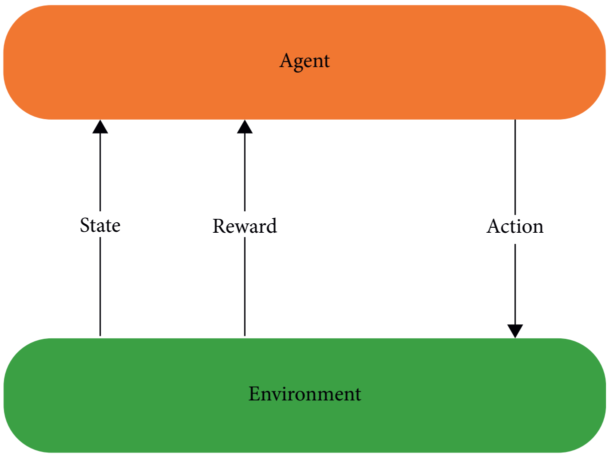


Figure 14: Basic structure of reinforcement learning.

drawn, and they are given as follows: The spam filter is based on a combination of rough set theory, genetic algorithm, and machine classifier XCS. Many metrics are used to assess spam mails filtering results by an alliance-based approach and provide a reasonable output indicator. They may draw two key conclusions which are the following:

(a) The rules that have been shared from many other email servers do help the spam filter to block more spam emails than before

(b) A blend of several techniques increases precision and decreases false positives for the spam detection task

5.3.1. Discussion and Learned Lesson. Reinforcement machine learning is a type of machine learning in which an agent communicates with its environment by producing behaviors and generating results or rewards. This method allows the software agents to find an optimal solution in a specific domain. An agent acts with the environment and gets the error or reward. Chiu et al. [86] used this approach on spam emails. The spam filter was built based on a mixture of rough set theory, genetic algorithm, an XCS classifier system, and good performance measure. Lai et al. [87] propose a practical approach for spam detection using rough set theory and XML format. They use reinforcement learning for the management exchange of spam rules. They suggest that outdated rules should be discarded as spammers are constantly changing their methods for doing spam. They further conclude that the spam filter can block more spam emails than a standalone system by sharing spam rules between the email servers. Samadi et al. [85] and Dou et al. [88] also used reinforcement learning techniques to detect spam and spammers.

6. Overall Insights of the Machine Learning Algorithms for Spam Detection

Figure 15 illustrates the percentage of work on email spam detection discussed in this survey. After discussing the literature, we observed that most of the datasets used to train, test, and implement different models are synthetically created. There is a lack of examples for analysis and the complexity of labeling all the supervised model data. So, the classifiers' results are not 100% trustworthy because of the synthetic datasets used for the models' training. These are not representative of real-world spam reviews as vast numbers of machine learning models are currently used for email spam detection or filtering. The three learning algorithms, logistic regression, Naïve Bayes, and support vector machine (SVM), are widely used, and they outperform the other learning algorithms in most of the discussed studies.

SVM generally gives the best performance; Naïve Bayes and logistic regression commonly beat it. But SVM should not be considered merely as the best algorithm since it is not compared to all others. Multiple learning models on various datasets should be evaluated in future studies using several different feature engineering methods. This survey paper elaborates the existing machine learning-based spam filtering techniques and models by exploring and observing numerous methods. The conclusions are discussed by the overview of several spam filtering techniques and summarizing the accuracy of different proposed approaches based on various parameters. We conclude that all the spam filtering techniques perform well. Some have outstanding results, while some are trying to use other methods to increase the accuracy level. Though all are effective, the spam filtering system still lacks some, which are the primary concern for researchers. They are trying to generate next-generation spam filtering processes that can work on multimedia data and prominently filter spam emails. Table 4 is reproduced from Awad and Elseuofi [13]. Table 4 summarizes the performance of various machine learning models on 100 selected features.

7. Research Gaps and Open Research Problems

This section discusses the research gaps and open research problems of the spam detection and filtration domain. In the future, experiments and models should be trained on real-life data rather than manually created datasets, because, in the various article, the models trained on artificial datasets perform very poorly on real-life data. Currently, supervised, unsupervised, and reinforcement learning algorithms are used for spam detection, but we can get higher accuracy and efficiency by using hybrid algorithms in the future. Feature extraction can be improved in the future by using deep learning for feature extraction. Using clustering techniques for spam filtering relevance feedback using dynamic updating can better cluster spam and ham. Along with machine learning, blockchain models and concepts can also be used for email spam detection in the future. Experts in linguistics and psycholinguistics can collaborate in the future for manual annotation of datasets, which will result in the development of effective and standard spam datasets with high dimensionality. In future, spam filters can be designed with faster processing and classification accuracy using Graphics Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs), which offer low energy consumption, flexibility, and real-time processing capabilities. Moreover, future research should concentrate on the availability of standard labeled datasets for researchers to train classifiers and the addition of more attributes to the dataset to improve the accuracy and reliability of spam detection models, such as the spammer's IP address and the location. The following are some other future research directions and open research problems in the domain of spam detection.

(i) Some studies considered header, subject of the email, and message body as a feature for spam classification. While these features are not enough for fully accurate results, manual feature selection and features should also be.

(ii) Almost all researchers presented their results based on accuracy, precision, recall, etc., while the time complexity of machine learning models should be considered an evaluation metric.

(iii) Some researchers show promising results in the process of feature extraction using a bag of words. They claim that the email header is as important for spam detection as the content of the body. So, deep feature extraction of the header line should be considered.

(iv) Fault tolerance, self-learning, and quick response time can be better by using comprehensive feature engineering and an accurate preprocessing phase.

(v) Deep learning models with dynamic updating of feature space are needed to implement for better spam classification. Most of the current filters cannot update their feature space.

(vi) The security of spam detection and filtration system is needed for better accuracy and reliable results.

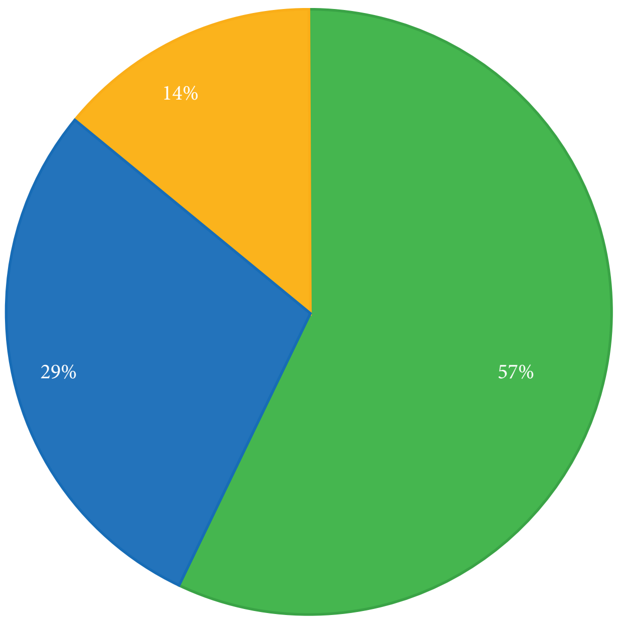


FIGURE 15: Ratio of machine learning techniques for email spam detection.

TABLE 4: Performance of various machine learning models on 100 selected features.

Algorithm	Recall	Precision	Accuracy
Naïve Bayes	98.46	99.66	99.46
SVM	95.00	93.12	96.90
KNN	97.14	87.00	96.20
Neural network	96.92	96.02	96.83
AIS	93.68	97.75	96.23
Decision tree	94.36	91.35	93.55

(vii) The false positive rate of many models is still higher than required. It must be reduced to the smallest possible value.

(viii) Few spam filters work on image spam detection and filtration. Expert spammers also use images for spam messages, so it should be considered in detecting spam.

(ix) Real-time spam classification is much needed as most of the proposed models cannot work on real-time data.

(x) Labeled data is one of the major issues in spam detection. There are a few new labeled and up-to-date datasets for this purpose.

(xi) Multilingual spam detection is also a significant research area that can be explored for better spam detection systems. There is less work done on multilingual spam detection using deep learning techniques.

(xii) Semisupervised and federated learning techniques can be used to enhance spam detection in various IoT and email frameworks.

(xiii) A combination of linguistic features for the spam detection approach can also be explored.

(xiv) The research community ignores the identification of spammers and spammer networks.

(xv) Many researchers manually annotate data, using spam features that they think to be accurate. As a result, the evaluation results of the detection systems that they propose are doubted. The ideal solution for this problem has yet to be discovered.

(xvi) There is a lack of a robust method of dealing with challenges regarding the spam filters' security. An attack of this nature can be a casual, exploratory, or targeted attack. The deep learning techniques with blockchain technology can be used for this purpose.

8. Challenges of Spam Detection

Some critical challenges faced by spam filters are discussed as follows:

(i) The growing amount of data on the Internet with various new features is a big challenge for spam detection systems.

(ii) Features' evaluation from several dimensions such as temporal, writing styles, semantic, and statistical ones is also challenging for spam filters.

(iii) Most of the models are trained on balanced datasets, while self-learning models are not possible.

(iv) Many spam detection models face adversarial machine learning attacks that will decrease their effectiveness. Adversaries can throw a variety of attacks during the training and testing of ML models. Adversaries can harm training data to cause a classifier to classify the data incorrectly (poisoning attack), create unfavorable samples during testing to evade detection (evasion attack), and obtain sensitive training data via a learning model (privacy attack)

(v) Deep fake is another big challenge that is being faced by spam detection systems. To generate, modify, and style pictures and videos, neural network models such as GPT-2,3 and image generation models like BigGAN, StyleGAN, and CycleGAN are adopted. Deep fakes can be used to disseminate false information.

9. Conclusion

In the last two decades, spam detection and filtration gained the attention of a sizeable research community. The reason for a lot of research in this area is its costly and massive effect in many situations like consumer behavior and fake reviews. The survey covers various machine learning techniques and models that the various researchers have proposed to detect and filter spam in emails and IoT platforms. The study categorized them as supervised, unsupervised, reinforcement learning, etc. The study compares these approaches and provides a summary of learned lessons from each category. This study concludes that most of the proposed email and IoT spam detection methods are based on supervised machine learning techniques. A labeled dataset for the supervised model training is a crucial and time-consuming task. Supervised learning algorithms SVM and Naïve Bayes outperform other models in spam detection. The study provides comprehensive insights of these algorithms and some future research directions for email spam detection and filtering.

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 research was supported by Taif University Researchers' Supporting Project number TURSP-2020/314, Taif University, Taif, Saudi Arabia.

References

[1] H. Faris, A. M. Al-Zoubi, A. A. Heidari et al., "An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks," *Information Fusion*, vol. 48, pp. 67–83, 2019.

[2] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering," *Artificial Intelligence Review*, vol. 29, no. 1, pp. 63–92, 2008.

[3] A. Alghoul, S. Al Ajrami, G. Al Jarousha, G. Harb, and S. S. Abu-Naser, "Email classification using artificial neural network," *International Journal for Academic Development*, vol. 2, 2018.

[4] N. Udayakumar, S. Anandaselvi, and T. Subbulakshmi, "Dynamic malware analysis using machine learning algorithm," in *Proceedings of the 2017 International Conference on Intelligent Sustainable Systems (ICISS)*, IEEE, Palladam, India, December 2017.
[5] S. O. Olatunji, "Extreme Learning machines and Support Vector Machines models for email spam detection," in *Proceedings of the 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, IEEE, Windsor, Canada, April 2017.
[6] J. Dean, "Large scale deep learning," in *Proceedings of the Keynote GPU Technical Conference*, San Jose, CA, USA, 2015.
[7] J. K. Kruschke and T. M. Liddell, "Bayesian data analysis for newcomers," *Psychonomic Bulletin & Review*, vol. 25, no. 1, pp. 155–177, 2018.
[8] K. S. Adewole, N. B. Anuar, A. Kamsin, K. D. Varathan, and S. A. Razak, "Malicious accounts: dark of the social networks," *Journal of Network and Computer Applications*, vol. 79, pp. 41–67, 2017.
[9] A. Barushka and P. Hájek, "Spam filtering using regularized neural networks with rectified linear units," in *Proceedings of the Conference of the Italian Association for Artificial Intelligence*, Springer, Berlin, Germany, November 2016.
[10] F. Jamil, H. K. Kahng, S. Kim, and D. H. Kim, "Towards secure fitness framework based on IoT-enabled blockchain network integrated with machine learning algorithms," *Sensors*, vol. 21, no. 5, p. 1640, 2021.
[11] M. H. Arif, J. Li, M. Iqbal, and K. Liu, "Sentiment analysis and spam detection in short informal text using learning classifier systems," *Soft Computing*, vol. 22, no. 21, pp. 7281–7291, 2018.
[12] X. Zheng, X. Zhang, Y. Yu, T. Kechadi, and C. Rong, "ELM-based spammer detection in social networks," *The Journal of Supercomputing*, vol. 72, no. 8, pp. 2991–3005, 2016.
[13] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, Article ID 102419, 2020.
[14] N. Kumar and S. Sonowal, "Email spam detection using machine learning algorithms," in *Proceedings of the 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 108–113, Coimbatore, India, 2020.
[15] I. Santos, Y. K. Penya, J. Devesa, and P. G. Bringas, "N-grams-based file signatures for malware detection," *ICEIS*, vol. 9, no. 2, pp. 317–320, 2009.
[16] S. Cresci, M. Petrocchi, A. Spognardi, and S. Tognazzi, "On the capability of evolved spambots to evade detection via genetic engineering," *Online Social Networks and Media*, vol. 9, pp. 1–16, 2019.
[17] A. J. Saleh, A. Karim, B. Shanmugam et al., "An intelligent spam detection model based on artificial immune system," *Information*, vol. 10, no. 6, p. 209, 2019.
[18] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: a review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.
[19] E. Blanzieri and A. Bryl, *E-mail Spam Filtering with Local SVM Classifiers*, University of Trento, Trento, Italy, 2008.
[20] H. Bhuiyan, A. Ashiquzzaman, T. Islam Juthi, S. Biswas, and J. Ara, "A survey of existing e-mail spam filtering methods considering machine learning techniques," *Global Journal of Computer Science and Technology*, vol. 18, 2018.
[21] A. Asuncion and D. Newman, "UCI machine learning repository," 2007, https://archive.ics.uci.edu/ml/index.php.
[22] T. Vyas, P. Prajapati, and S. Gadhwal, "A survey and evaluation of supervised machine learning techniques for spam e-mail filtering," in *Proceedings of the 2015 IEEE international conference on electrical, computer and communication technologies (ICECCT)*, IEEE, Tamil Nadu, India, March 2015.
[23] L. N. Petersen, "The ageing body in monty Python live (mostly)," *European Journal of Cultural Studies*, vol. 21, no. 3, pp. 382–394, 2018.
[24] L. Zhuang, J. Dunagan, D. R. Simon, H. J. Wang, and J. D. Tygar, "Characterizing botnets from email spam records," *LEET*, vol. 8, pp. 1–9, 2008.
[25] W. N. Gansterer, A. G. K. Janecek, and R. Neumayer, "Spam filtering based on latent semantic indexing," in *Survey of Text Mining II*, pp. 165–183, Springer, New York, NY, USA, 2008.
[26] D. Lee, M. J. Lee, and B. J. Kim, "Deviation-based spam-filtering method via stochastic approach," *EPL (Europhysics Letters)*, vol. 121, no. 6, Article ID 68004, 2018.
[27] A. K. Jain and B. B. Gupta, "Towards detection of phishing websites on client-side using machine learning based approach," *Telecommunication Systems*, vol. 68, no. 4, pp. 687–700, 2018.
[28] M. F. N. K. Pathan and V. Kamble, "A review various techniques for content based spam filtering," *Engineering and Technology*, vol. 4, 2018.
[29] A. K. Jain and B. B. Gupta, "A novel approach to protect against phishing attacks at client side using auto-updated white-list," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 9, 2016.
[30] A. Bhowmick and S. M. Hazarika, "Machine learning for E-mail spam filtering: review, techniques and trends," 2016, https://www.researchgate.net/publication/303812063_Machine_Learning_for_E-mail_Spam_Filtering_ReviewTechniques_and_Trends.
[31] M. Bassiouni, M. Ali, and E. A. El-Dahshan, "Ham and spam e-mails classification using machine learning techniques," *Journal of Applied Security Research*, vol. 13, no. 3, pp. 315–331, 2018.
[32] J. R. Méndez, T. R. Cotos-Yañez, and D. Ruano-Ordás, "A new semantic-based feature selection method for spam filtering," *Applied Soft Computing*, vol. 76, pp. 89–104, 2019.
[33] R. Alguliyev and S. Nazirova, "Two approaches on implementation of CBR and CRM technologies to the spam filtering problem," *Journal of Information Security*, vol. 3, no. 1, Article ID 16724, 2012.
[34] E. Alpaydin, *Introduction to Machine Learning*, MIT Press, Cambridge, UK, 2020.
[35] E. P. Sanz, J. M. Gómez Hidalgo, and J. C. Cortizo Pérez, "Chapter 3 email spam filtering," *Advances in Computers*, vol. 74, pp. 45–114, 2008.
[36] S. Pitchaimani, V. P. Kodaganallur, and C. Newell, "Systems and methods for controlling email access," Google Patents, 2020.
[37] A. d. A. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran, "Neural-symbolic computing: an effective methodology for principled integration of machine learning and reasoning," *Journal of Applied Logic*, vol. 6, 2019.
[38] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms," in *Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, New Delhi, India, March 2016.
[39] J. Tanha, M. van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 1, pp. 355–370, 2017.

[40] A. Subasi, S. Alzahrani, A. Aljuhani, and M. Aljedani, "Comparison of decision tree algorithms for spam E-mail filtering," in *Proceedings of the 2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, IEEE, Riyadh, Saudi Arabia, April 2018.

[41] W. Hijawi, H. Faris, J. Alqatawna, A. Z. Ala'M, and I. Aljarah, "Improving email spam detection using content based feature engineering approach," in *Proceedings of the Applied Electrical Engineering and Computing Technologies (AEECT)*, IEEE, IEEE, Aqaba, Jordan, 2017.

[42] D. DeBarr and H. Wechsler, "Using social network analysis for spam detection," in *Proceedings of the International Conference on Social Computing, Behavioral Modeling, and Prediction*, Springer, Ethesda, MD, USA, March 2010.

[43] H. Faris, I. Aljarah, and B. Al-Shboul, "A hybrid approach based on particle swarm optimization and random forests for e-mail spam filtering," in *Proceedings of the International Conference on Computational Collective Intelligence*, Springer, Halkidiki, Greece, September 2016.

[44] S. E. Kille, *Mapping Between X. 400 and RFC 822*, University College Department of Computer Science, London, UK, 1986.

[45] S. Jiang, G. Pang, M. Wu, and L. Kuang, "An improved K-nearest-neighbor algorithm for text categorization," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1503–1509, 2012.

[46] H. Takhmiri and A. Haroonabadi, "Identifying valid email spam emails using decision tree," *International Journal of Computer Applications Technology and Research*, vol. 5, 2016.

[47] I. Rish, "An empirical study of the naive Bayes classifier," in *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, University of British Columbia, Computer Science Department, Vancouver, Canada, 2001.

[48] M. Verma and S. Sofat, "Techniques to detect spammers in twitter-a survey," *International Journal of Computer Applications*, vol. 85, no. 10, 2014.

[49] J. Chhikara, R. Dahiya, N. Garg, and M. Rani, "Phishing & anti-phishing techniques: case study," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 5, 2013.

[50] S. Fine, Y. Singer, and N. Tishby, "The hierarchical hidden Markov model: analysis and applications," *Machine Learning*, vol. 32, no. 1, pp. 41–62, 1998.

[51] P. S. Keila and D. B. Skillicorn, "Structure in the Enron email dataset," *Computational & Mathematical Organization Theory*, vol. 11, no. 3, pp. 183–199, 2005.

[52] W. Li, W. Meng, Z. Tan, and Y. Xiang, "Design of multi-view based email classification for IoT systems via semi-supervised learning," *Journal of Network and Computer Applications*, vol. 128, pp. 56–63, 2019.

[53] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," 2003, https://www.csie.ntu.edu.tw/%7Ecjlin/.

[54] Q. Wang, Y. Guan, and X. Wang, "SVM-based spam filter with active and online learning," in *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006*, NIST, Gaithersburg, MD, USA, November 2006.

[55] M. T. Banday and T. R. Jan, "Effectiveness and limitations of statistical spam filters," 2009, https://arxiv.org/ftp/arxiv/papers/0910/0910.2540.pdf.

[56] W. Peng, L. Huang, J. Jia, and E. Ingram, "Enhancing the naive bayes spam filter through intelligent text modification detection," in *Proceedings of the 2018 17th IEEE International Conference on Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference on Big Data Science And Engineering (TrustCom/BigDataSE)*, IEEE, New York, NY, USA, August 2018.

[57] D. Steinberg, "CART: classification and regression trees," in *The Top Ten Algorithms in Data Mining*, pp. 193–216, Chapman and Hall/CRC, Boca Raton, FL, USA, 2009.

[58] Z. Zeng, X. Zheng, G. Chen, and Y. Yu, "Spammer detection on Weibo social network," in *Proceedings of the 2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, IEEE, Singapore, December 2014.

[59] K. Lei, Y. Liu, S. Zhong et al., "Understanding user behavior in Sina Weibo online social network: a community approach," *IEEE Access*, vol. 6, pp. 13302–13316, 2018.

[60] C. Lin, J. He, Y. Zhou, X. Yang, K. Chen, and L. Song, "Analysis and identification of spamming behaviors in sina weibo microblog," in *Proceedings of the 7th Workshop on Social Network Mining and Analysis*, ACM, Chicago, IL, USA, August 2013.

[61] S. O. Olatunji, "Improved email spam detection model based on support vector machines," *Neural Computing & Applications*, vol. 31, no. 3, pp. 691–699, 2019.

[62] K. Tretyakov, "Machine learning techniques in spam filtering," in *Data Mining Problem-Oriented Seminar*, vol. 3, no. 177, pp. 60–79, 2004.

[63] N. F. Rusland, N. Wahid, S. Kasim, and H. Hafit, "Analysis of Naïve Bayes algorithm for email spam filtering across multiple datasets," in *Proceedings of the IOP Conference Series: Materials Science and Engineering*, IOP Publishing, Busan, Republic of Korea, 2017.

[64] A. K. Sharma and S. Sahni, "A comparative study of classification algorithms for spam email data analysis," *International Journal on Computer Science and Engineering*, vol. 3, no. 5, pp. 1890–1895, 2011.

[65] A. Singh and S. Batra, "Ensemble based spam detection in social IoT using probabilistic data structures," *Future Generation Computer Systems*, vol. 81, pp. 359–371, 2018.

[66] N. Sutta, Z. Liu, and X. Zhang, "A study of machine learning algorithms on email spam classification," in *Proceedings of the 35th International Conference, ISC High Performance 2020*, vol. 69, pp. 170–179, Frankfurt, Germany, 2020.

[67] H. Xu, W. Sun, and A. Javaid, "Efficient spam detection across online social networks," in *Proceedings of the 2016 IEEE International Conference on Big Data Analysis (ICBDA)*, IEEE, Hangzhou, China, March 2016.

[68] H. Faris, I. Aljarah, and J. F. Alqatawna, "Optimizing feedforward neural networks using krill herd algorithm for e-mail spam detection," in *Proceedings of the 2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, IEEE, Amman, Jordan, November 2015.

[69] A. H. Wang, "Detecting spam bots in online social networking sites: a machine learning approach," in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, Springer, Berlin, Germany, June 2010.

[70] Z. Guo, Y. Shen, A. K. Bashir et al., "Robust spammer detection using collaborative neural network in Internet of thing applications," *IEEE Internet of Things Journal*, vol. 8, 2020.

[71] A. Makkar and N. Kumar, "An efficient deep learning-based scheme for web spam detection in IoT environment," *Future Generation Computer Systems*, vol. 108, pp. 467–487, 2020.

[72] M. Zavvar, M. Rezaei, M. Rezaei, and S. Garavand, "Email spam detection using combination of particle swarm optimization and artificial neural network and support vector machine," *International Journal of Modern Education and Computer Science*, vol. 8, no. 7, pp. 68–74, 2016.

[73] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for

email spam filtering: review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, Article ID e01802, 2019.

[74] M. Diale, T. Celik, and C. Van Der Walt, "Unsupervised feature learning for spam email filtering," *Computers & Electrical Engineering*, vol. 74, pp. 89–104, 2019.

[75] Z. Ghahramani, "Unsupervised learning," in *Summer School on Machine Learning*Springer, Berlin, Germany, 2003.

[76] R. Ahuja, A. Chug, S. Gupta, P. Ahuja, and S. Kohli, "Classification and clustering algorithms of machine learning with their applications," in *Nature-Inspired Computation in Data Mining and Machine Learning*, pp. 225–248, Springer, Cham, Switzerland, 2020.

[77] W.-F. Hsiao and T.-M. Chang, "An incremental cluster-based approach to spam filtering," *Expert Systems with Applications*, vol. 34, no. 3, pp. 1599–1608, 2008.

[78] A. Sharma and V. Rastogi, "Spam filtering using K mean clustering with local feature selection classifier," *International Journal of Computer Applications*, vol. 108, no. 10, 2014.

[79] E. Tan, L. Guo, S. Chen, X. Zhang, and Y. Zhao, "Unik: unsupervised social network spam detection," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, ACM, San Francisco, CA, USA, October 2013.

[80] A. Ahmed, "Improved spam detection using DBSCAN and advanced digest algorithm," *International Journal of Computer Applications*, vol. 69, 2013.

[81] Y. Cabrera-León, P. G. Báez, and C. P. Suárez-Araujo, "Non-email spam and machine learning-based anti-spam filters: trends and some remarks," in *Proceedings of the International Conference on Computer Aided Systems Theory*, Springer, Las Palmas de Gran Canaria, Spain, February 2017.

[82] M. Sasaki and H. Shinnou, "Spam detection using text clustering," in *Proceedings of the 2005 International Conference on Cyberworlds (CW'05)*, IEEE, Singapore, November 2005.

[83] K. Narisawa, H. Bannai, K. Hatano, and M. Takeda, "Unsupervised spam detection based on string alienness measures," in *Proceedings of the International Conference on Discovery Science*, Springer, Sendai, Japan, October 2007.

[84] P. Lison, *An Introduction to Machine Learning*, Language Technology Group, Edinburgh, UK, 2015.

[85] S. Smadi, N. Aslam, and L. Zhang, "Detection of online phishing email using dynamic evolving neural network based on reinforcement learning," *Decision Support Systems*, vol. 107, pp. 88–102, 2018.

[86] Y.-F. Chiu, C.-M. Chen, B. Jeng, and H.-C. Lin, "An alliance-based anti-spam approach," in *Proceedings of the Third International Conference on Natural Computation (ICNC 2007)*, IEEE, Haikou, China, August 2007.

[87] G.-H. Lai, C.-M. Chen, and C.-S. Laih, T. Chen, A collaborative anti-spam system," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6645–6653, 2009.

[88] Y. Dou, G. Ma, P. S. Yu, and S. Xie, "Robust spammer detection by nash reinforcement learning," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, Virtual Event CA, USA, July 2020.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 3306098, 12 pages
https://doi.org/10.1155/2022/3306098

Research Article

A Method for Identifying Tor Users Visiting Websites Based on Frequency Domain Fingerprinting of Network Traffic

Yuchen Sun,[1,2] **Xiangyang Luo**,[1,2] **Han Wang**,[1,2] **and Zhaorui Ma**[1,3]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, Henan, China*
[2]*Key Laboratory of Cyberspace Situation Awareness of Henan Province, Zhengzhou 450001, Henan, China*
[3]*Zhengzhou University of Light Industry, Zhengzhou 450001, Henan, China*

Correspondence should be addressed to Xiangyang Luo; xiangyangluo@126.com

Received 16 October 2021; Revised 19 December 2021; Accepted 12 January 2022; Published 31 January 2022

Academic Editor: Weizhi Meng

Although the anonymous communication network Tor can protect the security of users' data and privacy during their visits to the Internet, it also facilitates illegal users to access illegal websites. Website fingerprinting attacks can identify the websites that users are visiting to discern whether they are performing illegal operations. Existing methods tend to manually extract the traffic features of users visiting websites and construct machine learning or deep learning models to classify the features. While these methods can be effective in classifying unknown website traffic, the effect of classification in the use of defensive measures or onion service scenarios is not yet ideal. This paper proposes a method to identify Tor users visiting websites based on frequency domain fingerprinting of network traffic (FDF). We extract the direction and length features of circuit sequences in access traffic and combine and transform them into the frequency domain. The classification of access traffic is accomplished by using a deep learning classification model combining CNN, FC, and Self-Attention. In this paper, the proposed FDF method is experimentally validated in common scenarios of Tor networks. The results show that FDF outperforms the existing methods for classification in different Tor scenarios. It can achieve 98.8% and 94.3% classification accuracy in undefended and WTF-PAD defense scenarios, respectively. In the onion service scenario, the accuracy is improved by 4.7% over the current state-of-the-art Tik-Tok method.

1. Introduction

As people's awareness of protecting personal privacy continues to increase, more and more users are beginning to use anonymous communication systems to interact with the outside world. However, many criminals have also used anonymous communication systems to conduct illegal operations. Some criminals have established illegal trading websites. Users can purchase leaked database information and even network attack services through these websites. Driven by interests, a large number of network intrusions have spawned on the Internet. Tor is currently the most popular anonymous communication system, and it provides privacy to over 200 million users every day [1]. Tor protects the anonymity of user access by creating an encrypted link with three-hop relays. These relays are randomly selected, prevented from being traced through the bridge and pluggable transmission [2], and the links are changed periodically as the client accesses the server. Although it is difficult to directly crack the Tor anonymous communication system, previous studies have shown that network traffic analysis can affect the security of Tor [3–19], especially website fingerprint (WF) attacks. When users visit each website, they will generate different network traffic features, such as different numbers of data packets and different traffic burst patterns. In a WF attack, the enforcer intercepts network traffic and extracts the features of the traffic packets in an encrypted connection between the monitored user and the entry node of Tor. The classifier determines whether the intercepted traffic is associated with the website of interest to the enforcer, and if the traffic matches the classifier, it indicates that the monitored user is visiting the website of interest to the enforcer. The WF attack allows the enforcer to determine whether the monitored user is browsing illegal websites, especially websites that conduct black transactions, which is of great significance for combating illegal crimes.

The original intention of Tor is to provide users with anonymity during data communication. Tor should try to avoid the occurrence of WF attacks so as not to affect its security. Therefore, defense measures against WF attacks are proposed. But for enforcers, due to a large number of illegal activities in Tor, it is necessary to monitor criminals and illegal websites. Therefore, it is necessary to conduct further research on WF attacks on Tor that uses defensive measures. The proposed defense measures basically reduce the bursts in the original Tor traffic and confuse the traffic, which significantly reduces the efficiency of WF attacks. For measures that may be used by Tor in the future, it is important to improve their recognition accuracy. In addition, the onion service [20] is the most secure service provided by Tor, which contains a large number of illegal transactions. WF attacks on Tor networks using onion service are also of interest for research. To access the website in the onion service, users need to establish more complex links and have a more complete security verification mechanism. This makes the access traffic mixed with a lot of traffic noise generated for the purpose of authentication. Existing methods for fingerprinting Tor traffic using the onion service are not yet very effective. These methods can detect the behavioral patterns of users visiting different websites from different features such as timing and direction of traffic. However, none of them can reduce the influence of traffic noise on fingerprint recognition.

To address the shortcomings in existing studies, this paper adopts a frequency domain transformation method to deal with Tor traffic. Unlike the existing studies, the frequency domain processing method can effectively reduce the impact of noise on fingerprint recognition when users access the server. In particular, for scenarios where defensive measures and onion services are used, the impact of noise on fingerprint recognition is greater due to the increased security mechanisms. We achieve more significant results in these environments than the existing methods.

The contributions of our work are as follows:

(1) We propose FDF, a fingerprint recognition method for websites based on DWT frequency domain processing. We compared several frequency domain processing methods and found that the wavelet transform works best through theoretical as well as experimental analysis. Due to the properties of the frequency domain transform, we combine for the first time the signal element sequence direction as well as length features for the input of a deep learning model.

(2) We have improved the DF [15] model. The Self-Attention module is added to the original model to support intelligent and efficient analysis of website traffic. In the closed-world scenario (we assume that the monitored user only visits the websites we are interested in. The performance of the classifier can be observed more clearly through the closed world), the classification accuracy on Undefended, WTF-PAD [21], and Onion Service [20] datasets are better than the existing models. Especially for the Onion Service [20] dataset, the accuracy of FDF has reached 70.7%, while the accuracy of the current state-of-the-art Tik-Tok [18] method is only 66.0%.

(3) We evaluated FDF in a more realistic open-world scenario (we assume that monitored users can randomly visit different websites. These sites can be sites that we are interested in or sites that we are not interested in. Through the open world, a more realistic environment can be simulated), where we collected a dataset containing 40,000 unmonitored websites and achieved more desirable Precision and Recall in both undefended and WTF-PAD [21] environments, indicating that FDF is effective in real environments.

The remaining sections of this paper are organized as follows. The second part is the background and related work, which describes the existing approaches to the problem of website fingerprinting for the Tor system. The third part is the problem description, describing the process of fingerprint identification of the website. The fourth part introduces the FDF attack process in detail and explains the key steps of the model in principle. The fifth part introduces the experimental dataset, comparative experiments, and experimental results. The sixth part discusses the problems in the experiment of the method proposed. Finally, the seventh part is the conclusion of this paper.

2. Background and Related Work

For website fingerprint attacks, the data processing method of network traffic and the choice of classifier have a significant impact on its attack efficiency. The website fingerprinting methods that have performed well in recent years are shown in Table 1. In the earliest research, researchers used machine learning to classify website traffic [5–13]. The WF attack was first evaluated by Herrmann et al. [5] in 2009. In 2011, Panchenko et al. [6] used the Herrmann et al. dataset and employed an SVM classifier to classify Tor network traffic by various features such as packet traffic and time. The K-NN attack was proposed by Wang et al. [7] in 2014. The method employs a K-Nearest Neighbor (K-NN) classifier that uses a combination of features to evaluate the similarity between different websites through a distance metric. However, this method is not effective in reducing the impact of noise on WF attacks. In 2016, Panchenko et al. [8] proposed a novel WF attack CUMUL against Tor based on the cumulative representation of traces. The method considers the effect of real noise on WF attacks. However, the problem of overfitting occurs in the process of actual classification. In 2016, Hayes et al. [9] proposed a website fingerprinting attack K-FP based on random decision forest. This method uses a random forest to extract fingerprints for each traffic instance, uses Hamming distance to calculate the distance between these fingerprints, and finally classifies them by k-nearest fingerprint technology. This method shortens the classification time and reduces the impact of overfitting on the classification results. However, the noise will have a certain impact on K-FP.

In recent years, with the massive application of deep learning on WF attacks [14–18], the performance of WF attacks has been further improved. Sirinam et al. [15]

Table 1: Well-performing website fingerprinting methods.

Method	Accuracy (%)	Advantage	Disadvantage
K-NN [7]	95.0	Multiple features are used.	The noise is not considered.
CUMUL [8]	97.3	The noise is considered.	Overfitting.
K-FP [9]	95.5	Sorting time is shortened, and overfitting is reduced.	Classification effect is easily affected by the noise.
DF [15]	98.3	Complex CNN model is proposed.	Large datasets are needed.
TF [16]	95.0	Small datasets are needed.	The accuracy needs to be improved.
Tik-Tok [18]	98.4	Multiple time features are considered.	Classification effect is easily affected by circuit congestion.

proposed the Deep Fingerprinting (DF) attack in 2018. The highlight of DF is the design of complex convolutional neural network (CNN) structures. The deep learning model of DF can solve the WF attack problem well, and the framework of the DF model has been basically borrowed in the subsequent studies. However, DF has a long period to complete the WF attack and requires a large number of training sets to achieve better classification results. The data staleness problem also has an impact on the attack during the data collection. Sirinam et al. [16] studied a triplet fingerprint attack TF in 2019. This attack uses a triplet network for N-shot learning [22]. This method can effectively reduce the workload of data collection and training in the implementation of WF attacks, but the accuracy of the attack needs to be improved. Rahman et al. [18] proposed the Tik-Tok attack based on packet timing information in 2020. The method uses a set of new features based on burst level features related to timing. The information contained in each of these features is mutually exclusive and improves the robustness of the classifier. The method fully considers the timing features so that it can obtain effective information and achieve a high accuracy rate of website fingerprinting under the scenario of using defense. However, due to the instability of the Tor link, it is easy to cause circuit congestion [23, 24]. Circuit congestion can have an impact on the timing information, thus reducing classification accuracy.

In order to make the Tor network more secure, researchers have proposed some defensive measures [21, 25–28] to defend against WF attacks. The basic principle is to operate on packet traffic (add, delete, delay packets, etc.). In order to achieve the purpose of confusing flow features, the BuFLO [25] defense method was proposed by Dyer et al. This method achieves the effect of transmitting data packets close to a constant rate by sending data packets of a fixed length in the Tor network at a fixed time. Juarez et al. proposed WTF-PAD [21]. WTF-PAD is a probabilistic connection filling defense based on adaptive filling. It masks the features of traffic bursts by adding short-delay pseudotraffic bursts, thereby reducing the threat of WF attacks. Wang et al. proposed the Walkie-Talkie (W-T) [28] method. W-T modified the browser to communicate in half-duplex mode instead of the usual full-duplex mode. The half-duplex mode converts the cell sequence into a burst sequence, which not only saves additional overhead but also reduces the characteristics of the cell sequence, thereby leaking less information to the enforcer.

3. The Description of Problem

Tor consists of thousands of relays that form a worldwide network of volunteer overlays to direct Internet traffic. During a user's visit to a website, the traffic is encrypted in multiple layers so that an attacker cannot know which websites the user is visiting. Many illegal websites have emerged in the onion service, where users can log in to complete transactions without being tracked. Therefore, obtaining the websites that users are visiting with the knowledge of their identity is a problem worth investigating.

Although Tor can effectively protect the security and privacy of users, it is still possible to reduce the anonymity of users by means of traffic analysis. A series of associated traffic is generated when users visit a website, and the pattern of this traffic is relatively fixed within a certain period of time. That is, users visiting the same website in the same region within a certain time frame can obtain similar packets. Therefore, the user's access traffic can be analyzed to discern which website the user is visiting. As shown in Figure 1, an enforcer is deployed locally to collect the network traffic between the client and the server and identify the website that the user is visiting. This enforcer can be a router, an Internet Service Provider (ISP), an autonomous service, and so on, capable of arbitrarily collecting encrypted traffic between the client and the entry node. The enforcer cannot discard, modify, insert, and delay packets. If the traffic is tampered with during a user's visit to the site, it may result in errors or anomalies on the user's return page. This not only affects users' browsing but also alerts them to the possibility of their privacy being compromised. Especially for illegal users, it will make it more difficult to collect their incriminating evidence.

For sites of interest to the enforcer, we call them monitored sites. For other types of websites, we call them unmonitored websites. In WF attacks, the enforcer's task is to identify the monitored websites. The enforcer needs to set up a classifier, and in addition to that, he has to loop through the Tor network to the monitored website and collect the traffic during the visit. After the collection is complete, the enforcer has to manually extract the traffic features and construct a traffic matrix from all the processed traffic data for the training of the classifier. When the classifier is trained, the enforcer can passively collect the encrypted traffic during the monitored user's access to the server, process the traffic in the same way as the training set, and then use the classifier to classify the traffic to determine

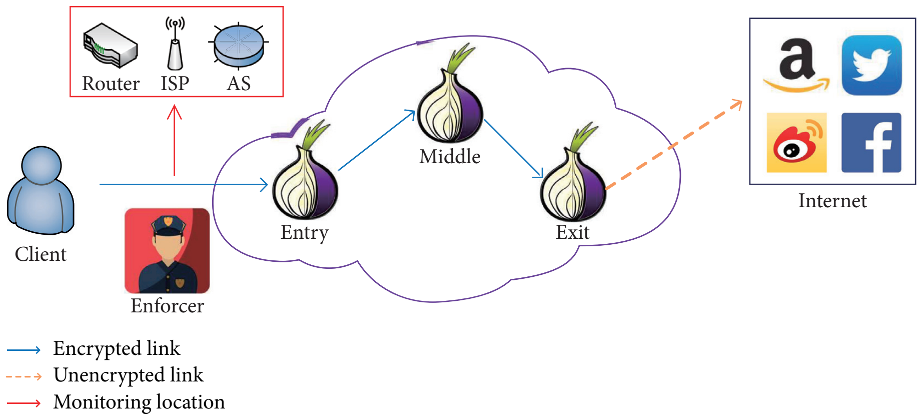


Figure 1: Schematic diagram of website fingerprint recognition process.

whether the website being visited by the monitored user is a monitored website.

In order to improve the readability of the paper, we summarize and explain the notations in our method, as shown in Table 2.

4. The Proposed Method

4.1. The Principle Framework and Main Steps of the Method. In the existing website fingerprint identification methods, the main factor affecting the classification of Tor traffic fingerprints is the noise in the traffic. These noises can effectively confuse the features of the original Tor traffic and reduce its classification accuracy. In response to this problem, we found that frequency domain transformation can reduce the impact of noise on classification and proposed a DWT-based website fingerprint recognition method. The complete fingerprint identification process is shown in Figure 2.

This method is divided into two stages: data preprocessing and classifier classification. Data preprocessing is mainly to extract features from the collected data packets and transform the extracted sequences into the frequency domain to form circuit frequency domain feature sequences. The method of frequency domain transformation can increase the difference of traffic patterns of different websites and can obtain better classification results. Classifier classification focuses on identifying and classifying the preprocessed data using deep learning techniques. Before classifying the unknown traffic, the classifier is trained. This process requires collecting a large number of circuit frequency domain feature sequences and corresponding the sequences to their site labels one by one to generate a training sequence matrix and a training label matrix. After the training is completed, the traffic to be tested is converted into a test sequence matrix for classifier classification. This method uses a deep learning classification model combining CNN, FC, and Self-Attention and uses various regularization techniques in the model to prevent overfitting in the website recognition process.

Step 1. Capture the traffic packets of users visiting the website. Capture the background traffic during the user's visit to the website and generate the raw traffic packets.

Step 2. Extract the feature sequence of the circuit. Extract the direction and length information of the sequence of circuits in the raw network traffic packets, and combine them to form the feature sequence of circuits.

Step 3. Generate the frequency domain feature sequence of the circuit. The feature sequence of the circuit is transformed into the frequency domain feature sequence of the circuit by DWT transformation, and the low-frequency sequence generated after DWT transformation is retained.

Step 4. Store the data into the database. Store the frequency domain feature sequences of the circuits and their corresponding site labels into the database.

Step 5. Generate training set. The frequency domain feature sequences of the circuits and the site labels are extracted from the database according to the model training requirements, and the training sequence matrix and the training label matrix are generated.

Step 6. Construction of the model framework. A suitable neural network framework is selected according to the data type and features of the traffic, and a series of overfitting prevention methods are used to improve the accuracy of the model classification.

Step 7. Model training. The deep learning model is trained using the above matrix. The appropriate hyperparameters are selected through training.

Step 8. Generate the test set. Extract the frequency domain feature sequence of the circuit to be tested from the database and generate the test sequence matrix.

Table 2: List of notations.

Notations	Description
$\mathrm{Seq}_{\mathrm{dir}}$	The direction sequence of the circuit.
$\mathrm{Seq}_{\mathrm{len}}$	The length sequence of the circuit.
$\mathrm{Seq}_{\mathrm{mix}}$	The feature sequence of the circuit.
$x(n)$	The feature sequence of the original circuit.
α	The number of layers of DWT decomposition.
$x_{\alpha,L}(n)$	The low-frequency sequence generated after DWT transformation.
$x_{\alpha,H}(n)$	The high-frequency sequence generated after DWT transformation.
$L(n)$	The low-pass filter.
$H(n)$	The high-pass filter.
Q	The downsampling multiples.
N	The length of the circuit feature sequence.

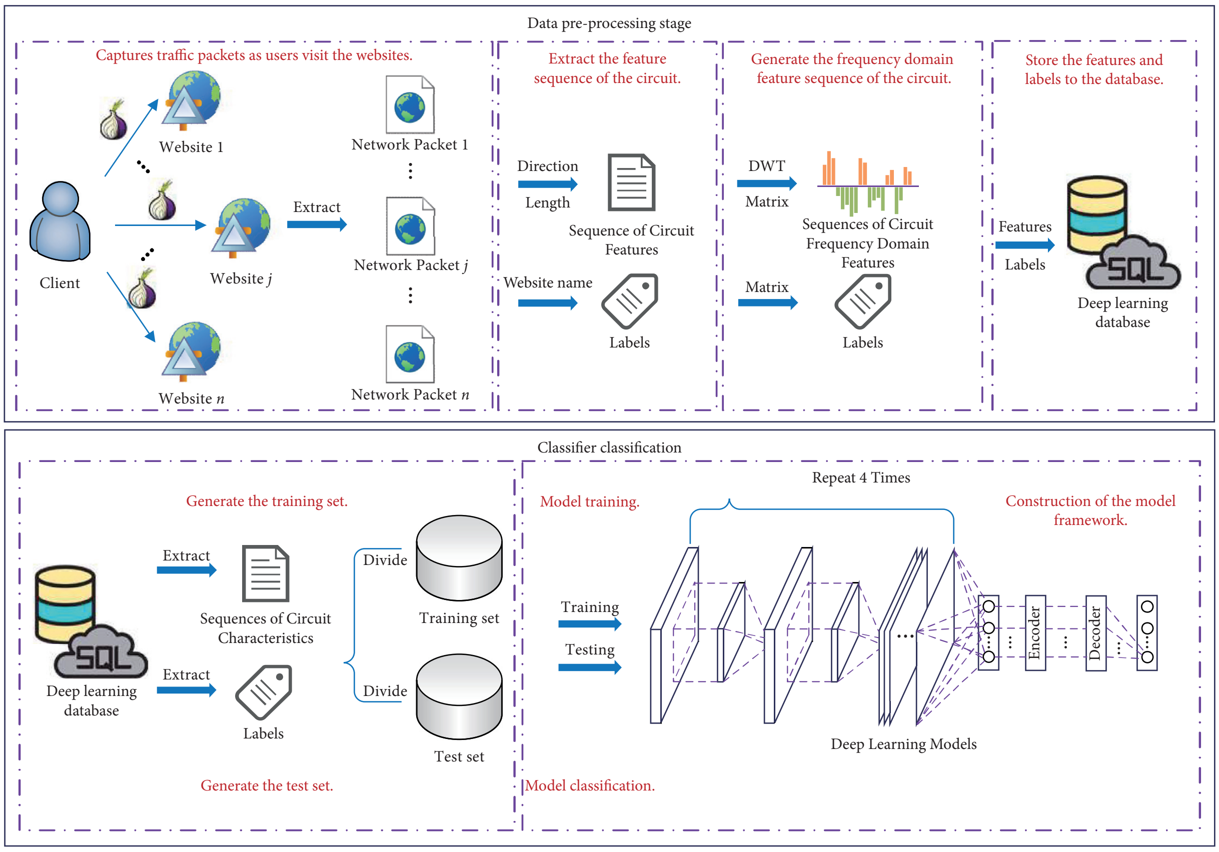

Data pre-processing stage
Captures traffic packets as users visit the websites.
Website 1
Website j
Website n
Client
Extract
Network Packet 1
Network Packet j
Network Packet n
Extract the feature sequence of the circuit.
Direction
Length
Sequence of Circuit Features
Website name
Labels
Generate the frequency domain feature sequence of the circuit.
DWT
Matrix
Sequences of Circuit Frequency Domain Features
Matrix
Labels
Store the features and labels to the database.
Features
Labels
Deep learning database
Classifier classification
Generate the training set.
Extract
Sequences of Circuit Characteristics
Extract
Labels
Deep learning database
Divide
Training set
Divide
Test set
Generate the test set.
Model training.
Repeat 4 Times
Construction of the model framework.
Training
Testing
Encoder
Decoder
Deep Learning Models
Model classification.

Figure 2: Identification process of Tor users visiting websites based on network traffic frequency domain fingerprinting.

Step 9. Model classification. Use the model to predict the test sequence matrix and obtain the website labels corresponding to the frequency domain feature sequences of the circuit to be tested. Complete the identification of the unknown traffic and correspond the traffic to the website.

4.2. Data Preprocessing

4.2.1. Extract the Feature Sequence of the Circuit. By capturing the packets during a user's visit to a website, we can obtain the circuit sequence of the packets of this website. By analyzing this sequence, we can extract various features, including the direction, length, timing, and burst of the circuit sequence. We select the direction and length of the sequence as the key features to be extracted.

Direction. The sequence of the original circuit is mapped into the value domain of [+1, −1], and the enforcer is usually monitored before the entry relay. Specify the direction of data inflow into the enforcer as "+1" and the direction of data outflow from the enforcer as "–1." By this method, the direction sequence of the circuit $\mathrm{Seq}_{\mathrm{dir}}$ is formed.

Length. Each packet in the sequence of the circuit is packaged by the protocol before transmission. Clients and servers interact with each other through TCP protocol, so packets that do not contain TCP protocol are to be filtered out. Next, the length of the TCP protocol layer circuit is extracted to compose the length sequence of the circuit $\mathrm{Seq}_{\mathrm{len}}$.

The length sequence and direction sequence of the circuit are combined, and a new feature sequence $\mathrm{Seq}_{\mathrm{mix}}$ is constructed by multiplying the terms of these two sequences.

$$\mathrm{Seq}_{\mathrm{mix}} = \mathrm{Seq}_{\mathrm{dir}} \times \mathrm{Seq}_{\mathrm{len}}. \tag{1}$$

In a previous study [15], researchers have verified through experimental arguments that the length of the circuit sequence does not significantly improve the accuracy of the attack. A good attack can be achieved by using only the direction of the circuit sequence. However, in our approach, the length of the circuit sequence is necessary. Any time sequence can be seen as an infinite superposition of sine waves of different frequencies and formed. Amplitude is the most basic characteristic of a sine wave. If only the direction of the circuit sequence is used, the full information of the sine wave cannot be reflected. Therefore, we consider combining the length and direction of the circuit sequence to be able to achieve better results in the frequency domain transformation.

4.2.2. Generate the Frequency Domain Feature Sequence of the Circuit.

The circuit sequence of a packet based on timing can be understood as the result of the variation of the signal over time. The analysis of the circuit sequence in the frequency domain allows us to obtain more useful information. It is possible to analyze the composition of the sequence frequency, more precisely to decompose the sequence into several subsequences. In this way, the internal connection of each packet in the circuit sequence is reflected, and it is convenient to achieve better results in the subsequent neural network training process.

DWT (Discrete Wavelet Transformation) can discretize the scales and translations of the fundamental wavelets. It can analyze the frequency domain features of local time-domain processes and is more suitable for the analysis of nonsmooth processes. The discrete wavelet transform uses a bandpass filter to decompose the circuit sequence into multiple frequency domain components, which greatly reduces the interference of noise and makes the presentation more intuitive. The architecture of the discrete wavelet transform decomposition process for discrete sequences is shown in Figure 3.

$L(n)$ and $H(n)$ represent the low-pass filter and high-pass filter, and their correspondence is shown in relation (2). $\downarrow Q$ denotes the Q-fold downsampling filter. The sequences decomposed at layer α in the architecture can be represented according to the relations (3) and (4). The high-frequency components are extracted in each layer, while the low-frequency components are deployed to the next layer to continue the decomposition. Since Q-fold downsampling is performed at each layer, if the length of the input circuit sequence is N, then the length of both $x_{\alpha,L}(n)$ and $x_{\alpha,H}(n)$ in the α th layer is N/Q^{α}.

$$L(N-1-n) = (-1)^n H(n), \tag{2}$$

$$x_{\alpha,L}(n) = \sum_{k=0}^{K-1} x_{\alpha-1,L}(Q\cdot n-k)L(k), \tag{3}$$

$$x_{\alpha,L}(n) = \sum_{k=0}^{K-1} x_{\alpha-1,L}(Q\cdot n-k)H(k). \tag{4}$$

In our model, we perform a one-layer architectural decomposition of the circuit sequence and set the multiplier of the downsampling filter to 2. Thus, we are able to obtain the sequence decomposition method as shown in relations (5) and (6).

$$x_{1,L}(n) = \sum_{k=0}^{K-1} x(2n-k)L(k). \tag{5}$$

$$x_{1,L}(n) = \sum_{k=0}^{K-1} x(2n-k)H(k). \tag{6}$$

The circuit sequence is processed in the frequency domain using the relation (5) after the feature processing. The frequency domain processing results in a low-frequency sequence $x_{1,L}(n)$ and a high-frequency sequence $x_{1,H}(n)$. $x_{1,L}(n)$ contains the slowly changing part of the circuit sequence. It is the basic frame of the sequence and belongs to the approximate information of the sequence. $x_{1,H}(n)$ contains the rapidly changing part of the circuit sequence. It belongs to the detailed information of the sequence, which contains the noise. We use the low-frequency part $x_{1,L}(n)$, which can represent the contour features of the sequence, for the training of the model. It can reduce the interference of noise in the sequence for fingerprint recognition.

In our method, both $L(n)$ and $H(n)$ are of constant length, independent of the length N of the circuit sequence. We only need the low-frequency part of the wavelet transform. The convolution of the circuit sequence and the filter requires $O(N)$ time complexity. After each layer of convolution, a branch of length $N/2$ is formed. Therefore, the time complexity required for the entire frequency domain transformation process is $O(N)$.

4.3. Classifier Classification

4.3.1. Construction of the Model Framework.

Website fingerprinting on Tor is a supervised classification problem. Starting from DF [15], deep learning techniques have achieved good results on the website fingerprinting problem. We have borrowed from these models and made improvements. In DF, two convolutional layers were used before each Max Pooling. The researchers believe that adding more convolutional layers to each Base Model can obtain a deeper network and extract features more efficiently. In our model, each Max Pooling layer is preceded by only one convolutional layer, which can effectively reduce the complexity of the neural network. After the Base Model, we add a Self-Attention layer. The reason for this is that

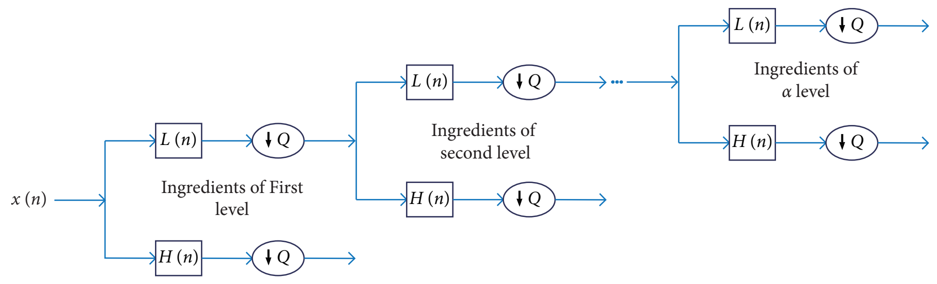


Figure 3: DWT decomposition process.

CNN only considers the information in the receptive field and only acts on a local scale. Self-Attention, on the other hand, considers the information on the entire circuit sequence. It contains a much wider range. Therefore, we consider extracting the local features in the circuit sequence by CNN first and then extracting the global features by Self-Attention, so as to form a complete model. This approach not only reduces the complexity of the neural network but also does not affect the extraction of features. The neural network classification model is shown in Figure 4.

Since neural networks have a fixed input size requirement, for one-dimensional circuit sequences, different lengths of circuit sequences need to be fixed to the same length. After data preprocessing, the circuit sequence length needs to be set to a fixed threshold. Sequences with length less than the threshold are filled with 0, and those with length greater than the threshold are truncated. All circuit sequences are combined to form the input matrix.

To address the selection of hyperparameters in different modules, we empirically assign a range of values to these hyperparameters. For hyperparameters with a small range of values, the hyperparameters are taken iteratively. For hyperparameters with a large range of values, the hyperparameters are taken using the dichotomous method. In the process of model construction, we filtered the hyperparameters module by module and finally obtained the best combination of hyperparameters.

In Tor, many useless data packets are generated when users visit websites due to network congestion, identity verification, and other reasons. This may cause the same user to make multiple visits to the same website in the near time to generate quite different traffic. These noisy data packets can cause overfitting problems during neural network training. For the overfitting problem, we use regularization techniques such as Dropout, Batch Normalization (BN), and Label Smoothing methods. Dropout reduces the interaction between hidden nodes and makes the model more generalizable by making a certain neuron probabilistically stop working. BN normalizes the output results so that the output obeys the standard normal distribution and reduces the internal covariance shift (ICS), which not only helps the network fit faster but also reduces the overfitting problem. Due to the small number of parameters in the convolutional layer, Dropout is rarely used after the convolutional layer, and BN is usually used. In our model, BN is connected immediately after each CNN, and Dropout is used after Max Pooling to prevent overfitting. There are many parameters in the FC and Prediction process, so BN and Dropout can be used together.

In order to approximate the predicted probability distribution to the true distribution during neural network prediction, a common practice is to encode the true labels using the one-hot method. This encoding approach can make the model lack adaptability and be overconfident in its predictions, which can lead to overfitting problems. Label Smoothing smoothes the empirical distribution of the gap between the maximum prediction and the mean of the other categories by adding a smoothing factor. The essence of Label Smoothing is to drive the classification probability results after the activation of the Softmax activation function in the neural network closer to the correct classification, so it is placed in the last part of the model.

4.3.2. Model Training. The selection process of the hyperparameters in the model is shown in Table 3, containing the range of values for each hyperparameter and the value that achieves the best results. We conducted experiments on tuning parameters using the collected Undefended Closed-World dataset and validated them using other datasets, all with good results.

5. Experiments and Results Analysis

To validate the performance of the proposed FDF method, we conducted a series of experiments based on the Undefended, WTF-PAD [21], and Onion Sites [20] datasets.

5.1. Dataset

5.1.1. The Closed-World Dataset. We performed a recursive crawl of the homepages of the top 100 websites ranked by Alexa [31] through the Tor network, with a total of 1000 crawls per site. We deployed the work on LXD containers on ten VPS servers in different countries.

5.1.2. The Open-World Dataset. Since it is not realistic to visit all Internet sites, we selected some of them for simulating the open-world experiment. We visited the top 40,000 websites in Alexa ranking in order. Because these sites are unmonitored sites, they cannot contain the 100 monitored sites collected in the closed-world experiment. We deployed the work in the same ten VPS servers.

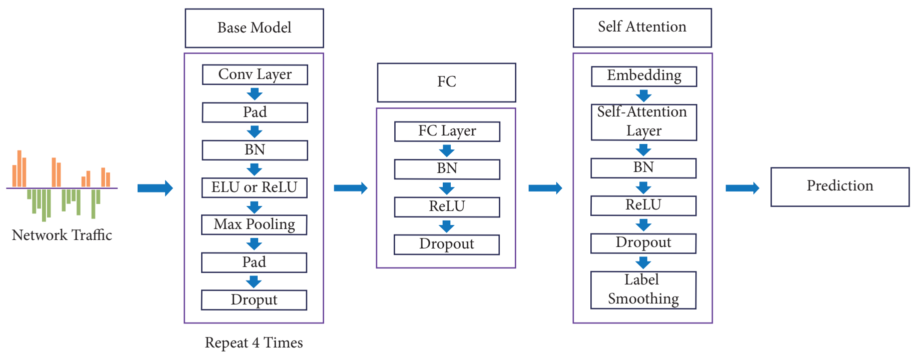


Figure 4: The neural network classification model of WF attack.

Table 3: Selection of hyperparameters for FDF.

Parameters	Search space	Selected value
Input dimension	[500 . . . 7000]	5000
Wavelet	Haar, Db, Sym, Coif, Bior, and Rbio	Coif
Base Model	GoogleNet [29], ResNet [30], and DF [15]	DF
Number of FC layers	[1 . . . 4]	1
Hidden units (FC)	[256 . . . 2048]	512
Hidden dim (Self-Attention)	[128 . . . 2048]	256
Optimizer	SGD, adam, adamax, and RMProp	Adamax
Batch size	[32 . . . 256]	128
Dropout (Pooling, Self-Attention, and FC)	[0.1 . . . 0.8]	[0.1, 0.1, 0.5]

5.1.3. The Onion Service Dataset. A collection of onion domains was conducted by Overdorf et al. [19], and the sites were fingerprinted. They published the dataset used for their experiments to the Internet in the form of tshark logs, which we chose to use for our experiments. Since collecting a large number of onion domains is a difficult task, we chose to use this dataset for experiments.

Table 4: Number of classes and instances in each dataset.

Dataset	Classes	Instances/class	Total
Undefended (CW)	100	1000	100000
Undefended (OW)	40000	10	400000
WTF-PAD (CW)	100	1000	100000
WTF-PAD (OW)	40000	10	400000
Onion Sites (CW)	539	77	41503

5.1.4. The Defense Dataset. We performed evaluation tests on the WTF-PAD [15] defense approach. For the WTF-PAD [21] defense, we adapted the raw traffic we collected using a script code posted by the researchers in GitHub. It is used to simulate the traffic generated during access in a real environment according to the defense protocol populated.

A total of five datasets were used in our experiments. In the closed-world scenario, data were collected for undefended, WTF-PAD [21] defense, and onion services, generating the Undefended (CW), WTF-PAD (CW), and Onion Sites (CW) datasets. In the open-world scenario, data collection was performed for both undefended and WTF-PAD [21] defense methods to generate Undefended (OW) and WTF-PAD (OW) datasets. Table 4 shows the website classes and the number of visited website instances in each dataset. We randomly divided each dataset into three parts: training set, validation set, and test set. Due to the large size of the dataset, we divided it according to the ratio of 8 : 1 : 1.

5.2. Website Fingerprinting Experiments on the Closed-World Dataset. The core of the FDF method is to process the circuit sequence in the frequency domain before performing the deep learning fingerprint recognition on the circuit sequence. In addition to DWT, Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT) are also mainstream frequency domain processing methods, which have good applications in the direction of image processing. We compared the above three frequency domain processing methods in the closed-world environment. FFT is an efficient and fast algorithm of DFT that can reduce the operation time. Therefore, we use FFT instead of DFT for our experiments.

Table 5 shows the accuracy results of fingerprint recognition on different datasets after processing by three frequency domain processing methods. It can be found that the accuracy rates of the two methods, DCT and FFT, are relatively close to each other. Meanwhile, DWT is significantly better than the other two methods on all three different datasets. This is because circuit sequences are nonstationary signals, and DWT has better results for nonstationary signals, while FFT is more suitable for handling stationary signals.

To show the good attack effect of FDF, we compared it with K-NN [7], K-FP [9], CUMUL [8], DF [15], and Tik-Tok [18] attacks.

Table 5: Comparison of the attack accuracy of three frequency domain processing methods in the closed world.

Method	The accuracy of different dataset		
	Undefended (CW) (%)	WTF-PAD (CW) (%)	Onion Sites (CW) (%)
DCT	98.2	92.6	64.3
FFT	98.3	92.8	65.1
DWT	98.8	94.3	70.7

Table 6 shows the accuracy results of different attack methods in the closed-world scenario for the three environments. It can be found that FDF outperforms the other attacks on the Undefended, WTF-PAD [21], and Onion datasets. Each attack method does not perform very well on the Onion dataset, which we believe is related to the dataset. The Onion dataset has 539 categories with only 77 traffic data per category, which is much less data than other datasets and therefore reduces the accuracy rate.

5.3. Website Fingerprinting Experiments on the Open-World Dataset. To simulate a realistic environment, we conducted experiments in the more realistic open-world scenario. In the open world, the adversary first determines whether the traffic data belongs to monitored or unmonitored sites and second classifies all the traffic belonging to monitored sites according to the limited set of monitored sites.

For open-world scenarios, Precision and Recall were suggested in literature [9, 21] for the evaluation of classifiers. Because the difference between the limited set of monitored sites and the limited set of unmonitored sites may be too large, True Positive Rate (TPR) and False Positive Rate (FPR) can be wrong in the interpretation of the model attack performance.

For the performance evaluation process, we used the standard model proposed in DF Attack [15]. In the open-world scenario, for the dataset of monitored websites, they are trained in the same way as in the closed world. For the dataset of unmonitored websites, it is trained as an additional class. We evaluated undefended and WTF-PAD [21] in the open-world scenario by tuning the attack for Precision and Recall. Precision and Recall are shown in relations (7) and (8).

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

$$\text{Recall} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Positive (FP)}}. \quad (8)$$

Tables 7 and 8 show the results of different methods to tune the attacks for Precision and Recall, respectively, for the two datasets in the open-world scenario. Figure 5 shows the Precision-Recall curves of the attacks in the open world. The above graphs show that DWT shows good results for the Undefended dataset. When attacking tuned for Precision, it achieves 0.99 Precision and 0.94 Recall, while when attacking tuned for Recall, it achieves 0.93 Precision and 0.99 Recall. For the WTF-PAD [21] dataset, the Precision and Recall of all methods decreased due to the added defenses. The best performance was achieved by DWT with a Precision of 0.98 and a Recall of 0.76 when attacking tuned for Precision and a Precision of 0.75 and a Recall of 0.96 when attacking tuned for Recall.

6. Discussion

In this section, we discuss the data preprocessing method before the feature frequency domain processing and the number of DWT decomposition layers for the feature frequency domain processing.

Input for feature frequency domain processing: The most important feature of packets in WF attacks is the direction of the circuit sequence. In DF [15], researchers have also compared packet processing methods and found that the best results can be achieved using only the direction of the circuit sequence. But for frequency domain transformation, not only the direction of the cell sequence but the amplitude of the cell sequence is also very important. If only the direction of the circuit sequence is used and its amplitude is ignored, a lot of information of the original sequence will be leaked during the frequency domain transformation. Thus, the effect of frequency domain processing of the signal cannot be achieved, and the result is degraded. Therefore, we choose to use the direction and length of the signal element sequence as the input for the feature frequency domain processing.

The number of decomposition layers of DWT: in other applications of DWT, multiple layers of wavelet decomposition are often required to achieve better results. Each DWT decomposition results in two components, a high-frequency component, and a low-frequency component. These two components are of the same length. In our method, we use the decomposed low-frequency components every time. In other words, for each layer of DWT decomposition, the length of the circuit sequence is halved. For example, if the input length of the circuit sequence in the FDF model is 5000, a two-layer DWT would require an original circuit sequence length of 20000. We found through statistical analysis that all the original circuit sequences are less than 10,000 in length, so a lot of padding is needed for the circuit sequences. This will have a great impact on the original sequences and lead to a decrease in the accuracy of the classification results. In summary, we choose to perform one layer of DWT decomposition for the feature frequency domain processing. We believe that future additions to the website content and updates to the security mechanisms will result in longer circuit sequences during visits to the website. On this basis, WF attacks using multilayer DWT are expected to achieve a better result.

Table 6: Comparison of attack accuracy between FDF and other methods in the closed world.

Method	The accuracy of different dataset		
	Undefended (CW) (%)	WTF-PAD (CW) (%)	Onion Sites (CW) (%)
K-NN [7]	95.2	16.1	40.9
K-FP [9]	95.6	68.9	45.4
CUMUL [8]	97.5	60.1	47.2
DF [15]	98.3	90.9	53.0
Tik-Tok [18]	98.4	93.5	66.0
Proposed FDF	98.8	94.3	70.7

Table 7: The methods tuned for Precision and tuned for Recall on the Undefended (OW) dataset in the open world.

Method	Tuned for Precision		Tuned for Recall	
	Precision	Recall	Precision	Recall
DF [15]	0.986	0.931	0.929	0.983
Tik-Tok [18]	0.984	0.935	0.918	0.987
Proposed FDF (DCT)	0.973	0.922	0.909	0.981
Proposed FDF (FFT)	0.977	0.931	0.915	0.988
Proposed FDF (DWT)	0.990	0.943	0.931	0.991

Table 8: The methods tuned for Precision and tuned for Recall on the WTF-PAD (OW) dataset in the open world.

Method	Tuned for Precision		Tuned for Recall	
	Precision	Recall	Precision	Recall
DF [15]	0.973	0.736	0.719	0.958
Tik-Tok [18]	0.978	0.751	0.748	0.957
Proposed FDF (DCT)	0.953	0.718	0.692	0.938
Proposed FDF (FFT)	0.961	0.723	0.700	0.947
Proposed FDF (DWT)	0.982	0.756	0.751	0.961

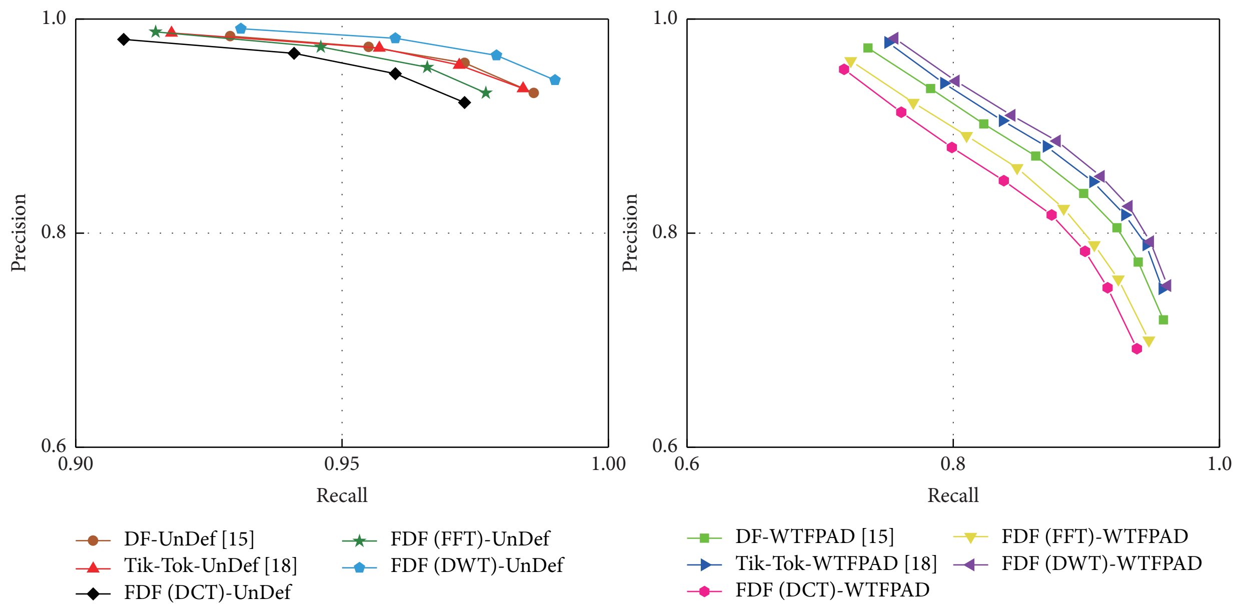


Figure 5: The Precision-Recall curve of attacks in the open world.

7. Conclusion

In this study, we propose an efficient DWT-based WF attack method FDF. We construct key features for traffic analysis by performing DWT on the length and directional features of circuit sequences. After that, we use neural networks to complete the learning and classification of the traffic frequency domain features. Overall, our results show that transforming circuit sequences to the frequency domain for deep learning can achieve good results. However, a large number of training sets are required for data support during the training process. This leads to longer data collection time and increases the difficulty of WF attacks. In the future, we should work to shorten the time to complete the fingerprint identification of the website. One possibility is to learn from the idea of the big data framework [32]. The fingerprint identification process of the website should be layered, especially the data collection process. Collect data through a distributed architecture and reasonably arrange the modules to add new data and delete old data. Ultimately, our methods can effectively respond to urgent tasks [33].

Data Availability

Previously reported Onion Service datasets were used to support this study and are available at DOI:10.1145/3133956.3134005. These prior studies (and datasets) are cited at relevant places within the text as references [12]. The closed-world datasets 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 National Natural Science Foundation of China (nos. U1804263, U1736214, and 62172435) and the Zhongyuan Science and Technology Innovation Leading Talent Project (no. 214200510019).

References

[1] "Users—tor metrics," 2010, https://metrics.torproject.org/userstats-relay-country.html.

[2] K. Shahbar and A. N. Zincir-Heywood, "Traffic flow analysis of tor pluggable transports," in *Proceedings of the 11th International Conference on Network and Service Management*, pp. 178–181, IEEE, Barcelona, Spain, November 2015.

[3] A. Montieri, D. Ciuonzo, G. Aceto, and A. Pescape, "Anonymity services tor, i2p, jondonym: classifying in the dark (web)," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 662–675, 2018.

[4] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, and A. Pescap, "A dive into the dark web: hierarchical traffic classification of anonymity tools," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1043–1054, 2019.

[5] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial Naïve-Bayes classifier," in *Proceedings of the ACM Workshop on Cloud Computing Security*, pp. 31–42, ACM, New York, NY, United States, November 2009.

[6] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pp. 103–114, ACM, New York, NY, United States, October 2011.

[7] T. Wang and I. Goldberg, "Improved website fingerprinting on Tor," in *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pp. 201–212, ACM, Berlin Germany, November 2013.

[8] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the USENIX Security Symposium*, pp. 143–157, USENIX Association, Sandieago CA, August 2014.

[9] A. Panchenko, F. Lanze, A. Zinnen et al., "Website fingerprinting at internet scale," in *Proceedings of the Network and Distributed System Security Symposium*, Febuary 2016.

[10] J. Hayes and G. Danezis, "k-fingerprinting: a robust scalable website fingerprinting technique," in *Proceedings of the USENIX Security Symposium*, pp. 1187–1203, USENIX Association, Sandieago CA, May 2014.

[11] H. Jahani and S. Jalili, "A novel passive website fingerprinting attack on tor using fast Fourier transform," *Computer Communications*, Elsevier, vol. 96, , pp. 43–51, 2016.

[12] R. Jansen, M. Juarez, R. Galvez, T. Elahi, and C. Diaz, "Inside job: applying traffic analysis to measure tor from within," in *Proceedings of the Network and Distributed System Security Symposium*, Febuary 2018.

[13] A. Shusterman, L. Kang, Y. Haskal et al., "Robust website fingerprinting through the cache occupancy channel," in *Proceedings of the USENIX Security Symposium*, pp. 639–656, USENIX Association, Sandieago CA, November 2014.

[14] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, MIT Press, vol. 11, , pp. 3371–3408, 2010.

[15] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1928–1943, ACM, Toronto Canada, October 2018.

[16] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: more practical and portable website fingerprinting with n-shot learning," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1131–1148, ACM, London United Kingdom, November 2019.

[17] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-CNN: a data-efficient website fingerprinting attack based on deep learning," *Proceedings on Privacy Enhancing Technologies* in *Proceedings of the Privacy Enhancing Technologies*, no. 4, pp. 292–310, Springer, Sandieago CA, Febuary 2019.

[18] M. S. Rahman, P. Sirinam, N. Mathews, K. G. Gangadhara, and M. Wright, "Tik-Tok: the utility of packet timing in website fingerprinting attacks," in *Proceedings on Privacy Enhancing Technologies*vol. 2020, no. 3, , pp. 5–24, Springer, 2020.

[19] R. Overdorf, M. Juarez, G. Acar, R. Greenstadt, and C. Diaz, “How unique is your .onion?: an analysis of the fingerprintability of tor onion services,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2021–2036, ACM, Dallas Texas USA, October 2017.
[20] *Tor: Onion Service Protocol*, https://2019.www.torproject.org/docs/onion-services, 2019.
[21] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, “Toward an efficient website fingerprinting defense,” in *Computer Security - ESORICS 2016*, I. Askoxylakis, S. Ioannidis, S. Katsikas, and C. Meadows, Eds., vol. 9878, , pp. 27–46, Springer, 2016.
[22] L. Li Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*vol. 28, no. 4, , pp. 594–611, IEEE, 2006.
[23] R. Dingledine and S. J. Murdoch, *Performance Improvements on Tor or, Why Tor Is Slow and what We’re Going to Do about it*, Technical report, The Tor Project, United States, 2009.
[24] P. Dhungel, M. Steiner, I. Rimac, V. Hilt, and K. W. Ross, “Waiting for anonymity: understanding delays in the tor overlay,” in *Proceedings of the 2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pp. 1–4, IEEE, Delft, Netherlands, August 2010.
[25] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Peek-a-Boo, I still see you: why efficient traffic analysis countermeasures fail,” in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 332–346, IEEE, San Francisco, CA, USA, May 2012.
[26] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, “A systematic approach to developing and evaluating website fingerprinting defenses,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 227–238, ACM, Scottsdale Arizona USA, November 2014.
[27] X. Cai, R. Nithyanand, and R. Johnson, “CS-BuFLO: a congestion sensitive website fingerprinting defense,” in *Proceedings of the Workshop on Privacy in the Electronic Society*, pp. 121–130, ACM, New York, NY, United States, November 2014.
[28] T. Wang and I. Goldberg, “Walkie-talkie: an efficient defense against passive website fingerprinting attacks,” in *Proceedings of the 26th USENIX Security Symposium*, pp. 1375–1390, USENIX Association, Vancouver BC, August 2017.
[29] C. Szegedy, W. Liu, Y. Jia et al., “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, IEEE, Boston, MA, June 2015.
[30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, IEEE, Las Vegas, NV, USA, June 2016.
[31] *The Top 500 Sites on the Web*, https://www.alexa.com/topsites, 2016.
[32] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, “A big data-enabled hierarchical framework for traffic classification,” *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2608–2619, 2020.
[33] Wtf-Pad, https://github.com/wtfpad/wtfpad, 2020.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 5203217, 12 pages
https://doi.org/10.1155/2022/5203217

Research Article

Deep Neural Embedding for Software Vulnerability Discovery: Comparison and Optimization

Xue Yuan,[1] Guanjun Lin,[2] Yonghang Tai ,[1] and Jun Zhang [1]

[1]*School of Physics and Electronic Information, Yunnan Normal University, Kunming 650000, China*
[2]*School of Information Engineering, Sanming University, Sanming, Fujian 365004, China*

Correspondence should be addressed to Yonghang Tai; taiyonghang@ynnu.edu.cn and Jun Zhang; junzhang@ynnu.edu.cn

Received 1 September 2021; Revised 31 October 2021; Accepted 6 November 2021; Published 18 January 2022

Academic Editor: Weizhi Meng

Due to multitudinous vulnerabilities in sophisticated software programs, the detection performance of existing approaches requires further improvement. Multiple vulnerability detection approaches have been proposed to aid code inspection. Among them, there is a line of approaches that apply deep learning (DL) techniques and achieve promising results. This paper attempts to utilize CodeBERT which is a deep contextualized model as an embedding solution to facilitate the detection of vulnerabilities in C open-source projects. The application of CodeBERT for code analysis allows the rich and latent patterns within software code to be revealed, having the potential to facilitate various downstream tasks such as the detection of software vulnerability. CodeBERT inherits the architecture of BERT, providing a stacked encoder of transformer in a bidirectional structure. This facilitates the learning of vulnerable code patterns which requires long-range dependency analysis. Additionally, the multihead attention mechanism of transformer enables multiple key variables of a data flow to be focused, which is crucial for analyzing and tracing potentially vulnerable data flaws, eventually, resulting in optimized detection performance. To evaluate the effectiveness of the proposed CodeBERT-based embedding solution, four mainstream-embedding methods are compared for generating software code embeddings, including Word2Vec, GloVe, and FastText. Experimental results show that CodeBERT-based embedding outperforms other embedding models on the downstream vulnerability detection tasks. To further boost performance, we proposed to include synthetic vulnerable functions and perform synthetic and real-world data fine tuning to facilitate the model learning of C-related vulnerable code patterns. Meanwhile, we explored the suitable configuration of CodeBERT. The evaluation results show that the model with new parameters outperform some state-of-the-art detection methods in our dataset.

1. Introduction

Software vulnerability has long been a severe but crucial research issue in cybersecurity [1–3]. These security vulnerabilities threaten the IT infrastructure of organizations and government sectors. There are increasingly more vulnerabilities being discovered. Multiple vulnerabilities released in the Common Vulnerabilities and Exposures were approximately 4,600 in 2010. However, it grows to approximately 153,955 in 2021. Software vulnerability [4–6], as a threat, is increasing in frequency, scale, and severity, which are similar to natural disasters; it may lead to unintended and severe consequences. Once vulnerability in a key system is exploited by attackers, millions of computer systems may be affected [7].

Despite the efforts that have been invested in pursuing the low probability of mistake when programming, software vulnerabilities remain and will continue to be a high-profile problem [8]. Recent years have witnessed a tremendous change in defending against vulnerabilities, from primarily reactive detection towards attempting to actively predict the code snippet whether it contains a vulnerability.

Deep learning-based vulnerability detection has attracted much attention recently. These techniques have been applied in the field of communications and networking and have achieved promising outcomes [9, 10]. For vulnerability detection tasks, neural networks are applied for automated feature extraction, which helps to improve the generalization ability of a model being capable of extracting high-level and latent features automatically. Therefore, researchers are motivated to improve the usefulness of deep learning-based vulnerability detection solutions from various aspects. The process of applying deep learning techniques in the context of vulnerability detection can be divided into four steps: data collection, data preparation, model building, and evaluation/test. Some existing vulnerability detectors with traditional embedding solutions such as Word2Vec, GloVe, and FastText often incur low precision and recall [11]. The bimodal feature of code [12] demands a model to be able to handle long-distance contextual dependencies. However, conventional embedding methods adopted by existing studies, such as Word2Vec, can only generate a unique embedding for a given code token and are unable to produce embeddings based on different contexts. Hence, we consider using CodeBERT as a code embedding and feature generator.

CodeBERT is capable of producing different embeddings based on different contexts. On the one hand, CodeBERT is based on a bidirectional transformer which can capture long-distance dependencies of code sequences. It can preserve the relationship between contexts, capture latent vulnerable code patterns, and minimize the loss of information. On the other hand, CodeBERT inherits the structure of multihead attention, which makes the model focus on multiple key points of a code sequence. When there is a loop in a code fragment, the value of a variable is constantly changing according to the loop condition; however, using the noncontextual embedding methods, the vector corresponding to this variable is constant. This means that the generated vector representation fails to represent the change of the value that happened to the variable. That is, the noncontextual embedding models can only generate a fixed representation for one word, incapable of producing different representations according to the difference of code contexts. This can be one of the reasons that the work [11] that uses Word2Vec as the code embedding method yielded relatively low precision.

In this paper, we fill this gap by incorporating a CodeBERT-based embedding solution for vulnerable function detection, and the frequently used acronyms in this paper are summarized in Table 1. Recent research has achieved impressive results on embedding source code by applying natural language techniques such as Word2Vec, GloVe, and FastText. CodeBERT is based on the structure of the bidirectional transformer, which can help the understanding of code semantics in a relatively large context. In addition, CodeBERT inherits the mechanism of multihead attention, employing 12 parallel attention heads, which allows the model to jointly attend to vulnerable features from different representation subspaces at various positions. Therefore, as an extractor, CodeBERT has the potential of generating more rich and meaningful code embeddings compared with noncontextual embedding methods such as Word2vec, GloVe, and FastText. CodeBERT achieved a promising result in many code processing/analysis tasks such as clone detection, defection detection, and natural language code search. However, it has not been used in the context of C language vulnerability detection. In summary, the contributions of the paper are three-fold:

(i) We perform a systematic evaluation on several mainstream code embedding solutions including Word2vec, GloVe, FastText, and CodeBERT. We discover that utilizing CodeBERT as a code embedding solution yielded the best performance in terms of vulnerability detection in C open-source projects.

(ii) We apply synthetic vulnerability data derived from SARD (Software Assurance Reference Dataset) to fine tune the parameters of CodeBERT and to address the data imbalance problems frequently faced in vulnerability detection in practice. Experiments demonstrate that applying synthetic vulnerability data can further improve the usefulness of codeBERT and achieve optimized results in vulnerability detection.

(iii) We examine important parameters of CodeBERT in terms of code feature extraction and evaluate the optimal parameters identified for effective vulnerability detection.

The rest of this paper is organized as follows. Section 2 presents some existing studies for deep learning-based vulnerability detection. In Section 3, the research framework, the code representation learning, the process of fine tuning, and the impact of various sequence-length for fine tuning CodeBERT are presented. Section 4 is our evaluation and the analysis of experimental results. In Section 5, we conclude the present paper and discuss the limitations of the proposed scheme and open problems for future research.

2. Related Work

In the field of software vulnerability detection, various techniques have been proposed. There have been several survey articles providing systematic reviews of many approaches in this field from various perspectives [5, 28–32].

Meanwhile, we review several existing deep learning-based vulnerability detection studies on different neural networks (see Table 2). Many deep learning architectures include convolutional neural network (CNN) [13–16], deep belief network (DBN) [17], multilayer perceptron (MLP) [18, 19], long short-term memory (LSTM) [20–22], and gated recurrent unit (GRU) [11]. In addition, a large body of studies focuses on employing different embedding techniques for generating vector representations as input for the training process. A summary of the reviewed studies is shown in Table 3. Pradel and Sen [23] used Word2Vec for generating code vectors derived from the custom Abstract Synthetic Trees (ASTs)-based contexts. These vectors were used to train deep learning models to detect vulnerabilities in JavaScript code. The Word2Vec was applied for making vector representations from C/C++ source code [13].

Instead of using Word2Vec, Henkel et al. [24] applied the GloVe model to produce vectors learned from the Abstracted Symbolic Traces of C programs. Furthermore, FastText was used in FastEmbed [25] for vulnerability prediction based on ensemble machine learning models.

Table 1: List of the acronyms involved in the manuscript.

Acronyms	Definition	Acronyms	Definition
DL	Deep learning	NLP	Natural language processing
ML	Machine learning	MLM	Masked language modeling
SARD	Software assurance reference dataset	RTD	Replace token detection
DBN	Deep belief network	NVD	National vulnerability dataset
LSTM	Long short-term memory	CVE	Common vulnerabilities and exposures
BiLSTM	Bidirectional long short-term memory	GRU	Gated recurrent unit
ELMo	Embeddings from language models	CBOW	Continuous bag-of-words model
CuBERT	Code understanding BERT	CNN	Convolutional neural network
MLP	Multilayer perceptron		

Table 2: Reviewed studies which applied various neural networks for software vulnerability detection.

Neural network	Paper
Convolutional neural network	Harer et al. [13], Lee et al. [14], Russell et al. [15], and Wu et al. [16]
Deep belief network	Wang et al. [17]
Multilayer perceptron	Lin et al. [18] and Shar and Tan [19]
Long short-term memory	Li et al. [20], Lin et al. [21], and Lin et al. [22]
Gated recurrent unit	Lin et al. [11]

Table 3: Reviewed studies which applied various embedding techniques for software engineering.

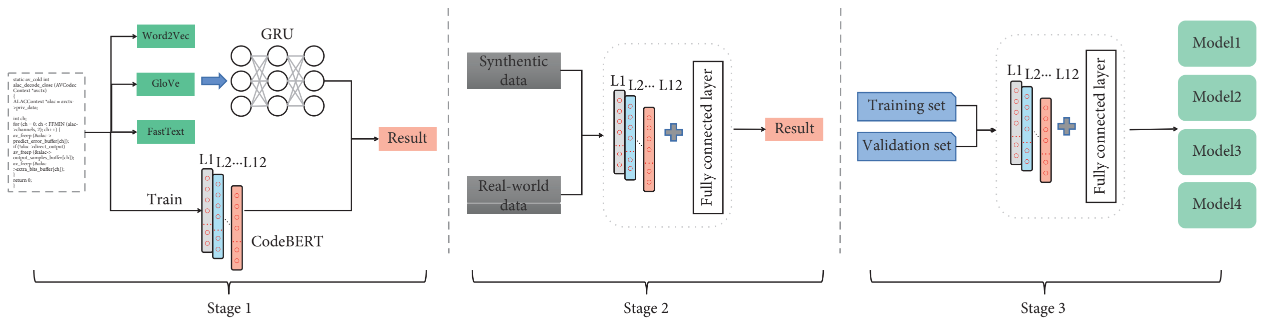
Paper	Type of data	Embedding model	Whether to consider contextual information
Pradel and Sen [23]	150,000 JavaScript files collected from various open-source projects	Word2Vec	No
Harer et al. [13]	C/C++ packages distributed with the Debian Linux distribution C/C++ functions collected from github	Word2Vec	No
Henkel et al. [24]	19,000 API-usage analogies extracted from the Linux kernel	GloVe	No
Fang et al. [25]	Projects are extracted from open-source intelligence data such as NVD	FastText	No
Kanade et al. [26]	150k Python files from github	CuBERT	Yes
Karampatsis and Sutton [27]	150,000 JavaScript files consisting of various open-source projects	SCELMo	Yes

Recently, several contextualized embedding models have been applied for generating code representations. Kanade et al. [26] proposed CuBERT (Code Understanding BERT), which generates contextual embeddings by training a BERT model on software source code. Karampatsis and Sutton [27] proposed a model named SCELMo to generate contextual code representations. Both studies have proved that contextualized embedding models are effective for various code analysis tasks. In our work, we utilize CodeBERT, which is also a pretrained contextualized model on six programming languages, for a specific code analysis task which is vulnerability detection.

3. Methodology

3.1. Research Framework. We collect several dissimilar software projects and create the ground truth dataset; meanwhile, we select multiple synthetic vulnerabilities at the function level. Figure 1 presents our workflow. In detail, it includes three stages. Firstly, the source code is loaded and transformed to JSON format that is recognized by CodeBERT. For conventional models, the source code files are loaded and processed to generate sequence data and labels. The data is then passed to the next stage to be transformed into the code embedding vectors. These vectors will then be partitioned and fed to the GRU neural network for the training process. We utilized a system to train code vulnerability detectors for evaluating three traditional embedding methods. The system was built based on the open-source API benchmark proposed by [11]. Secondly, we feed CodeBERT with the synthetic data to obtain a fine-tuned model; the vulnerable probabilities of the test set were generated correspondingly. Thirdly, based on fine tuning, we evaluated the impact of the various parameters such as the length of the input sequence, batch size, epoch, and learning rate. A suitable input sequence length for extracting vulnerability features from programs written in C is finally determined. In order to verify the proposed method is successful in producing the desired result, we perform the tasks towards answering the following three research questions. We will address each question with the results.

Research Question 1 (RQ1): how effective is CodeBERT when compared with other embedding methods? This research question is meaningful because one may argue that CodeBERT cannot be used to extract features of programs written in C. For



```
static av_cold int
alac_decode_close (AVCodec
Context *avctx)
{
ALACContext *alac = avctx-
>priv_data;

int ch;
for (ch = 0; ch < FFMIN (alac-
>channels, 2); ch++) {
av_freep (&alac->
predict_error_buffer[ch]);
if (!alac->direct_output)
av_freep (&alac->
output_samples_buffer[ch]);
av_freep (&alac-
>extra_bits_buffer[ch]);
}
return 0;
}
```

Figure 1: The workflow consists of three stages. In stage 1, we compare the performance of four models. In stage 2, a synthetic dataset derived from SARD projects is poured into the real-world dataset to fine tune the parameters of CodeBERT. In stage 3, we examine important parameters of CodeBERT in terms of code feature extraction.

answering this question, we will compare CodeBERT with other approaches, including Word2vec, GloVe, and FastText.

Research Question 2 (RQ2): how to improve the effectiveness of CodeBERT for vulnerability detection? To answer this question, we apply a real-world dataset and a synthetic dataset to fine tune CodeBERT and conduct comparative experiments.

Research Question 3 (RQ3): how does the input sequence length affect CodeBERT? We compare the performance of the fine-tuned model with various sequence lengths on the same classification tasks.

3.2. Code Representation Learning. Word2Vec, GloVe, FastText, and CodeBERT are embedding models for converting textual tokens to meaningful vectors; they have some similarities and differences. We compare four models from the following perspective.

From the perspective of model structure, in general, CodeBERT has a complex structure, while the other three models have simple structures. Word2Vec contains two models: Skip-gram and CBOW. Skip-gram employs middle words to predict nearby words, and CBOW applies context words to predict middle words. Both models have three layers, which are the input layer, mapping layer, and output layer. The hidden layer has a linear structure, so it is very fast to train. Word2Vec generates a unique vector for each word in the corpus; however, it ignores the connection between words, for example, "apple" and "apples." The two words are similar, that is, their internal morphology is close. However, both words are converted by Word2Vec; this internal form will be ignored. In order to avoid this problem, FastText uses character-level n-grams to represent a word. The FastText model also has three layers: an input layer, a hidden layer, an output layer, and the network structure is relatively simple. GloVe uses the matrix factorization method, and the training speed is also very fast. However, it only focuses on co-occurrence; the generated word vector contains limited semantic information and is only suitable for limited tasks such as similarity calculation. Compared with the three models mentioned above, the model structure of CodeBERT is much more complicated. CodeBERT is based on a multilayer two-way transformer. Specifically, CodeBERT contains 12 layers, each layer has 12 self-attention heads, the size of each self-attention head is set as 64, and the hidden dimension is set as 768. These differences in model structure and computer system will affect the detection of vulnerabilities.

From the perspective of outcome, for a word, CodeBERT can generate different vectors according to context information, however, Word2Vec, GloVe, and FastText cannot. This perspective is important because a target word may express multiple contents, meaning that an embedding model that is noncontextual would be too limited.

Context information is crucial for analyzing vulnerabilities of many different types. A motivating example is shown in Figure 2. In the vulnerable code (see Figure 2(a)), both a and b are short types, and $b = b + a$ may cause b to exceed its range. In the corrected function (see Figure 2(b)), the range of a is limited so that it can only be less than 0, which can avoid this error. In the two functions, the range of a is different. For different contexts, a certain variable has various meanings.

This paper applies CodeBERT for extracting high-level code representations for detecting vulnerable functions written in C. Traditional models such as Word2Vec, GloVe, and FastText are usually applied to convert a word into a fixed vector [33], regardless of the variation of the values of these variables. This will affect the correct understanding of code semantics. In CodeBERT, a pretrained model for natural language and programming language, two objectives are used for training. The first objective is Masked Language Modeling (MLM) and the second is to replace token detection (RTD). RTD further uses a large number of unimodal data, such as source codes without paired natural language data. For vulnerability detection, we take advantage of the second objective. CodeBERT is considered as an embedding model proposed in [34] which is based on the concept of transformer [35]. Furthermore, we leverage the component of the encoder; the encoder consists of a multihead attention layer and feedforward network. Designed to improve the ability to retain useful information, residual connections are used for two layers. CodeBERT has not learned the patterns of C language; by using transfer learning, CodeBERT makes practical and effective use of the relevant syntactic and semantic information learned from other programming languages.

```
short add (short a)
{
    short b = 32767;
    b = b + a;
    return b;
}
```

(a)

```
short add (short a)
{
    short b = 32767;
    if (a<0){
        b = b + a;
    }
    return b;
}
```

(b)

Figure 2: The example of C functions. (a) A vulnerable function. (b) Tthe revised function.

For many embedding tasks, checking the contiguous information is generally insufficient to generate semantically rich vector representations [36], vulnerable patterns usually contain declarations, assignments, control flow, and other operational logic. Hence, it is necessary to pay attention to multiple key points in the vulnerable functions. Multihead attention enables the algorithm to focus on multiple key points, which facilitates the capture of potentially vulnerable program patterns. In addition, the pattern of vulnerabilities is long-term dependent, meaning that long-distance contextual information is crucial for vulnerable function detection. For vulnerable functions, words, such as X_{i-n} and X_{i+n} of the word X_i, can also be useful. To obtain the dependencies of surrounding words of the word, the positional embedding is designed to serve this purpose. A positional encoding layer is added to the input embedding layer, and the dimension of positional encodings d_{model} is the same as the embeddings so that both parameters can be summed. Due to the source code structure being according to the rules of logic or formal argument and relating to meaning in language or logic, it is closely connected and tightly coupled. Hence, the occurrence of a vulnerable code fragment is usually having constituent parts linked or connected to either previous or subsequent code, or even to both. A tiny vulnerable code snippet usually holds multiple lines of code that can be distributed across a function block [37]. For traditional embedding models such as Word2Vec [38], it is needing much effort or skill to exactly pinpoint which line of code gives rise to software programs; however, with the structure of the transformer, it is presenting few difficulties to confirm the distance between each word. Hence, the structure of CodeBERT can make it easier for the model to detect a long-term dependency of both forward and backward by offering services, which can effectively capture the vulnerable programming patterns. CodeBERT has not been trained in C language; however, by using transfer learning, CodeBERT is capable of utilizing the relevant patterns from other programming languages to be applied to a related detection task.

We explored four embedding models for feature extraction from source code. Word2Vec: the dimension of the feature vector is set at 100 and the maximum distance between the current and predicted word within a sentence is 5. If a word appears less than 5 times, the word will be ignored. GloVe: dimensionality of the output word vectors is set at 100; the maximum distance between the current and predicted word within a sentence is 5. The number of iterations over the corpus is set at 40. The learning rate for training is set as 0.001. FastText: the dimension of the feature vector is set as 100. The maximum distance between a current and predicted word within a sentence is 5. Ignore the words with a total frequency lower than 5. The number of epochs is set at 20. CodeBERT: the number of epochs is set at 5. The default sequence length is 400. The number of data samples captured in one training session is 4, and we used the Adam optimizer throughout. The default learning rate is $2e-5$. The architecture of the CodeBERT-based embedding model is illustrated as follows:

Input layer: it feeds the open-source code to this model

Embedding layer: source codes are transformed into a low dimension vector

Encoder layer: 12 encoder layers are employed to learn high-level feature representation

Fully connected layer: only used for fine tuning

Output layer: it delivers the learned high-quality description of features for vulnerability detection

3.3. Synthetic Data Fine Tuning. In this paper, a fine-tuned solution is provided to allow CodeBERT to learn the syntax and structure of the C programming language and capture the semantics of C code. CodeBERT is pretrained in 6 programming languages which does not include the C programming language. CodeBERT requires to be familiar with the code pattern of the C language. Through transfer learning, CodeBERT does not need to use C language data to train; it only requires following the fine-tuning strategy. Fine-tuning CodeBERT requires a large number of samples and corresponding labels. Manually picking the faults on the real vulnerable functions is time-consuming. It is difficult to obtain a lot of real data to fine tune the CodeBERT. The reasons for adding an artificial synthetic dataset are as follows. First of all, the synthetic dataset has sufficient quantity and diversity. It includes basic code patterns and syntax. In addition, it possesses accurate labels which are more beneficial to the training and optimization of the model. We are using the dataset named SARD; synthetic vulnerable functions are poured off the training set and validation set, respectively, so that, in the total dataset, the vulnerability functions are one-tenth of the total functions. In addition, this also solves the issue of data imbalance for us. In reality, for multiple classification tasks, such as financial fraud and fault diagnosis, the data are often imbalanced. The model receives the constraints of the data distribution and learns more of the features of the majority class, while ignoring the features of the minority class. This leads to a decrease in the classification performance of the model. After adding artificially synthesized vulnerability data, the ability of the model to extract vulnerability features can be improved.

Regarding the mixed dataset settings [39], we divide the real-world dataset into the training set, validation set, and test set according to the ratio of 6 : 2 : 2; the vulnerable functions in these datasets are 1189, 395, and 399, respectively. Among the synthetic dataset, 7486 and 2495 vulnerable functions are selected to pour into the training set and validation set, respectively, making sure that the vulnerable functions account for 1/10 of total functions. No synthetic vulnerable function was added to the test set.

3.4. Evaluate the Impact of Fine-Tuned CodeBERT with the Various Sequence Length. Source code functions have varying lengths when they are converted to sequences, and initial experiments suggest that using different sequence lengths exerted tremendous influence on detection results. When mapping code to vectors, the over-long codes are needed to truncate when converting codes to vectors of the fixed length for balancing between excessively long vectors and information loss. If a code sequence of a function is not long enough, it is padded with 1 s. The sequence length affects the performance of the CodeBERT model significantly. When the input sequence is too short, a large number of functions will be truncated, resulting in loss of information. The model cannot fully learn the features of functions. What it learns may only be the declaration or the definition of the variables. When the sequence is too long, a large number of functions will be filled with 1 at the end, and useless information occupies most of the space of the input sequence. In addition, there are also long-distance dependencies within the programs, given that some vulnerable functions may lie many sentences away from their locus of attention. Due to the bidirectional structure of CodeBERT, when the distance is too long, previous information may slip from the model's memory. Therefore, an appropriate input sequence length is required, and it will make the model focus exactly on the feature of the vulnerability.

The lengths of functions in the real-world dataset are depicted in Table 4; the entire real-world dataset includes 132,018 functions. We observed that approximately 44.4% of samples are within 128 elements in length. There are 90,696 functions less than 256, accounting for 68.7% of the total. Although the highest proportion is elements within 128, 128 is not necessarily the most suitable input sequence length. We need to make a trade-off between the different lengths of function code sequences.

We fine-tuned CodeBERT using different sequence lengths (block size) of 128, 256, 384, and 512, respectively. The model was firstly fine tuned, and the Adam optimizer with a learning rate of $2e-5$ is applied. Considering the limitation of GPU memory, the batch size is set to 4. Besides, a relatively small batch size helps the generalization of the model. The configuration is one of the many hyperparameters we tuned to obtain the optimal parameters. CodeBERT has been pretrained on six different programming languages, having weights to be initialized. Therefore, fine tuning is needed to allow the model to fit the specific task.

4. Experiment and Evaluation

4.1. Dataset. We believed that vulnerabilities are generally reflected in the pattern of source code, particularly at the function level. Hence, we focus on function-level vulnerabilities in this paper. The statistics of the aforementioned datasets are presented in Table 5.

4.1.1. Real-World Dataset. The real-world dataset used for evaluation consists of 12 popular open-source software projects and libraries. There are Asterisk, Httpd, ImageMagick, LibPNG, LibTIFF, OpenSSL, Pidgin, qemu, samba, VLC Player, and Xen. This dataset was built by Lin et al. [11] and was further extended to form a dual-granularity vulnerability detection dataset, providing vulnerabilities at the file and function level. For vulnerable files and functions, labels were manually attached based on the records and description of the National Vulnerability Dataset (NVD) and Common Vulnerabilities and Exposures (CVE). For the experiments, we acquire 1,983 vulnerable functions and 130,035 nonvulnerable functions from the dataset. The real-world vulnerability dataset enables classifiers to learn real-world vulnerable patterns.

4.1.2. Synthetic Dataset. The synthetic vulnerability dataset contains function samples derived from the Software Assurance Reference Dataset (SARD) project. Samples contain artificially constructed code fragments based on currently known vulnerable source code patterns. Meanwhile, each test fragment comprises one main function to guarantee the fragment of code is compilable. The synthetic dataset enables classifiers to learn the simplified and straightforward vulnerable patterns.

4.2. Experiment Settings. The implementation of CodeBERT is based on Pytorch (1.7.1) backend, and we implement the Word2Vec, GloVe, and FastText in Python using Keras with TensorFlow (1.14.0). We carry out experiments on a machine with NVIDIA GeForce GTX 1070 GPU and an Intel Core i7-6700k CPU operating at 4.00 GHz.

The comparison of four models: the real-world dataset is applied to evaluate the effectiveness of code features extracted by four embedding models, namely, Word2vec, FastText, GloVe, and CodeBERT. There are a total of 132,018 functions across 12 open-source projects, of which 1,983 functions are vulnerable. The selected source code function samples are divided into three sets, training set, validation set, and test set, with a ratio of 6 : 2 : 2. Fine-tuning stage: a contrast experiment was conducted to demonstrate the effectiveness of the proposed method that

Table 4: The statistics on code lengths for the total real-world dataset and test set involved in experiments. The functions are divided into five categories according to length.

Length of functions	<128	≥128 and <256	≥256 and <384	≥384 and <512	≥512
No. of samples (% of total sets)	58,556 (44.4%)	32,140 (24.3%)	15,052 (11.4%)	8,065 (6.1%)	18,205 (13.8%)
No. of samples (% of test set)	11,735 (44.4%)	6,446 (24.4%)	2,944 (11.1%)	1,630 (6.2%)	3944 (13.9%)

Table 5: The vulnerable functions and nonvulnerable functions are elaborated in this table. The datasets are derived from 12 open-source projects written in C programming language and the Software Assurance Reference Dataset (SARD) project which contains artificially constructed test cases. In the real-world dataset, the vulnerable functions are labeled based on the description of CVE and NVD. The first column lists the name of the dataset, the second column lists the projects, and the last two columns list the number of vulnerable functions and nonvulnerable functions, respectively.

Data source	Dataset/collection	No of functions used/collected	
		vulnerable	Nonvulnerable
Test cases from the SARD projects	C source code samples	83710	52290
	Asterisk	94	17620
	FFmpeg	249	5549
	Httpd	57	3843
	ImageMagic	344	2361
	LibPNG	45	577
	LibTIFF	123	726
Real-world open-source projects	OpenSSL	159	7004
	Pidgin	29	8547
	qemu	143	36063
	samba	26	32819
	VLC Player	44	6013
	Xen	670	8913
	Total	1983	130035

helps to fine-tune CodeBERT. Among the functions listed in Table 6, we used the training set and validation set to fine-tune CodeBERT; subsequently, we fed the model with the test set to obtain a CSV file that sorts functions according to the probability of being vulnerable. Evaluation of various parameters: we employed the previously used three sets in the fine-tuning stage as the dataset for evaluating the optimal parameters.

4.3. Evaluation Metrics. For classification tasks, precision and recall are both mainstream evaluation metrics. However, there are significantly more nonvulnerable functions than vulnerable ones. The proportion is approximately ninety six to one. The severe data imbalance may let the classifier focus on the majority class, while ignoring the minority one during the training process. To correctly monitor the performance, we apply the top-k percentage precision ($P@K\%$) and top-k percentage recall ($R@K\%$) to evaluate the effectiveness of the proposed methods. This standard of measurement is widely used in the research of information retrieval systems of top-k retrieved documents.

$P@K\%$ alludes to the data in the test set, that is, the vulnerable data, and has been successfully identified by the vulnerability detector. $R@K\%$ represents the percentage of $K\%$ data in the test set, that is, the data of vulnerability. They can be calculated by the following two mathematical expressions:

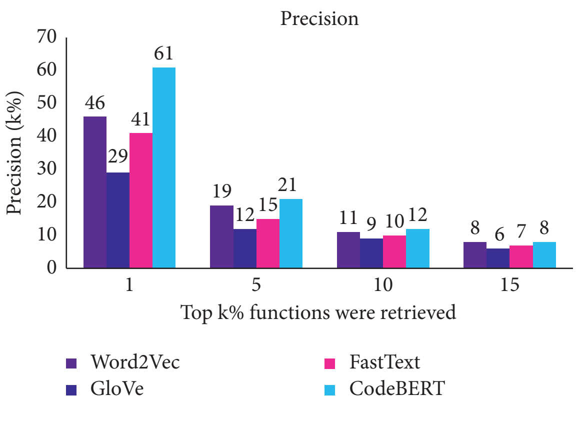
$$\begin{aligned} P@K\% &= \frac{TP@K\%}{TP@K\% + FP@K\%}, \\ R@K\% &= \frac{TP@K\%}{TP@K\% + FN@K\%}. \end{aligned} \quad (1)$$

4.4. Results and Analyses. There are various studies dedicated to addressing the issue of detecting vulnerabilities, such as the system proposed in [11] and an open-source detector named Flawfinder [40]. These systems are applied as the baselines because we have full access to the code and dataset, and Flawfinder is a well-known open-source tool which is widely used in practice. We structure the assessment by completing three research questions step by step.

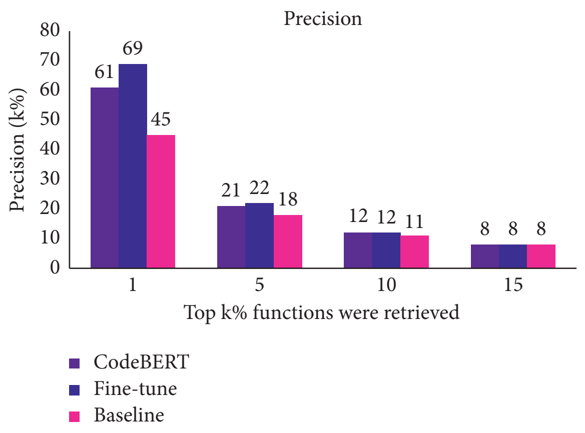
RQ1: selecting the suitable embedding method can be a critical task since it can affect the performance of the vulnerability detectors. Thus, we compare the effectiveness of CodeBERT with other traditional embedding models to determine the most viable embedding model for vulnerability detection. We here report the results of applying four embedding models in detecting vulnerable functions written in the C programming language. Figure 3(a) elaborates the comparison of precision obtained from four embedding models, which are 46%, 28%, 41%, and 61%, respectively, when retrieving the top 1% of most probably vulnerable functions. Figure 3(c) shows that

Table 6: The number of vulnerable functions and nonvulnerable functions when fine tuning the parameters of CodeBERT. In the training set and verification set, aiming to make vulnerable functions account for 1/10 of the total number of functions, we added synthetic data to the original dataset.

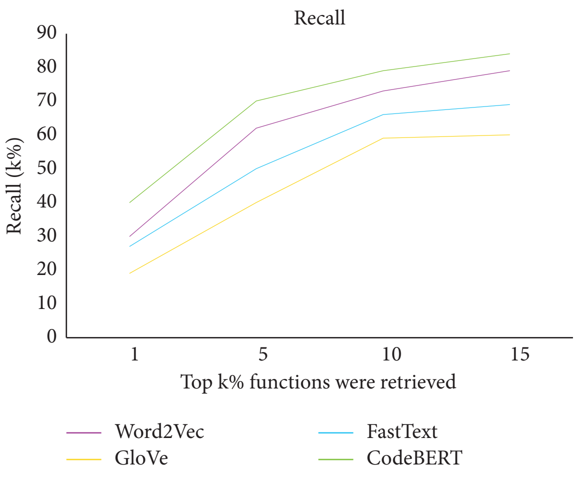
Dataset	No of vul. Functions (real-world | SARD)	No. of total functions
Training set	8675 (1189 | 7486)	86759
Validation set	2891 (395 | 2495)	28919
Test set	399 (399 | 0)	26425



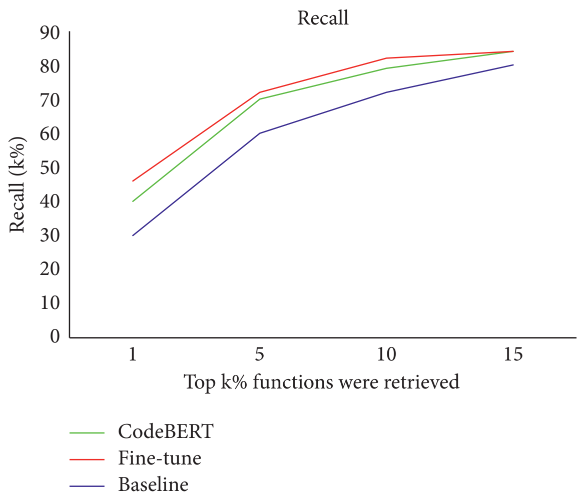
(a)



(b)



(c)



(d)

Figure 3: Results of two comparative experiments. (a) and (c) The precision and recall of several embedding methods, respectively; (b) and (d) the precision and recall of three models.

the green line which is the recall obtained by CodeBERT lies above the other three lines being the recall achieved by Word2Vec, GloVe, and FastText. CodeBERT could identify 40% of total vulnerable functions when retrieving 1% of functions. With Word2Vec, only 30% of total vulnerable functions were found. When using the FastText and GloVe models, only 27% and 20% of actual vulnerable functions could be found. This indicates that compared with the other three models, code embeddings generated by CodeBERT facilitate vulnerability detection.

In addition, we considered the computational complexity of CodeBERT. To directly measure the computational complexity of CodeBERT is a challenging task because the implementations of the encoder structure of transformer and the multihead attention mechanism are encapsulated by the deep learning framework (e.g., PyTorch). Therefore, we chose to compare the training and test time of CodeBERT with those of other embedding methods and measured the efficiency of CodeBERT and other embedding methods. By comparison, how computationally complex codeBERT it is can be evaluated. Table 7 summarizes the training (measured in one epoch) and test time of CodeBERT and the other three models, respectively. We observe that it took CodeBERT 6720 seconds to complete one epoch during the training phase. In contrast, the training times of the three embedding models were similar, which did not exceed 300 seconds. The noncontextual model (Word2Vec, GloVe, and FastText) outperforms CodeBERT in terms of training time. This can be explained by the fact that the CodeBERT model structure is complex.

RQ2: to evaluate whether the performance of CodeBERT can be improved via fine tuning. We add a fully connection layer at the bottom of the output. Meanwhile, to further fine tune the parameters, we apply a synthetic vulnerability dataset derived from software.

Assurance Reference Dataset (SARD) project: the synthetic dataset contains artificially defined test cases to simulate vulnerable code patterns. As shown in Figure 3(b) and Figure 3(d), when retrieving 1% of vulnerable functions, we observe that, after fine tuning, there is an 8% of precision improvement, and the fine-tuned model could find 84% of total vulnerable functions when returning 15% of potentially vulnerable functions. In addition, the fine-tuned model is more effective than previous models such as the vulnerability detector [11]. When retrieving 1% of total functions, the improvement in each of the metrics is substantial, that is, 24% in precision and 16% in recall. It was proved that the performance of the detectors was improved by using the fine-tuned approach.

RQ3: the purpose of this experiment is to understand how much suitable sequence length help. We construct experiments to demonstrate the effectiveness of suitable sequence lengths for vulnerability detection. Tables 8 and 9 show the results of models with various sequence lengths. The testing set contains 26,425 functions among which there are 399 vulnerable functions. For detecting software vulnerabilities written in C language, we can conclude that the most satisfying sequence length is 256. Because whether it is retrieving the top 1%, 5%, 10%, or 15%, the model works best when the sequence length is 256. When examining the top 1% of total functions, the model with a sequence length of 128 only incurs a precision of 26%.

For a fair comparison, we analyzed the results obtained above with a baseline. We observe that the output of Flawfinder is ranking the functions according to their vulnerable level. The results of the proposed method are ranking functions according to the probability of being vulnerable. Hence, we applied the abovementioned performance metrics to describe both detectors. We here report the comparison between their effectiveness in detecting real world.

Vulnerabilities: Tables 8 and 9 show the comparison results; the model with a sequence length of 256 substantially outperforms the FlawFinder. The model with a sequence length of 256 incurs an $P@1\%$ of 70% and $R@1\%$ of 47%; however, when we retrieved the top 1% of functions in the test set that contains 399 vulnerable functions, Flawfinder only achieved 3% precision and 2% recall. Retrieving 5%, 10%, and 15% of functions ranked by the vulnerable level, the results of the Flawfinder are also poor. This phenomenon further highlights the effectiveness of the CodeBERT-based embedding solution for vulnerability detection.

5. Discussion

This section discusses the balance between model effectiveness and model complexity. To directly measure the computational complexity of CodeBERT is a challenging task because the implementations of the encoder structure of transformer and the multihead attention mechanism are encapsulated by the deep learning framework. We take an epoch as an example to show the time overhead incurred by the process of training. The training time corresponding to these four models (CodeBERT, Word2Vec, GloVe, and FastText) was, respectively, 6720 s, 287 s, 285 s, and 286 s; it is clear that CodeBERT takes the longest. The underlying reason is that the structure of CodeBERT is more complex than the other three models, and CodeBERT has a large capacity; it may save a large body of information that other models have not captured, such as some potential code patterns and semantic features. In addition, the parameter is complex and requires a lot of calculation, so it takes more time. Even so, CodeBERT can improve the precision and recall of vulnerability detection tasks; it is still worth selecting.

In addition, when using CodeBERT extracting code features, different sequence lengths exhibited varying performances; the statistic is shown in Tables 8 and 9. In the current task, the balance between functions length and the input sequence length is also a challenge. We discuss the possible causes of performance behavior of different sequence lengths. Compared with other lengths, the sequence length of 256 outperformed the real-world dataset. The underlying reason is that the vulnerable functions contain information such as a head file, variable declaration, parameters, logic code, and return value. If the sequence length is too short, the obtained features may only include information such as header files and variable declarations. Vulnerability features are usually hidden in the logic code. Therefore, the features of the vulnerabilities may be cut off, and the model learns all the useless features. When the sequence length is longer than 256, performance is not satisfactory; the reason for this phenomenon may be that the length of most vulnerable functions is within 256; if the sequence length is set too long, the sequence will automatically be filled with 1. Too much irrelevant information will interfere with the model's judgment on the features of the vulnerability; meanwhile, models will focus on irrelevant information. In general, setting the sequence length to 256 can greatly reduce the redundancy and loss of information.

Table 7: The complexity of training and test when we compare four models.

Models	Training time per epoch (s)	Test time per epoch (s)	Number of epochs	Total training time with all epoch completed (s)
CodeBERT	6720	600	5	33600
Word2Vec	287	32	150	43050
GloVe	285	32	150	42750
FastText	286	32	150	42900

Table 8: Test precision of various sequence lengths against Flawfinder on the same classification tasks.

Precision calculated when top-k% functions were retrieved					
		1 (%)	5 (%)	10%	15 (%)
Different sequence lengths	Block size = 128	26	7	6	5
	Block size = 256	70	22	12	8
	Block size = 384	56	18	10	9
	Block size = 512	63	20	12	8
Flawfinder		3	3	7	7

Table 9: Test recall of various sequence lengths against Flawfinder on the same classification tasks.

Recall calculated when top-k% functions were retrieved					
		1 (%)	5 (%)	10 (%)	15 (%)
Different sequence length	Block size = 128	12	25	37	46
	Block size = 256	47	74	81	86
	Block size = 384	37	60	69	78
	Block size = 512	41	66	78	85
Flawfinder		2	2	45	45

6. Conclusions and Future Work

This paper proposes an embedding solution for vulnerability detection which is based on CodeBERT. CodeBERT is not familiar with the syntax and semantic of the C language; however, it has been pretrained on other programming languages and has great potential of learning effective features of the C language. We have employed C open-source projects and manually constructed functions written in C to fine-tune CodeBERT. Meanwhile, we have constructed a useful real-world dataset for evaluating and estimating the ability and quality of the solution and other deep learning-based vulnerability detectors that will be expanded in the future. The experimental results show that the proposed embedding solution can achieve precision that exceeded expectation when returning K (1, 5, 10, 15)% of total functions, indicating that the approach is capable of facilitating the detection of vulnerabilities in C open-source projects.

The present scheme can be further improved by addressing the following limitations. Firstly, CodeBERT consists of 12 encoder layers and has approximately 110 million parameters, which are expensive to train and deploy. Therefore, proposing a lightweight model could be valuable. Secondly, the present embedding solution for vulnerability detection is limited to dealing with the software code written in C# and C++. Further research could be conducted to adapt to more programming languages. Thirdly, there is still lack of a large real-world dataset providing multiple detection granularities. Further research effort could be developing an automated vulnerability data labeling solution to speed up the data collection process.

Data Availability

The public dataset is available online for research (https://cybercodeintelligence.github.io/CyberCI/).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Xue Yuan and Guanjun Lin contributed equally to this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant no. 62062069), Optoelectronic Information Technology Key Laboratory Open Project Fund of Yunnan Province, China, under Grant YNOE-2020-01, and Natural Science Foundation Project of Fujian Province, China, under Grant 2021J011131.

References

[1] M. Wang, T. Zhu, T. Zhang, J. Zhang, S. Yu, and W. Zhou, "Security and privacy in 6g networks: new areas and new challenges," *Digital Communications and Networks*, vol. 6, no. 3, pp. 281–291, 2020.

[2] Y. Miao, C. Chen, L. Pan, Q.-L. Han, J. Zhang, and Y. Xiang, "Machine learning-based cyber attacks targeting on controlled information," *ACM Computing Surveys*, vol. 54, no. 7, pp. 1–36, 2022.

[3] X. Chen, C. Li, D. Wang et al., "Android hiv: a study of repackaging malware for evading machine-learning detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 987–1001, 2019.

[4] G. Lin, W. Xiao, L. Y. Zhang, S. Gao, Y. Tai, and J. Zhang, "Deep neural-based vulnerability discovery demystified: data,

model and performance," *Neural Computing and Applications*, pp. 1–14, Springer, New York, NY, USA, 2021.

[5] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, "Detecting and preventing cyber insider threats: a survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1397–1417, 2018.

[6] D. Votipka, R. Stevens, E. Redmiles, J. Hu, and M. Mazurek, "Hackers vs. testers: a comparison of software vulnerability discovery processes," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, pp. 374–391, IEEE, San Francisco, CA, USA, May 2018.

[7] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257–1270, 2014.

[8] S. Liu, G. Lin, L. Qu et al., "Cd-vuld: CD-VulD: cross-domain vulnerability discovery based on deep domain adaptation," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[9] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[10] T. O'shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[11] G. Lin, W. Xiao, J. Zhang, and Y. Xiang, "Deep learning-based vulnerable function detection: a benchmark," in *Proceedings of the International Conference on Information and Communications Security*, pp. 219–232, Springer, Beijing, China, December 2019.

[12] M. Allamanis, D. Tarlow, A. Gordon, and Y. Wei, "Bimodal modelling of source code and natural language," in *Proceedings of the International Conference on Machine Learning*, pp. 2123–2132, PMLR, Lille, France, July 2015.

[13] J. A. Harer, L. Y. Kim, R. L. Russell et al., "Automated software vulnerability detection with machine learning," 2018, https://arxiv.org/abs/1803.04497.

[14] Y. J. Lee, S.-H. Choi, C. Kim, S.-H. Lim, and K.-W. Park, "Learning binary code with deep learning to detect soft-ware weakness," in *Proceedings of the KSII the 9th international conference on internet (ICONI) 2017 symposium*, Vientien, Laos, December 2017.

[15] R. Russell, L. Kim, L. Hamilton et al., "Automated vulnerability detection in source code using deep representation learning," in *Proceedings of the 17th IEEE international conference on machine learning and applications (ICMLA)*, pp. 757–762, IEEE, Orlando, FL, USA, December 2018.

[16] F. Wu, J. Wang, J. Liu, and W. Wang, "Vulnerability detection with deep learning," in *Proceedings of the 2017 3rd IEEE international conference on computer and communications (ICCC)*, pp. 1298–1302, IEEE, Chengdu, China, December 2017.

[17] S. Wang, T. Liu, and L. Tan, "Automatically learning semantic features for defect prediction," in *Proceedings of the 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pp. 297–308, IEEE, 2016.

[18] G. Lin, J. Zhang, W. Luo et al., "Software vulnerability discovery via learning multi-domain knowledge bases," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2469–2485, 2021.

[19] L. K. Shar and H. B. K. Tan, ""Predicting common web application vulnerabilities from input validation and sanitization code patterns," in *Proceedings of the2012 27th IEEE/ACM international conference on automated software engineering*, pp. 310–313, IEEE, Essen, Germany, September 2012.

[20] Z. Li, D. Zou, S. Xu et al., "Vuldeepecker: a deep learningbased system for vulnerability detection," 2018, https://arxiv.org/abs/1801.01681.

[21] G. Lin, J. Zhang, W. Luo, L. Pan, and Y. . Xiang, "Poster: vulnerability discovery with function representation learning from unlabeled projects," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2539–2541, Dallas, TX, USA, October 2017.

[22] G. Lin, J. Zhang, W. Luo et al., "Cross-project transfer representation learning for vulnerable function discovery," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3289–3297, 2018.

[23] M. Pradel and K. Sen, "Deep learning to find bugs," *TU Darmstadt, Department of Computer Science*, vol. 4, no. 1, 2017.

[24] J. Henkel, S. K. Lahiri, B. Liblit, and T. Reps, "Code vectors: understanding programs through embedded abstracted symbolic traces," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 163–174, Boise, ID, USA, November 2018.

[25] Y. Fang, Y. Liu, C. Huang, and L. Liu, "FastEmbed: p," *PLoS One*, vol. 15, no. 2, Article ID e0228439, 2020.

[26] A. Kanade, P. . Maniatis, G. Balakrishnan, and K. Shi, "Learning and evaluating contextual embedding of source code," in *Proceedings of the International Conference on Machine Learning*, pp. 5110–5121, PMLR, Shenzhen, China, February 2020.

[27] R.-M. Karampatsis and C. Sutton, "Scelmo: source code embeddings from language models," 2020, https://arxiv.org/abs/2004.13214.

[28] P. Zeng, G. Lin, L. Pan, Y. Tai, and J. Zhang, "SOftware vulnerability analysis and discovery using deep learning techniques: a survey," *IEEE Access*, vol. 8, pp. 197158–197172, 2020.

[29] S. K. Singh and A. Chaturvedi, "Applying deep learning for discovery and analysis of software vulnerabilities: a brief survey," *Advances in Intelligent Systems and Computing*, vol. 1154, pp. 649–658, 2020.

[30] N. Sun, J. Zhang, P. Rimba, S. Gao, L. Y. Zhang, and Y. Xiang, "Data-driven cybersecurity incident prediction: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1744–1772, 2019.

[31] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, "A survey of android malware detection with deep neural models," *ACM Computing Surveys*, vol. 53, no. 6, pp. 1–36, 2020.

[32] G. Lin, S. Wen, Q.-L. Han, J. Zhang, and Y. Xiang, "Software vulnerability detection using deep neural networks: a survey," *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1825–1848, 2020.

[33] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, https://arxiv.org/abs/1301.3781.

[34] Z. Feng, D. Guo, D. Tang et al., "Codebert: a pretrained model for programming and natural languages," 2020, https://arxiv.org/abs/2002.08155.

[35] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," in *Proceedings of the Advances in neural information processing systems*, pp. 5998–6008, Long Beach, CA, USA, December 2017.

[36] V. J. Hellendoorn, C. Sutton, R. Singh, P. Maniatis, and D. Bieber, "Global relational models of source code," in

Proceedings of the International conference on learning representations, New Orleans, Louisiana, May 2019.

[37] S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer, “Mapping language to code in programmatic context,” 2018, https://arxiv.org/abs/1808.09588.

[38] J. Zhang, L. Pan, Q.-L. Han, C. Chen, S. Wen, and Y. Xiang, “Deep learning based attack detection for cyber-physical system cybersecurity: a survey,” *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 3, pp. 377–391, 2022.

[39] C.-L. Zhang, J.-H. Luo, X.-S. Wei, and J. Wu, “In defense of fully connected layers in visual representation transfer,” in *Pacific Rim Conference on Multimedia*Springer, New York, NY, USA, 2017.

[40] FlawFinder: https://dwheeler.com/flawfinder/.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 6568602, 11 pages
https://doi.org/10.1155/2021/6568602

Research Article

TapChain: A Rule Chain Recognition Model Based on Multiple Features

Keyu Jiang,[1] Hanyi Zhang,[2] Weiting Zhang,[1] Liming Fang,[1,3,4] Chunpeng Ge,[1,3,4] Yuan Yuan,[2] and Zhe Liu[1]

[1]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211 100, Jiangsu, China*
[2]*Alibaba Group, Hangzhou 311 100, Zhejiang, China*
[3]*Nanjing University of Aeronautics and Astronautics Shenzhen Research Institute, Shenzhen 518 000, Guangdong, China*
[4]*Science and Technology on Parallel and Distributed Processing Laboratory (PDL), Changsha 410 000, Hunan, China*

Correspondence should be addressed to Liming Fang; fangliming@nuaa.edu.cn

Received 24 September 2021; Revised 27 October 2021; Accepted 18 November 2021; Published 23 December 2021

Academic Editor: Jinguang Han

Trigger-action programming (TAP) is an intelligent tool, which makes it easy for users to make intelligent rules for IoT devices and applications. Unfortunately, with the popularization of TAP and more and more rules, the rule chain from multiple rules appears gradually and brings more and more threats. Previous work pays more attention to the construction of the security model, but few people focus on how to accurately identify the rule chain from multiple rules. Inaccurate identification of rule chains will lead to the omission of rule chains with threats. This paper proposes a rule chain recognition model based on multiple features, TapChain, which can more accurately identify the rule chain without source code. We design a correction algorithm for TapChain to help us get the correct NLP analysis results. We extract 12 features from 5 aspects of the rules to make the recognition of the rule chain more accurate. According to the evaluation, compared with the previous work, the accuracy rate of TapChain is increased by 3.1%, the recall rate is increased by 1.4%, and the precision rate can reach 88.2%. More accurate identification of the rule chain can help to better implement the security policies and better balance security and availability. What's more, according to the rule chain that TapChain can recognize, there is a new kind of rule chain with threats. We give the relevant case studies in the evaluation.

1. Introduction

Trigger-action programming (TAP) [1] is a programming method that users can use to stitch devices and applications (APPs) together, such as IoT devices, Twitter, and Google Calendar. The users do not need to know the specific programming language. They just need to use the graphical interface to create rules such as "if this, then that." For example, the rule "if your room temperature is too high, then turn on your A/C" indicates that the air conditioner will be turned on when the temperature in your room is too high. TAP can be used not only in Internet of Things but also in the interaction between APPs. For example, the rule "if you share a photo on Instagram, then upload a photo from URL in your Facebook" instructs the photo will be uploaded in your Facebook when you share a photo on Instagram. This kind of personalized and simple programming model provides great convenience for end-users.

Unfortunately, as more and more devices and APPs need to be associated, this intelligent method will also bring vulnerabilities to the end-user system due to the formation of a variety of rule chains vulnerabilities. For example, one rule turns on the heater, while another rule opens the window when the temperature is too high. These two rules may cause windows to open unexpectedly and then the open windows can be exploited by attackers. However, a rule adds any new iOS reminders to your Google Calendar, and another rule posts a tweet when any new event is added to your Google Calendar. These two rules may upload reminders you do not want to upload to public space to Twitter, as shown in Figure 1.

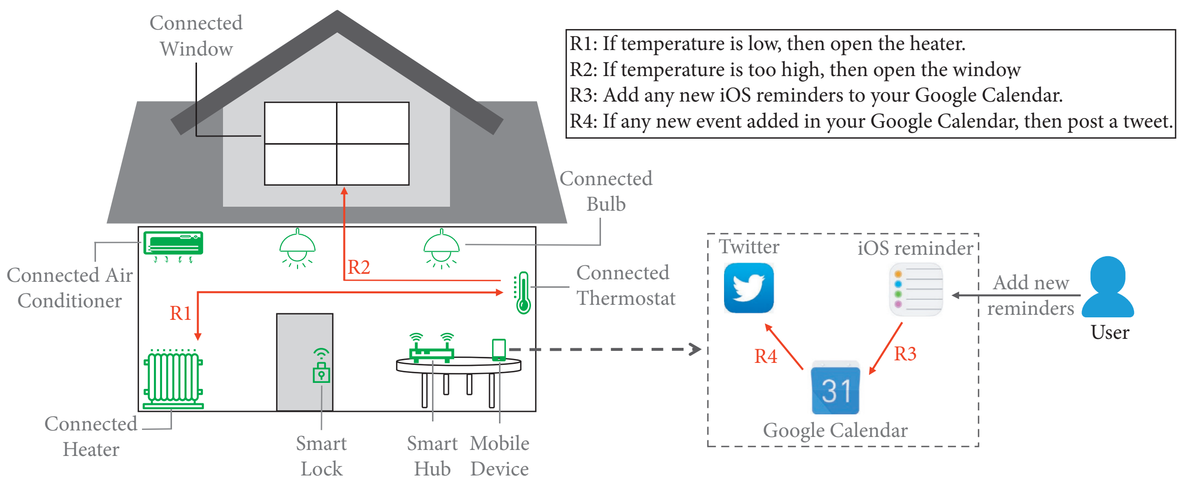

Connected Window
R1: If temperature is low, then open the heater.
R2: If temperature is too high, then open the window
R3: Add any new iOS reminders to your Google Calendar.
R4: If any new event added in your Google Calendar, then post a tweet.
Connected Bulb
Connected Air Conditioner
R2
R1
Connected Thermostat
Twitter
iOS reminder
Add new reminders
User
R4
R3
31
Google Calendar
Connected Heater
Smart Lock
Smart Hub
Mobile Device

Figure 1: Application scenario of TAP.

In previous work, researchers pay more attention to the construction of a security model to identify threats such as SOTERIA [2] but ignored the identification of the rule chain. Although the construction of the security model is important, the accurate identification of the rule chain is also crucial. Only by accurately identifying the rule chain, the excellent security model can work more perfectly. Celik et al. [3] can accurately identify the rule chain in SmartThings by using code analysis tools on the SmartThings platform. However, it is difficult to apply the closed nature of IoT platforms, such as IFTTT, on a large scale [3]. Although Wang et al. [4] can identify the rule chain without source code, its accuracy and precision are not good enough.

In this paper, we propose a rule chain recognition model based on multiple features, TapChain. We define a $\langle \text{action}, \text{trigger} \rangle$ pair as a rule chain. TapChain can improve the recognition accuracy of the rule chain through the natural language description of rules. To overcome the closed nature, we use natural language processing (NLP) technology to analyze and process trigger description and action description, respectively. Due to the marking error of the NLP tool, we design a correction algorithm to help us get the correct mark. Since there is no label data of rule chain at present, we manually combine and identify $\langle \text{action}, \text{trigger} \rangle$ pair in the same service and label 0 or 1 for each pair to indicate whether the rule chain will be formed. Then, we extract 12 features from trigger description and action description and then judge whether an action will cause a trigger to start.

We evaluate TapChain according to triggers and actions from 279,828 real rules on the IFTTT website. We find that TapChain can achieve 89.9% accuracy, 88.2% precision, and 92.1% recall in identifying rule chains. Compared with the previous work, TapChain's recognition accuracy of the rule chain is improved by 3.1% and the recall rate is improved by 1.4%. In addition, through TapChain, we find a new kind of rule chain with threat and we give a case study.

We summarize our contributions as follows:

(1) We propose TapChain, a rule chain recognition model based on multiple features. We extract 12 features to identify the rule chain. These features can more finely represent the relationship between action and trigger in $\langle \text{action}, \text{trigger} \rangle$.
(2) We design a correction algorithm for TapChain to help us get the more accurate NLP analysis results.
(3) Compared with the previous work, TapChain's recognition accuracy of the rule chain is improved by 3.1%, the recall rate is improved by 1.4%, and the precision rate can achieve 88.2%.
(4) Thanks to the improvement of the accuracy of rule chain recognition, we find a new kind of rule chain with threats. We give the relevant case study in the evaluation.

The organization of this paper is as follows. Section 2 outlines the related work. Section 3 shows the data processing. Section 4 introduces TapChain. In Section 5, we evaluate TapChain. Section 6 discusses usability and limitations. Section 7 concludes the paper.

2. Related Work

Comprehension of Single Rule in TAP. Researchers have been studying TAP for seven years. The earliest research can be traced back to 2014. To test the usability of TAP, Ur et al. [1] gathered 266 participants to carry out a user study, which laid a good foundation for the research of TAP. Since then, many researchers [5–9] have been devoted to the research in this field. In 2020, Zhao et al. [10] proposed a visual interface to interpret TAP rules to help users understand the operation of rules. These researches are of great significance for other researchers to assume that users can understand a single rule when they study the rule chain.

Security of Single Rule in TAP. In 2016, Fernandes et al. [11] researched the potential security problems caused by TAP on the SmartThings platform for the first time and found the vulnerabilities in the event subsystem of SmartThings through static analysis. Then, some researches [12–16] summarized the security problems of a single rule and put forward corresponding solutions. The security of a single rule is also helpful to the security of the rule chain.

Security of Rule Chain in TAP. On the SmartThings platform, Celik et al. [2, 3] identified the threats in TAP and proposed corresponding solutions. However, their solution needs to obtain source code, which is difficult for the IFTTT platform, because the IFTTT platform is a closed source platform. Wang et al. [4] proposed iRule, which not only constructs the security model but also realizes the identification of the rule chain without obtaining the source code. However, the performance of iRule in the chain of recognition rules can be optimized. They only extracted the features of the predicate and object from the description. It is difficult to make an accurate judgment when identifying the rule chain which has a great difference between predicate and object, especially when it involves environmental factors. Our work is to improve the recognition rule chain to better identify more threat rule chains.

3. Data Processing

In this section, we describe how we process data. Then, we introduce how we label the rule chain.

3.1. Dataset. Our work is based on the rules on the IFTTT website. So, first we need to obtain the data. We try to use Blase et al.'s method [6] to obtain the latest rules. However, at present, the links to the rules on the IFTTT website have been processed additionally. Although the URL of each rule is still "https://ifttt.com/applets/ID," the ID of each rule is no longer a number, but a combination of numbers and letters, such as "https://ifttt.com/applets/SdbAq2ce." At present, it is difficult for us to find crack methods from the ID of rules. Therefore, we have to use the dataset provided by Mi [17]. The dataset contains rules, triggers, and actions.

3.2. Data Preprocessing. It seems that we obtain a lot of rules, but there is little practical help for us to identify the rule chain. This is because each rule is composed of a trigger and an action. When users make rules, they can combine trigger and action in a variety of ways. But in the end, trigger and action come from the trigger dataset and action dataset, respectively. Therefore, we can decompose the rules into trigger and action, analyze them respectively, and then find the rule chain. Besides, the dataset contains a lot of invalid information, so we need to clean the data. For example, in the triggers dataset, "triggerUrl," "triggerchannelURL," and so on, are invalid information. So, we need to filter out this information.

3.2.1. Trigger Dataset. The trigger dataset is used to determine the detail of each trigger for comparison with action. We filter the trigger to keep only the information we need. There is the following example:

(1) **triggerTitle**: door opened
(2) **triggerDesc**: this trigger fires when the selected door is opened
(3) **triggerChannelName**: abode
(4) **triggerFieldList**: ["Select door"]

Among them, "triggerTitle" and "triggerChannelName" can uniquely determine a trigger. "TriggerDesc" describes the detail of the trigger. "TriggerFieldList" is supplementary to "triggerDesc" and it is a personalized value that can be set by users. In this example, we can use the "triggerFieldList" to determine which door the trigger exactly corresponds to. "TriggerFieldList" is also extracted as features.

3.2.2. Action Dataset. The action dataset is used to determine the function and impact of each action for comparison with a trigger. We also sift out the information we need. There is the following example:

(1) **actionTitle**: change mode
(2) **actionDesc**: this action will change your abode system to the selected system mode
(3) **actionChannelName**: abode
(4) **actionFieldList**: ["Which mode?"]

"ActionTitle" and "actionChannelName" can uniquely determine an action. "ActionDesc" describes the function and influence of the action in detail. "ActionFieldList" is supplementary to "actionDesc" and it is a personalized value which can be set by users. In this example, we can use the "actionFieldList" to determine which mode the action will change to.

3.3. Description Processing. First of all, at the beginning of the description of each trigger, there are expressions such as "this trigger fires when." Similarly, at the beginning of the description of each action, there are expressions such as "this action will." This kind of information is not only helpful for us to analyze the rule chain but also increases the analysis complexity of the description statement. So, we delete these prefixes and capitalize the first letter of the new first word to make it a sentence.

Second, we find that some trigger descriptions have multiple sentences. For example, the description "this trigger fires every time anyone shares a public photo with a tag you specify. Note: limited to 30 photos per check" contains two sentences. We notice that the first sentence of the description mainly describes the function of the trigger. The rest of the sentence is usually a prompt to the user. So, we only keep the first sentence of the description.

Third, in the description, some special symbols appear, such as % and /. For %, it often appears in statements describing percentage, such as humidity level and lighting level. We convert it into a word, percentage, to avoid errors in NLP tool analysis. For /, it often appears in the abbreviation of air conditioning. We find that A/C often appears in the description instead of air conditioning. We directly delete these /, to avoid the NLP tool taking A/C apart for analysis.

3.4. Dataset Labeling. When we look for the rule chain, we must find which trigger can be affected by the action. We define the pair as $\langle \text{action}, \text{trigger} \rangle$. If an action can start a trigger in $\langle \text{action}, \text{trigger} \rangle$, we label it as 1; otherwise, we label it as 0.

For the convenience of describing our method, we give some examples of action and trigger, as shown in Table 1. ID is used to distinguish each data represents action or trigger, A represents action, and T represents trigger. The description is the description of the action or trigger.

We divide $\langle \text{action}, \text{trigger} \rangle$ labeled as 1 into two parts. One is that action can start trigger directly, which is called direct chain, and the other is that action can start trigger by affecting the physical environment, which is called indirect chain. For the two types of data, we describe them separately.

3.4.1. Direct Chain. We define that action can only start triggers belonging to the same services. When we label, comparing the description of action and trigger, we can judge whether most $\langle \text{action}, \text{trigger} \rangle$ can form a direct chain. For example, A1 and T1 in Table 1; we can directly judge that A1 can start T1. First of all, A1 and T1 belong to the same service. Secondly, according to the description of T1, as long as the mode of abode system changes, it can be triggered. A1 is to change the mode of abode system. Finally, we need to determine whether the values in "FieldList" are the same. Therefore, we can label $\langle A1, T1 \rangle$ as 1 if the mode of action is the same as that of the trigger; otherwise, label 0.

3.4.2. Indirect Chain. In contrast, labeling the indirect chain is more difficult. We first need to define the environment for the device. For example, the operation of air conditioning and heaters can affect both temperature and humidity. We have made a summary of the devices appearing in the triggers dataset and actions dataset, as shown in Table 2.

The second difficulty is that we need to distinguish how devices affect the environment. For example, air conditioning has many functions, such as refrigeration and heating. Refrigeration will reduce the temperature and humidity, but heating will increase the temperature and reduce the humidity. "FieldList" can help us.

For example, A2 describes that this action will turn on your A/C in the specified room and run it in temperature mode. From the description, it is difficult to determine the impact of air conditioning on the environment. However, the "keep temperature at" in "actionFieldList" clearly indicates that the function of A2 is to keep the temperature at a certain value. Therefore, we can know whether A2 can start T3 by comparing the "temperature threshold" in A2 "actionFieldList" with the "keep temperature at" in T3 "triggerFieldList." If the value in the "actionFieldList" of A2 is less than the value of the "triggerFieldList" of T3, we can label $\langle A2, T3 \rangle$ as 1; otherwise, we label 0. At the same time, since A2 turns on the air conditioner, T2 may be started. If the first value in T2 "triggerFieldList" is "turn on" and the second value is the same as the first value in A2 "actionFieldList," we can label $\langle A2, T2 \rangle$ as 1; otherwise, we label 0.

Based on the above analysis, the personalized values, values in the "actionFieldList" and "triggerFieldList" are very important for us to judge whether a rule chain can be formed. So, we randomly set these values to complete labeling.

4. TapChain

In this section, we describe TapChain, our model to recognize the rule chain. Because the IFTTT platform is a closed source platform, program analysis technology is difficult to apply to it [4]. To overcome this difficulty, TapChain recognizes the rule chain by using natural language processing (NLP).

4.1. Overview. TapChain first takes the data processed in Section 3 as the input. These data are divided into two categories. One is "Desc" data, including "triggerDesc" and "actionDesc." The other is "FieldList" data, including "triggerFieldList" and "actionFieldList." For "desc," we use the NLP tool, Stanford CoreNLP [18], to perform part-of-speech (POS) tagging and dependency parsing on sentences. When the tool is used for sentence analysis, some special sentences are analyzed inaccurately, so we correct these inaccurate analysis results. Then, we extract semantic related features from the corrected results. For "FieldList," the data inside are personalized, and users can set it by themselves. To overcome the uncertainty brought by personalized data, we abstract the data into the comparison of the same value or the same range. Then, we extract the "FieldList" features. We also extract environment-related features based on "FieldList" and "Desc." Finally, we input the extracted features into the machine learning models to obtain the recognition results of the rule chain. The overview is shown in Figure 2.

4.2. Description Analysis Based on NLP. We first perform part-of-speech (POS) tagging and dependency parsing using Stanford CoreNLP. The generated dependency tree is shown in Figure 3.

The dependency relationship analyzed by Stanford CoreNLP is complex. There are some invalid elements, such as the analysis of stop words and punctuation. So, we summarize the syntax elements that we focus on, as shown in Table 3.

(1) Predicate: it is a verb, expressed as root. Sometimes, there is a verb phrase. At this time, we need to combine the word marked as root with the word marked as compound:prt as verb, such as "turn on." When conjunctions appear in a sentence, there will be coordinate verbs, expressed as conj.

(2) Direct object: the action is executed on it. The direct object is expressed as obj. When there is no word marked obj in the sentence, we take the word marked obl as the direct object, because indirect objects cannot exist independently. We also define the subject in the passive voice as the direct object, expressed as nsubj:pass.

(3) Indirect object: the indirect object is expressed as obl, such as "mode" in "change your abode system to the selected system mode."

(4) Modifier: modify a noun. In our work, most of them modify the object, such as "new" in "write a new message in a given sphere."

TABLE 1: Example of action and trigger.

ID	Service	Description	FieldList
A1	Abode	Change your abode system to the selected system mode	["Which mode"]
A2	Ambi climate	Turn on your A/C in the specified room, and run it in temperature mode	["Keep temperature at," "Which A/C"]
T1	Abode	Your abode system mode is changed	["Select mode"]
T2	Ambi climate	The air conditioner has been turned on or off	["Turn on/off," "Select A/C"]
T3	Ambi climate	Your device detects the temperature is below the threshold you specified	["Select A/C," "Temperature threshold"]

TABLE 2: Physical environment.

Physical environment	Source of trigger	Device or service of action	Keyword
Temperature	Temperature rises above	Air conditioner, heat pump	Temperature
Humidity	Humidity rises above	Air conditioner, heat pump, humidifier	Humidity
Brightness	Brightness higher than light dimmed to specific level	Dimmer, light	Brightness, light, dim
Air flow	Zone air flow rises above threshold	iZone	Air flow, airflow
Air quality	Air quality changes	Air purifier	Air quality
Sound	Sound detected says a specific phrase	Musaic, Nightingale, Woopla, GO, Android device, D-Link Siren	Sound, say, ask, play
Motion	Motion detected	Samsung robot vacuum	Motion, robot
Smoke	Smoke detected	\	Smoke

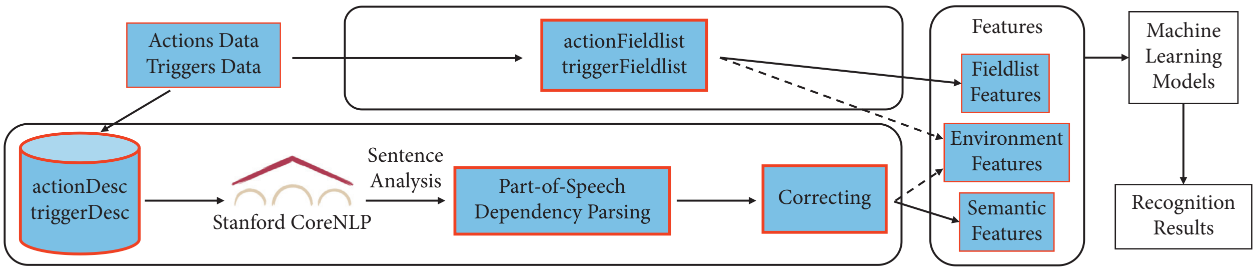


FIGURE 2: Overview of TapChain.

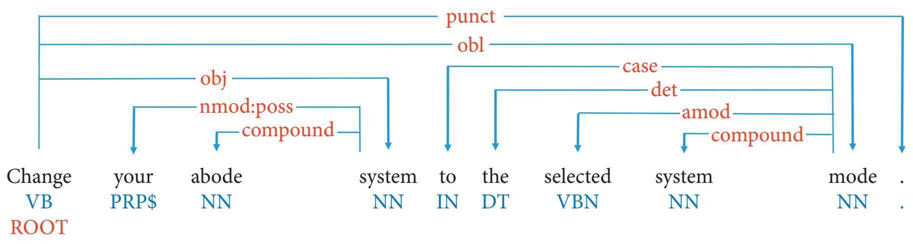


FIGURE 3: Example of dependency parsing.

TABLE 3: Explanation of the analysis results.

Sentence constituent	Analysis of Stanford CoreNLP
Predicate	Root, compound:prt, conj
Direct object	obj, obl, nsubj:pass
Indirect object	obl
Modifier	amod, compound

We do not extract the subject. Because we have deleted the beginning of the sentence in the text preprocessing, most of the sentences have become imperative sentences, and there is no subject. The valid subject only exists in the description of some triggers but, it does not exist in the description of actions.

4.3. Correcting. The Stanford CoreNLP sometimes makes analysis an error when analyzing sentences. We call this error predicate dislocation. For example, for the analysis of the sentence "add mixes to your favorite mixes," the tool will mark "add" as the subject and "mixes" as the predicate, as shown in Figure 4. However, "add" should be the predicate and "mixes" should be the direct object.

This error has caused great obstacles to rule chain recognition, so we designed an algorithm to correct it. According to the observation, NLP tools only mark predicate and the direct object incorrectly but mark the indirect object and other sentence components starting from the indirect object correctly. Therefore, we modify the predicate and direct object and their relationship with other sentence elements. The correction algorithm is shown in Algorithm 1.

The algorithm first determines whether the syntax tree is empty and whether there is a subject. Then, the position of the subject and the part of speech of the subject are obtained in turn. If the part of speech of the subject is the original form of the verb and the position of the subject is at the beginning of the sentence, we can judge that it is a predicate dislocation. If the part of speech of our root word is the plural of verbs, it is further determined that the plural of nouns is wrongly judged as the third person singular. We update the part of speech of the root node to NNS (noun plural). Then, we swap the root node and the subject node and mark their relationship as a direct object. The corrected marking results are shown in Figure 5. Because there is only such predicate dislocation in the current dataset, our algorithm does not discuss other cases. In the future, we will update the algorithm if new marking errors appear.

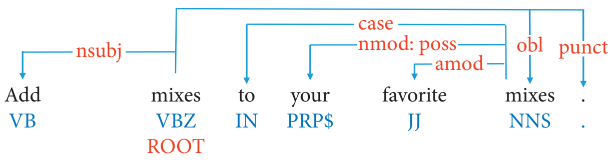


Figure 4: Wrong mark.

4.4. Feature Extraction. Our goal is to determine whether an action can start a trigger. Wang et al. [4] only extracted features from verbs and objects, and their effect can be further improved. Therefore, we extract more features to help us identify the rule chain.

First of all, we use the lemma function of Stanford CoreNLP to transform the predicate and object words into word prototypes for feature extraction. Then, we extract features from five different aspects.

4.4.1. Predicate. Usually, the predicate is the core of a sentence; it has the function of connecting the preceding and the following. Through the predicate, we can know how to complete a task.

Same Predicate. The role of the predicate in a sentence determines its importance. Therefore, in a ⟨action, trigger⟩, we compare the predicate in action with the predicate in the trigger and construct a binary feature according to whether it is the same or not. In the description of trigger and action, there are parallel predicates. In this case, when we compare predicates, as long as one predicate is the same, we think that the predicate of action is the same as that of the trigger. We use the f(s_p) to represent this feature, a_ps to represent all predicates in an action, and t_ps to represent all predicates in a trigger. We use a_p to represent predicate from a_ps and use t_p to represent predicate from t_ps. We abstract the feature extraction as a formal expression, as follows:

$$f(s\_p) = \begin{cases} 1, & \text{if } \exists a\_p \in a\_ps = \exists t\_p \in t\_ps, \\ 0, & \text{else.} \end{cases} \tag{1}$$

4.4.2. Object. Generally speaking, the object is the receiver of the predicate. Different objects, even if they have the same predicate, have opposite meanings. Therefore, it is equally important for us to identify the rule chain.

Same Direct Object. The direct object is the receiver of the predicate. We take it as a binary feature whether the direct object in action is the same as that in the trigger.

Same Indirect Object. The indirect object is a part affected. We also take it as a binary feature.

4.4.3. Modifier. Modifiers are used to modify nouns, which can describe some special properties of the modified nouns. Since we have extracted two types of objects, we also divide attributives into two types.

Same Modifier of Direct Object. Modifier of the direct object is used to modify the direct object. We take it as a binary feature.

Same Modifier of Indirect Object. Modifier of the indirect object is used to modify the indirect object. We also take it as a binary feature.

4.4.4. FieldList. On the IFTTT website, action and trigger have their own "FieldList," which are "actionFieldList" and "triggerFieldList," respectively. In "FieldList," there is a lot of specific information. Therefore, we also extract three features from "actionFieldList" and "triggerFieldList."

Exist Same Fields. We first traverse the "FieldList" of action and trigger respectively to find out whether there are the same fields. We take the number of same fields as a continuous feature.

Same Values in Same Fields. We also need to know if the values are the same in all the same fields. We take it as a binary feature.

The Number of Same Values. This is a supplement to the second feature. We calculate the number of the same personalized values. We take it as a continuous feature.

4.4.5. Environment. We first extract physical environmental factors from triggers. This is because our goal is to identify the rule chain. Only when the trigger can receive the physical environmental factors can the rule chain be formed. When we label manually, we do this work at the same time. We

```
Input: $R$ (root of syntax tree)
Output: $R'$ (root of corrected syntax tree)
(1) function FixPredicate($R$)
(2)   if ($R$ == NULL) and (getSubjct($R$) == NULL) then
(3)     return $R$
(4)   end if
(5)   subj_pos ⟵ getSubjectPosition($R$)
(6)   subj_PoS ⟵ getPartOfSpeech(subj_pos)
(7)   if (subj_PoS == VB)and(subj_pos == 1) then
(8)     root_PoS ⟵ getPartOfSpeech($R$)
(9)     if root_PoS == VBZ then
(10)    subjNode ⟵ getSubjectNode($R$)
(11)    root_pos ⟵ getRootPosition($R$)
(12)    root_PoS ⟵ NNS
(13)    SwapNode($R$, subjNode)
(14)    $R'$ ⟵ subjNode
(15)    $R'$ ⟵setRelation($R'$, $R$, $obj$)
(16)    end if
(17)  end if
(18)  return$R'$
(19) end function
```

Algorithm 1: Correcting predicate dislocation.

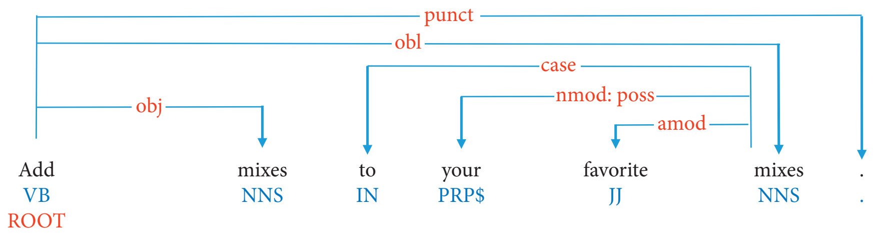


Figure 5: Corrected mark.

extract eight kinds of physical environment factors, and at the same time, we extract keywords to extract features, as shown in Table 2.

For the physical environment, we can distinguish whether the indirect chain can be formed according to four characteristics.

Same Environmental Factor. First of all, the physical environment affected by an action needs to be the same as that detected by a trigger before the action can start the trigger. Therefore, we regard whether the physical environment affected by action is the same as the physical environment detected by trigger as the first physical environment feature, which is a binary feature. For the first three environmental factors, we extract them directly from the "actionFieldList" and "triggerFieldList." But we cannot extract the hidden environmental factors of device effect and other environmental factors in the same way. Therefore, for a trigger, we extract it from "triggerDesc" by string matching and, for action, we summarize the environment affected by the device or service and the keywords to extract features in Table 2. According to the terms of the device or service and the keywords we extracted, we judge whether the action will cause environmental changes.

The Types of Environmental Factors. Second, if the environmental factors are the same, we extract the types of environmental factors. We take it as a continuous feature. The values of this feature are from 1 to 8, representing 8 environmental factors in Table 2. 0 is used to indicate that the environment variables are different.

Same Value Range. Third, we need to judge whether the values of the physical environment are the same. When "higher, above, exceed, greater" appear in "triggerDesc," we map the environment value range of trigger (T(E_v_r)) to $[T(E\_v), \infty)$. When "lower, below, less" appear in "triggerDesc," we map the environment value range of trigger (T(E_v_r)) to $(-\infty, T(E\_v)]$. Then, we can get values from the "actionFieldList" (A(E_v)) and "triggerFieldList" (T(E_v)) and compare them. This is important in the first three environmental factors: temperature, humidity, and brightness because they all have certain values in trigger and action. We also define this feature as a binary feature. We use f(s_v_r) to represent this feature. For the convenience of understanding, we abstract it as a formal expression, as follows:

$$T(E\_v\_r) = \begin{cases} [T(E\_v), \infty) & \text{higher, above,} \\ (-\infty, T(E\_v)], & \text{lower, below,} \end{cases}$$
$$f(s\_v\_r) = \begin{cases} 0, & A(E\_v) \notin T(E\_v\_r), \\ 1, & A(E\_v) \in T(E\_v\_r). \end{cases} \tag{2}$$

Same Trend. Finally, we also make it a feature whether the changing trend of the physical environment is the same. For other environmental factors, there may be an explicit numerical expression in the trigger, but we cannot find the description of numerical value in action. Besides, some devices will affect other physical environments. For example, we can get the temperature value of the air conditioner setting through the "actionFieldList," but we cannot get the value that the air conditioner affects the humidity; we can only judge that it will lower the humidity. Therefore, we take it as a supplement to the second physical environmental feature. We define this feature as a binary feature. For the first five environmental factors, we define the trend as up or down according to the description. For the latter three environmental factors, we only define whether they exist.

4.4.6. Summary. We extract 9 binary features and 3 continuous features from five different perspectives. Among them, the five features of grammatical elements are based on NLP technology. The other features are more extracted by string matching and numerical comparison.

5. Evaluation

In this section, we evaluate the performance of TapChain. In the following evaluation, we use i7-5700HQ@2.70GHz quad-core processors, 8 GB memory.

5.1. Setup. We obtain 279,828 rules, 431 services, 1,470 triggers, and 896 actions. In the above data, only 150 services have both triggers and actions, so we only construct 5,528 ⟨action, trigger⟩ pairs with the same service and label them. It takes us about 60 hours to complete the task of labeling data. One of our collaborators reviews the labeled dataset and revises it. In the process of labeling data, we find that the positive and negative data are unbalanced, and the positive data only accounted for about one-tenth of all datasets. We do not have a large number of data, so we use SMOTE algorithm in imblearn [19] to balance the data. We randomly select the data of 80 services as the training set of the classifier, and the data of 70 services as the test set of the classifier.

In the choice of classifier, we do not consider too much. Since our task is a supervised binary classification task, we use four main supervised learning classifiers from scikit-learn [20] to train the model. The four classifiers are random forest, support vector machine (SVM) (RBF kernel), multiple perceptron, and logistic regression.

5.2. Performance. We use test sets to evaluate TapChain with accuracy, precision, recall, and F1-score [21]. In our opinion, precision rate and recall rate are equally important in the recognition rule chain. Because, if the precision of the model is low, the future security policy based on the model will confuse users and seriously affect the user experience. If the recall rate is low, the identified rule chain will be incomplete, which will affect the security of users. Therefore, besides accuracy rate, precision rate, and recall rate, we use F1-score to evaluate TapChain. The results are shown in Table 4.

All evaluation indicators are calculated based on the following concepts. True positive (TP) is the number of rule chains correctly predicted. False positive (FP) is the number of rule chains incorrectly predicted. True negative (TN) is the number of correct recognition that will not form a regular chain. False negative (FN) is the number of incorrect recognition that will not form a regular chain. All is the number of all data. The accuracy rate (A), precision rate (P), recall rate (R), and F1-score (F1) are defined as follows:

$$\begin{cases} A = \dfrac{\text{TP} + \text{TN}}{\text{All}}, \\ P = \dfrac{\text{TP}}{\text{TP} + \text{FP}}, \\ R = \dfrac{\text{TP}}{\text{TP} + \text{FN}}, \\ F1 = \dfrac{2 \times P \times R}{P + R}. \end{cases} \tag{3}$$

According to the results, our model of identifying rule chain is better than that of Wang et al. [4]. TapChain performs better when using the same machine learning model. The reason why TapChain works better is that we extract more features and correct the results of NLP tool analysis. We not only extract additional semantic features from indirect objects but also extract additional environment and personalized features from "FieldList." For the analysis error of the NLP tool, we design an algorithm to correct it, which can get the correct part-of-speech tagging and dependency parsing. Then, we can extract the correct features and improve the performance of TapChain. In addition, we split the sentences with conjunctions that are easy to cause trouble. When two sentences are compared and there is a same predicate, we think that the predicates of the two sentences are the same. It also improves the performance of TapChain.

Besides, they do not use the personalized data in "FieldList" when they label datasets, and in practical application, TapChain has more advantages in identifying rule chains. Because when users use these actions and triggers, they will set specific values. For example, for action "turn on light," the user needs to set "which light" in "actionFieldList." At this time, if there is a trigger "light turned on," and the "which light" in the "triggerFieldList" is different from the "which light" set in the "actionFieldList," then the action will not start the trigger.

To prove the rationality of our conjecture, we calculate the importance of every feature. Because the model based on

Table 4: Results (percentage).

	Model	Accuracy	Precision	Recall	F1-score
TapChain	SVM (RBF kernel)	89.7	87.8	92.1	89.9
	Random forest	89.5	87.9	91.7	89.7
	Multiple perceptron	88.1	87.8	88.5	88.1
	Logistic regression	89.9	88.2	92.1	90.1
Wang et al. [4]	SVM (RBF kernel)	80.2	—	90.7	—
	Random forest	85.7	—	88.2	—
	Multiple perceptron	86.8	—	88.6	—
	Logistic regression	83.1	—	84.4	—

logistic regression performs better, we calculate the importance of features according to logistic regression. In logistic regression, the importance of features is strongly correlated with the coefficients of features. Therefore, we reflect the importance of features through the coefficients of features. We can understand this more easily through the formal expression of logical regression. The formal expression of logistic regression is as follows:

$$y = \frac{1}{1 + e^{\left(-\omega^{T} x\right)}}, \tag{4}$$

where x is a vector composed of different features, and ω is a vector composed of coefficients of different features. The solution process of ω is as follows:

$$\omega_{t+1} = \omega_t + \eta\left(y_n - \frac{1}{1 + e^{\left(-\omega^{T} x_n\right)}}\right)x_n, \tag{5}$$

where η is a constant, ω_0 is a random vector, and x_n and y_n are from a random training sample. Because ω is updated continuously, and finally an appropriate weight is calculated for every feature, the coefficient of every feature can represent the importance of the feature.

In addition, since scikit-learn officially provides the calculation method of feature importance in the random forest model, we also calculate the importance of each feature in the random forest, as shown in Figure 6.

To facilitate drawing, we use the abbreviation of feature as the coordinate axis. The relationship between abbreviations and features is shown in Table 5. For the feature importance of random forest, the closer the value is to 1, the more important the feature is. For logistic regression, the larger the absolute value of the coefficient of a feature, the more important the feature is. According to Figure 6, we find that the most important features come from predicates and direct objects, followed by "FieldList"-related features and environment-related features. The features extracted from indirect objects play a very limited role in the classifier. Our newly extracted features play an important role in the accurate recognition of rule chains.

5.3. Case Study. Because we extract more features, TapChain can identify the rule chains more accurately and more fine-grained. We find that there are some new threatening rule chains that cannot be recognized by the existing work. We give a relevant case study.

(i) R1: if a trigger, then play audio from URL (Musaic).

(ii) R2: if say a simple phrase (Google Assistant), then open the garage door you specify (Gogogate).

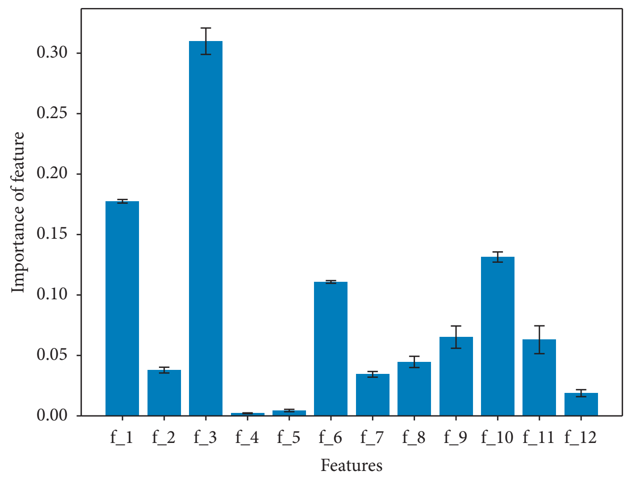
R1 is a virtual rule, while the action of R1 is real action and R2 is a real rule. However, the action of R1 is real action, and if this action is used, there will be a huge threat from this rule chain. An attacker can make a high-frequency signal with "open the garage door" coding and then insert the high-frequency signal into the audio file. Finally, the attacker uploads the audio file to the URL. This URL can be a song list subscribed by the victim or a public link used by the victim. When the victim starts R1, the audio file uploaded by the attacker will be played. Because the current sound receiving equipment cannot filter the high-frequency signal [22], Google Assistant will receive the high-frequency signal, "open the garage door," from the audio. This causes R2 to be activated unexpectedly. Then, the attacker can commit further criminal acts, as shown in Figure 7.

Generally speaking, if an attacker wants to take advantage of the domino effect of the rule chain, they need to use a malicious rule to start the rule chain [4]. However, the existence of this rule chain makes the attacker does not even need to construct a malicious rule, and the attacker can directly use the rule chain.

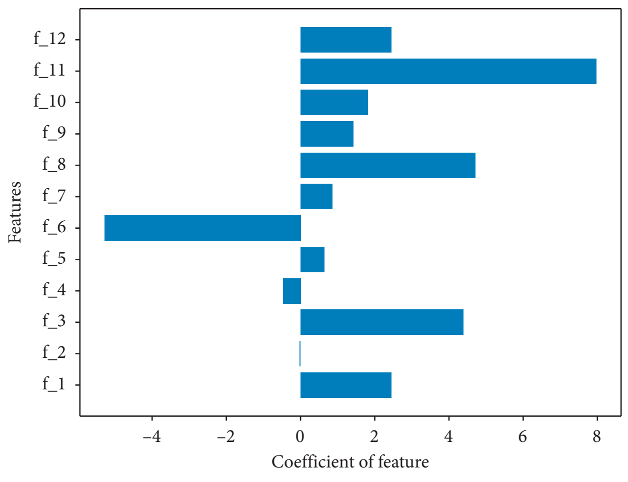
In this rule chain, none of the predicate and object of R1's action and R2's trigger are the same, which can only be identified according to the features of environmental factors that we extracted. Previous work cannot recognize this kind of rule chain. However, it is very dangerous for users to construct this kind of rule chain unintentionally when using TAP.

6. Discussion

Usability. Our goal is to better identify the rule chain and then more accurately identify threatening rule chains for end-users. When users use TAP to automate devices or APPs, TapChain can help users identify rule chains. Users can decide whether to change the TAP rules according to their own needs. Because the recognition of the rule chain by TapChain is based on the description of the text, TapChain can be effective as long as the Internet of things platform or TAP platform has a complete description. In future work, we plan to use a user survey to obtain real datasets to evaluate and modify TapChain. In future work, an important aspect will be the construction of a security model. In this way, TapChain can directly present the threatened rule chain to users.



(a)



(b)

FIGURE 6: Feature importance. (a) Random forest. (b) Logistic regression.

TABLE 5: The abbreviation of feature.

Abbreviation	Feature
f_1	Same predicate
f_2	Same modifier of direct object
f_3	Same direct object
f_4	Same modifier of indirect object
f_5	Same indirect object
f_6	Same field
f_7	Same value in same field
f_8	The number of same value
f_9	Same environmental factor
f_10	The type of environmental factor
f_11	Same value range
f_12	Same trend

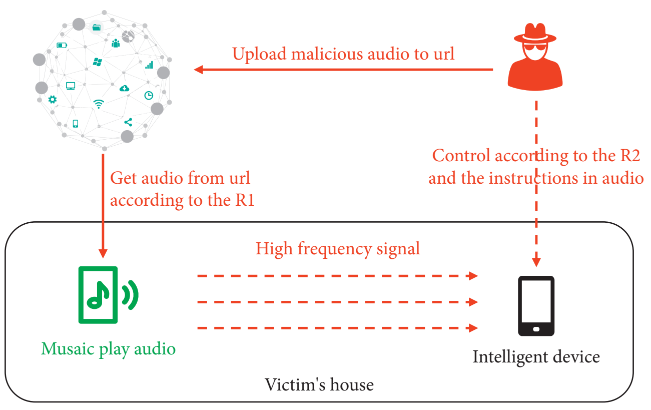

Upload malicious audio to url
Get audio from url according to the R1
Control according to the R2 and the instructions in audio
High frequency signal
Musaic play audio
Intelligent device
Victim's house

FIGURE 7: New potential threat.

Limitation. There are also some limitations to our work. First of all, manually labeling a dataset can lead to potential errors. Although we avoid errors by labeling and reviewing, manually labeling datasets can still be the source of the errors. Secondly, TapChain is difficult to identify some rule chains with time constraints. For example, one rule is to turn on the washing machine, and another rule sends a message to me when the washing machine is turned off. Generally speaking, the washing machine stops automatically after running for a while. When the washing machine stops, a message will be sent. However, since TapChain cannot obtain the running time, it cannot accurately identify such rule chains. Finally, because TapChain's recognition of the rule chain is based on the description of the text, the performance of TapChain depends on the correctness of the text description. If there is a trigger or action description that does not conform to the actual behavior, TapChain will make a judgment error.

7. Conclusion

In this paper, we propose TapChain, a rule chain recognition model based on multiple features. We extract 12 features from 5 different aspects. We designed a correction algorithm to obtain more accurate NLP analysis results and make us extract features more accurately. Our evaluation shows that the recognition accuracy of TapChain is up to 89.9%, the precision is up to 88.2%, and the recall is up to 92.1%. Compared with the existing work, the accuracy rate is increased by 3.1%, and the recall rate is increased by 1.4%. In addition, TapChain's recognition of the rule chain is more fine-grained and we find a new kind of rule chain with a potential threat, which will cause serious consequences once exploited by attackers.

Data Availability

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

Conflicts of Interest

The authors declare that the funding in Acknowledgements section did not lead to any conflicts of interest regarding the publication of this manuscript. Also, there are no any other conflicts of interest in the manuscript.

Acknowledgments

This work was support by the National Key R&D Program of China (Grant Nos. 2021YFB3100700 and 2020AAA0107703), the National Natural Science Foundation of China (Grant Nos. 61 872 181, 62 032 025, 62 076 125, 62 071 222, U20A201092, U20B2049, U20B2050, and 61 702 236), the Natural Science Foundation of Jiangsu Province (Grant Nos. BK20200418 and BE2020106), the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2021A1515012650), and the Shenzhen Science and Technology Program (Grant No. JCYJ20210324134810028).

References

[1] B. Ur, E. McManus, M. P. H. Yong, and M. L. Littman, "Practical trigger-action programming in the smart home," in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 803–812, Toronto Ontario, Canada, April 2014.

[2] Z. B. Celik, P. McDaniel, and G. Tan, "Soteria: automated iot safety and security analysis," in *Proceedings of the 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, pp. 147–158, Boston, MA, USA, July 2018.

[3] Z. B. Celik, G. Tan, and P. D. McDaniel, "Iotguard: dynamic enforcement of security and safety policy in commodity iot," in *Proceedings of the Network and Distributed System Security Symposium*, February 2019.

[4] Q. Wang, P. Datta, W. Yang, S. Liu, A. Bates, and C. A. Gunter, "Charting the attack surface of trigger-action iot platforms," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pp. 1439–1453, London, United Kingdom, November 2019.

[5] J. Huang and M. Cakmak, "Supporting mental model accuracy in trigger-action programming," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 215–225, Osaka, Japan, September 2015.

[6] B. Ur, M. P. H. Yong, S. Brawner et al., "Trigger-action programming in the wild: an analysis of 200,000 ifttt recipes," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 3227–3231, San Jose California, USA, May 2016.

[7] C. Nandi and M. D. Ernst, "Automatic trigger generation for rule-based smart homes," in *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security*, pp. 97–102, Vienna, Austria, October 2016.

[8] S. Yarosh and P. Zave, "Locked or not? mental models of iot feature interaction," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 2993–2997, Denver Colorado, USA, May 2017.

[9] W. Brackenbury, A. Deora, J. Ritchey et al., "How users interpret bugs in trigger-action programming," in *Proceedings of the 2019 CHI conference on human factors in computing systems*, pp. 1–12, Glasgow Scotland, Uk, May 2019.

[10] V. Zhao, L. Zhang, B. Wang, S. Lu, and B. Ur, "Visualizing differences to improve end-user understanding of trigger-action programs," in *Proceedings of the Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–10, Honolulu HI, USA, April 2020.

[11] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Proceedings of the 2016 IEEE symposium on security and privacy (SP)*, pp. 636–654, IEEE, San Jose, CA, USA, May 2016.

[12] M. Surbatovich, J. Aljuraidan, L. Bauer, A. Das, and L. Jia, "Some recipes can do more than spoil your appetite: analyzing the security and privacy risks of ifttt recipes," in *Proceedings of the 26th International Conference on World Wide Web*, pp. 1501–1510, Perth, Australia, April 2017.

[13] Y. J. Jia, Q. A. Chen, S. Wang et al., "Contexlot: towards providing contextual integrity to appified iot platforms," in *Proceedings of the 4th Annual Network and Distributed System Security Symposium*, vol. 2, NDSS, San Diego, California, USA, March 2017.

[14] I. Bastys, M. Balliu, and A. Sabelfeld, "If this then what? controlling flows in iot apps," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pp. 1102–1119, Toronto, Canada, October 2018.

[15] 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.

[16] L. Zhang, W. He, J. Martinez, N. Brackenbury, S. Lu, and B. Ur, "Autotap: synthesizing and repairing trigger-action programs using ltl properties," in *Proceedings of the 2019 IEEE/ACM 41st international conference on software engineering (ICSE)*, pp. 281–291, IEEE, Montreal, QC, Canada, May 2019.

[17] X. Mi, F. Qian, Y. Zhang, and X. Wang, "An empirical characterization of ifttt: ecosystem, usage, and performance," in *Proceedings of the 2017 Internet Measurement Conference*, pp. 398–404, London United, Kingdom, November 2017.

[18] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of the 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55–60, Baltimore, Maryland, January 2014.

[19] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 559–563, 2017.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[21] Z. H. Zhou, *Machine Learning*, Springer, Berlin, Germany, 2021.

[22] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: inaudible voice commands," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 103–117, Dallas Texas, USA, October 2017.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 2713211, 16 pages
https://doi.org/10.1155/2021/2713211

Research Article

Identifying IoT Devices Based on Spatial and Temporal Features from Network Traffic

Feihong Yin,[1] Li Yang,[1] Jianfeng Ma,[2] Yasheng Zhou,[1] Yuchen Wang,[1] and Jiahao Dai[3]

[1]*School of Computer Science and Technology, Xidian University, Xi'an 710071, China*
[2]*School of Cyber Engineering, Xidian University, Xi'an 710071, China*
[3]*Science and Technology on Communication Information Security Control Laboratory, Jiaxing 314000, China*

Correspondence should be addressed to Li Yang; yangli@xidian.edu.cn

Received 31 August 2021; Revised 13 October 2021; Accepted 29 October 2021; Published 25 November 2021

Academic Editor: Jinguang Han

With the rapid growth of the Internet of Things (IoT) devices, security risks have also arisen. The preidentification of IoT devices connected to the network can help administrators to set corresponding security policies according to the functionality and heterogeneity of the devices. However, the existing methods are based on manually extracted features and prior knowledge to identify the IoT devices, which increases the difficulty of the device identification task and reduces the timeliness. In this paper, we present CBBI, a novel IoT device identification approach. On the one hand, CBBI uses a hybrid neural network model Conv-BiLSTM to automatically learn the representative spatial and temporal features from the network traffic, such as the position relationship of the internal organization structure in network communication traffic, the time sequence of the data packets, and the duration of the network flow. On the other hand, CBBI contains the data augmentation module FGAN that solves the problem of data imbalance in deep learning and improves the accuracy of the model. Finally, we used the public dataset and laboratory dataset to evaluate CBBI from multiple dimensions. The evaluation results for different datasets show that our approach achieves the accurate identification of IoT devices.

1. Introduction

With the rapid development of IoT technology, the types and numbers of IoT devices have been growing quickly. The powerful connectivity and convenience of the IoT devices make their applications increasingly widespread, and they penetrate almost every corner of life, including smart wear, smart homes, smart entertainment, and smart travel. According to [1], approximately 31 billion IoT devices were used globally by the end of 2020, and approximately 75 billion IoT devices will be used by 2025, of which smart homes [2] will account for 41%, reaching 12.86 billion.

However, many vulnerabilities exist in current IoT devices and execution environments [3–5]. Attackers are increasingly concentrating on these vulnerable IoT devices, using device vulnerabilities to launch attacks [6–11]. For example, malicious attackers used a cloud of vulnerable IoT devices to build a sizable and highly destructive botnet to launch large-scale DDoS attacks. In the first 1 TB DDoS attack conducted on the Krebs Security website, more than 400,000 IoT devices were utilized [3]. Additionally, attackers can use these vulnerable IoT devices as proxies for malicious activities, further deteriorating the network environment. With the proliferation of IoT devices with security flaws, IoT device-centric attacks could further increase.

From the defensive aspect, network administrators must implement network access control on all connected devices. Whenever a new device is connected to the network, and if the device can be identified, then the network administrator can take appropriate security precautions. For example, the administrator configures the corresponding firewall rules according to the security requirements of the device, verifies whether the device has known vulnerabilities, or notifies the intrusion detection system to isolate vulnerable devices.

Many of the current studies on IoT device identification concentrate on features based on statistics and manual extraction and then combine fingerprint matching, machine learning, or deep learning to recognize IoT devices. These traditional IoT device identification methods face the following problems: (1) extracting features manually is a tedious and time-consuming process, and the low efficiency of feature extraction will affect the real-time performance of the classification model. (2) Feature extraction requires a professional domain prior knowledge and even professional feature engineering. Feature engineering involves feature extraction, feature construction, and feature selection, which, undoubtedly, further increases the difficulty of feature extraction. (3) The generalization ability of the model is also a concern. Before the training process in the traditional method, some seemingly trivial features might be discarded. These features could help the model improve its generalization ability, which would enable the model to be extended to more IoT devices. The feature space generated by traditional feature engineering is relatively small, and it is difficult to extract these subtle features. With the increase in the number and diversity of devices in practical applications, the recognition accuracy of the model could drop sharply. (4) A surging number of IoT devices use encryption protocols, increasing the difficulty of feature extraction and device identification. (5) With the increasing number of IoT devices, the amount of data generated by them is also increasing, and hence, feature extraction requires more time and resources. Therefore, it is difficult to meet the current needs using traditional features based on manual extraction.

In this paper, we present CBBI, a novel IoT device identification approach based on Conv-BiLSTM to learn the spatial and temporal features of the network traffic. CBBI contains three modules. The first module is the data preprocessing module, whose main task is to quickly process the raw network traffic generated by the IoT device and convert it into an input that can be used in the deep learning model. The second module is the data augmentation module FGAN that solves the problem of data imbalance in deep learning methods [12, 13]. The third module is to establish a deep learning model. We designed a hybrid deep learning model Conv-BiLSTM. Convolutional neural networks (CNNs) can learn the spatial characteristics of network communication traffic, such as the positional relationship of internal organizational structures in the network communication traffic. A bidirectional long short-term memory network (BiLSTM) can extract the time-domain characteristics of the network communication traffic, especially the timing relationship and flow duration of the data packets. The accuracy and generalization ability of the model are further improved by learning the spatial and temporal features simultaneously. Even when confronted with the IoT devices that have similar functions produced by the same device manufacturer, CBBI can use the powerful feature learning capabilities of deep learning to extract the representative features and some potential subtle features from the original traffic, and finally, it can realize the accurate identification of the IoT devices based on these learned features. This paper is an extended version of [4]. Firstly, compared with [4], we added a data augmentation module FGAN to solve the data imbalance. At the same time, the corresponding comparative test was also added. Secondly, we added our own dataset to the experiment and made some experimental evaluations based on the dataset. Finally, we added an additional visualization part to prove that CBBI can learn the representative spatial and temporal characteristics from the device communication traffic.

We summarize the main contributions as follows:

(1) We propose a novel IoT device identification method, CBBI. This method does not require any prior knowledge about feature engineering. It avoids the overhead of manual feature extraction and decreases the complexity of IoT device identification tasks.

(2) CBBI extracts the spatial and temporal features from the original traffic generated by the device, including some potential subtle features to identify the IoT devices, increasing the generalization ability of the model.

(3) CBBI contains the data augmentation module FGAN that solves the problem of data imbalance in deep learning and effectively improves the accuracy of the model.

(4) We conduct extensive experiments on the public dataset and laboratory dataset to evaluate the performance of CBBI. The results show the superiority of the proposed model.

The remainder of this paper is organized as follows. Section 2 summarizes the related work on IoT device classification. Section 3 describes our proposed IoT device identification method, which includes data preprocessing, data augmentation, and Conv-BiLSTM. Section 4 describes the experiment setup. Section 5 presents the evaluation results and analyses. Finally, we conclude this work in Section 6.

2. Related Work

For the identification of IoT devices, researchers have proposed many solutions. In this paper, the existing research studies are summarized and discussed from two aspects: device identification technology based on a classification model and device identification technology based on active detection.

2.1. Device Identification Technology Based on a Classification Model. Because of the differences in the software and hardware used in the IoT devices, there will also be subtle differences among the different devices produced by the same manufacturer. Researchers use the subtle differences in the hardware of the device, such as the clock offset [14–17], as the fingerprint of the device. Then, they construct a classification model to realize the accurate identification of the target device. In the traditional method, wireless devices can be identified by some unique radio frequency (RF) fingerprints caused by radio circuits [18, 19]. Yuan et al. [20] fingerprinted wireless devices by extracting the features caused by the hardware defects in the analog circuits. An important advantage of using these physical defects as device signatures for device identification is that it is difficult to use other wireless devices to spoof the signature. Brik et al. [21] designed and implemented a technology that uses the passive radio frequency analysis to identify the source network interface card (NIC) of IEEE 802.11 frames.

Radhakrishnan et al. [22] used the arrival interval time of packets in specific traffic types generated by the devices as the feature vectors. They used these feature vectors to train the artificial neural network (ANN). Miettinen et al. [23] proposed IoT Sentinel, which is a system for the automatic recognition of IoT devices. The system extracts 23 features from the data packets as device fingerprints and identifies the devices using a two-step classification method.

Guo and Heidemann proposed a method that analyses the DNS traffic to detect the IoT devices and identify their type [24]. Marchal et al. [25] automatically identified the type of IoT devices in the local network based on the periodic background network traffic of the IoT devices. The method needed 30 minutes to identify the type of devices, and the accuracy rate reached 98.2%.

Thangavelu et al. [26] used controllers to control the gateways based on a software-defined network (SDN). The controller implements the training and updating of the model and sends the newly trained model to each gateway. Sivanathan et al. [27] used statistical attributes such as the device traffic activity cycle, port number, signaling mode, and cipher suite as fingerprint features of the device. Then, they used a multistage machine learning classification algorithm to identify the device.

WDMTI [28] uses 18 features extracted from DHCP messages to establish a hierarchical Dirichlet process (HDP) model to identify the wireless devices. This method relies on the bursts of traffic when the device is connected to the network. OWL [29] analyzed the broadcast and multicast packets in the wireless local area networks (WLANs), built a multiview deep learning (MvWDL) model based on the features extracted from each protocol message, and classified the IoT devices.

According to the unique network traffic pattern of the IoT devices, Deng et al. [30], firstly, extracted all available features from each TCP flow header. Secondly, they used the principal component analysis (PCA) algorithm to select the main features that affect device recognition. Finally, they learned the device-specific network traffic signature based on a random forest classifier to achieve device identification. Yin et al. [4] proposed an end-to-end IoT device identification method that directly uses the original communication traffic generated by the device. This method fails to fully consider the problem of data imbalance. In the face of extremely unbalanced datasets, the performance of the model may be greatly compromised.

2.2. Device Identification Technology Based on Active Detection. Active detection refers to actively sending detection packets to the devices in the network, obtaining response packets, and extracting device information by analyzing the information in the response packets. Attackers usually obtain information about vulnerable devices in the network through active detection before launching an attack to improve the accuracy of the attack. Researchers also use the active detection method to determine the state of the devices in the network, so they can take further security measures to ensure the safety of the devices in the network.

In practice, because of the large number of IoT devices and the lack of training data, researchers use banner information instead of device fingerprints to identify the IoT devices. Antonakakis et al. [31] applied the banner rules to analyze the online devices from Censys [32] and Honeypot. Shodan [33] and Censys [32] are the two popular search engines that are mainly used to discover online devices. Both search engines use different protocols (such as HTTP, SSH, FTP, and TELNET) to perform Internet-wide scans.

Many researchers [34, 35] use banner information acquisition to actively scan the devices in the IP space. They collect and check the text features from the response, such as hard-coded keywords, and match them with known fingerprints for device identification.

Li et al. [36] established a framework for searching devices on the Internet using network measurement and banner grabbing to obtain services running on the network hosts and to match the response header fields with prestored keywords to retrieve device information.

Feng et al. [37] proposed an acquisition rule-based engine (ARE) that can automatically generate rules for discovering and annotating the IoT devices without any training data. ARE uses the application layer response data from the IoT devices and product descriptions in the related websites to obtain device comments, thereby constructing device rules. It solves the cumbersome and incomplete shortcomings of traditional methods based on manually writing banner information capture rules.

Table 1 summarizes the main references aforementioned and shows the features and methods used in the relevant references.

The aforementioned research works made important contributions to the identification of IoT devices and promoted the development of network security. The device fingerprint identification method based on the classification model mainly uses the physical difference of the device as the fingerprint of the device. Otherwise, it manually extracts some field values and related statistical characteristics in the device communication traffic. Then, it is combined with machine learning or deep learning methods to construct a classification model. The device identification method based on active detection must actively send a multitude of detection packets to the target devices in the network, which is susceptible to packet loss and network delay. In addition, the frequent transmission of probe packets will increase the load on the network and aggravate the deterioration of the network environment. More importantly, if the device does not generate a response or if there is no valid information in the response packets, the device cannot be further identified.

Table 1: Summary of related works.

References	Features	Method
[14–17]	Clock skew	—
[18–21]	Radio frequency fingerprint	—
[22]	Clock skew	ANNs
[23]	Features from the packet head	Twofold identification technique (Random Forest + Edit Distance)
[24]	Flow-level network traffic and knowledge of servers run by the manufacturers	—
[25]	Periodic communication traffic features	KNN
[26]	Features from DNS queries and HTTP URI's	Improved k-means algorithm, Random Forest, SDN
[27]	Statistical attributes such as activity cycles, port numbers, signaling patterns, and cipher suites	A multistage machine learning (Naive Bayes + Random Forest)
[28]	18 features of DHCP	Dirichlet process
[29]	Features from passively received broadcast and multicast packets	Multiview wide and deep learning framework
[30]	Features in TCP header per TCP flow	PCA, Random Forest
[4]	Raw network traffic from devices	CNN, BiLSTM
[31]	Banners, honeypots	Active scanning
[34–36]	Banners	Active scanning, match
[37]	Banners	Active scanning, search and match

3. Proposed Framework

The overall structure of the CBBI framework is shown in Figure 1. CBBI is composed of three modules: data preprocessing, data augmentation, and Conv-BiLSTM. Initially, the data preprocessing module converts the raw network traffic generated by the IoT device into an input that can be used in the deep learning model. Furthermore, the data augmentation module FGAN solves the problem of data imbalance in deep learning. Finally, the Conv-BiLSTM module simultaneously learns the spatial and temporal characteristics of the original traffic of the device, which improves the accuracy and generalization ability of the model.

3.1. Data Preprocessing. In general, deep learning models cannot directly use the raw pcap data. These original pcap files need to be processed into a format suitable for model input. The entire data preprocessing process includes three parts: flow generation, irrelevant field removal, and traffic vectorization.

3.1.1. Flow Generation. The original communication traffic generated by the IoT devices contains different numbers of data packets, and the length of each data packet is also inconsistent. In other words, the original communication traffic generated by the devices can be defined as $P = \{p_1, \ldots, p_n\}$, and each data packet can be defined as $p_i = (x_i, s_i, t_i)$. The value of i is $i = 1, 2, \ldots, |n|$, where x_i represents the 5-tuple information (source IP address, source port number, destination IP address, destination port number, and transport layer protocol type) of the packet, s_i represents the size of packet p_i, and t_i represents the starting time of packet p_i.

In this paper, the existing Splitcap tool [38] is utilized to process the original network traffic into a network flow with the same 5-tuple information, where the network flow can be defined as $F_i = \{p_1 = (x_1, s_1, t_1), p_2 = (x_2, s_2, t_2), \ldots, p_m = (x_m, s_m, t_m)\}$. Here, m represents the number of data packets in the network flow. As the data packets in the network flow have the same 5-tuple information, $x_1 = x_2 = \cdots = x_m$. The network flow has a certain time order, and thus, the data packets in the network flow have a sequence, represented by $t_1 < t_2 < \cdots < t_m$.

Each network flow is composed of several packets. The network flow contains substantial behavior characteristics of IoT device communication traffic, including the closeness of the relationship among the bytes in the data packet, the duration of each network flow, the number and size of the data packets that constitute the network flow, and the timing relationships among the data packets. These traffic behavior characteristics can help the deep learning model to better recognize the device and improve the accuracy of the model.

3.1.2. Irrelevant Fields Removal. CBBI makes use of the traffic behavior characteristics of IoT devices. Here, we need to eliminate some interference data, such as MAC addresses and IP addresses, to prevent these data from affecting the experimental results. In a small LAN, the number of devices is limited, and the MAC addresses of the devices can uniquely identify the devices. These field values can occupy a relatively large weight in the process of the feature extraction of the deep learning model, which could affect the real recognition and classification ability of the model. It can even lead to the overfitting of the model. The IP address of the device has the same interference effect as the MAC address. In this paper, these interference fields are eliminated in the data processing module to prevent them from affecting the process of model feature learning.

3.1.3. Traffic Vectorization. The neural network requires the input data to have a standardized format, and we must convert the processed data aforementioned into a suitable input format. The number of data packets in each network

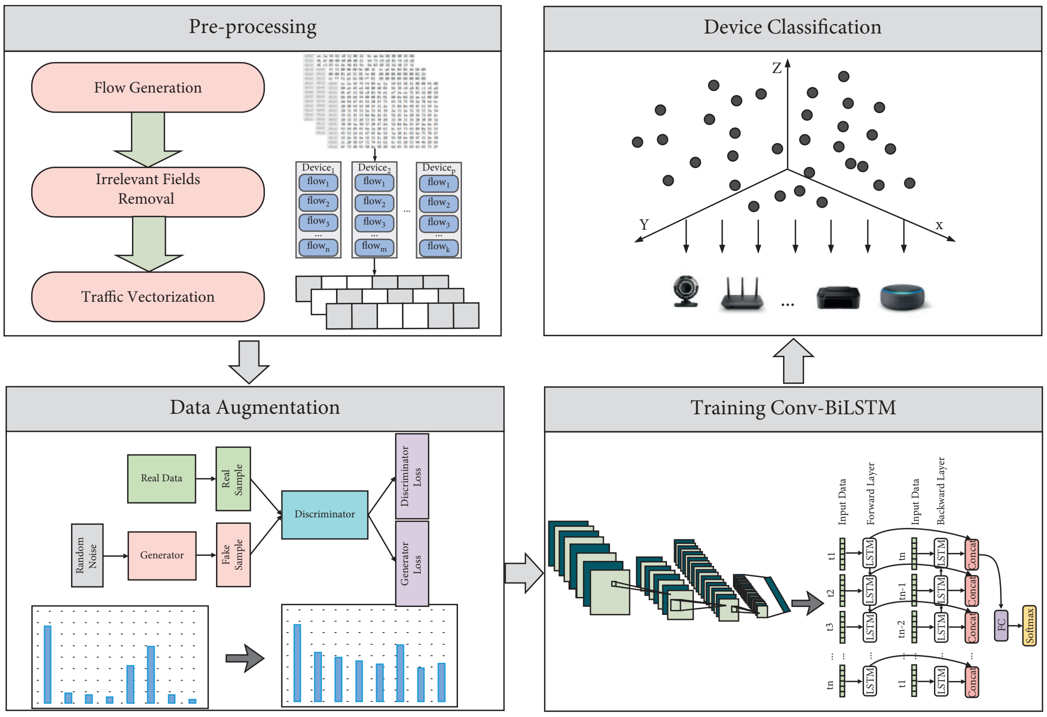


Figure 1: An overview of the CBBI framework.

flow and the size of each data packet are different, and thus, a unified standard must be determined to vectorize the features in the network flow. We performed a statistical analysis based on the public dataset and laboratory dataset. As shown in Figure 2, we found that the number of data packets in the network flows in the two datasets is mostly within 10, and most of the data packets are within 250 bytes in size. According to the statistical information, each network flow intercepts 2500 bytes of data samples. In other words, each network flow selects the first 10 packets ($n = 10$), and each packet intercepts the first 250 bytes ($L = 250$ bytes). If the number of data packets N in the network flow is less than 10, or the length of the data packet L is less than 250 bytes, then it is directly filled with 0. The representation of network flow characteristics is shown in Figure 3. The complete data preprocessing algorithm is shown in Algorithm 1.

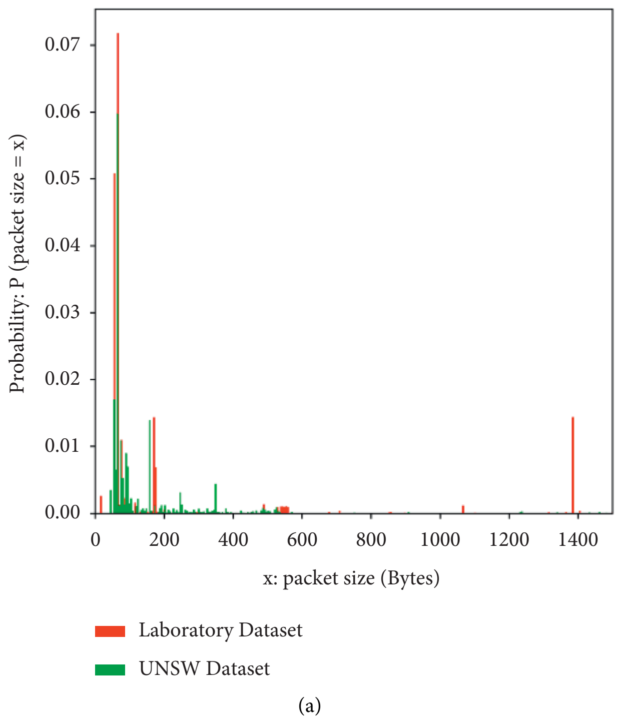
3.2. Data Augmentation. The network traffic generated by the IoT devices can be transformed into different numbers of data samples after the preprocessing stage. Because of the different functions, the software, and the hardware of the devices, the traffic model generated by each device is very different. For example, the network traffic generated by the video monitoring devices is very large, whereas the network traffic generated by some sensors is relatively limited. Therefore, there is a large difference in the number of samples that correspond to the device, which leads to the serious problem of data imbalance. As far as the learning model is concerned, the sample is usually considered to be an unbiased sample of the true distribution. When the training set is largely skewed, it usually does not reflect the true distribution. The imbalance of the sample distribution causes the model prediction result to be biased; in other words, the classification result is biased toward more sample categories, and the result is misleading. Therefore, we must adjust the generated sample data to alleviate the imbalance and further improve the performance of the model.

This paper uses the GAN-based data augmentation module FGAN, as shown in Figure 4. The generative adversarial network (GAN) is an adversarial network proposed by Goodfellow [39] in 2014. The network framework consists of two parts, a generator and a discriminator. The generator tries to cheat the discriminator by constructing false data. It accepts arbitrary noise $p_z(z)$ and generates false data according to the noise, which is recorded as $G(z)$. The discriminator tries to distinguish whether the data came from a real sample or fake data constructed by a forger. The input parameter of the discriminator is x, which comes from $p_{\text{data}}(x)$. The output $D(x)$ of the discriminator represents the probability that x is the real data. Both models improve their abilities using continuous learning. In other words, the generator hopes to generate more real fake data to cheat the discriminator, and the discriminator hopes to learn how to more accurately identify the fake data of the generator. The objective function v of FGAN is as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{\mathbf{z}\sim p_z(z)}[\log(1-D(G(v)))]. \quad (1)$$

The basic flow of the network is as follows:

(i) Initialize parameter θ_d of discriminator D and parameter $\theta_g$of generator G.



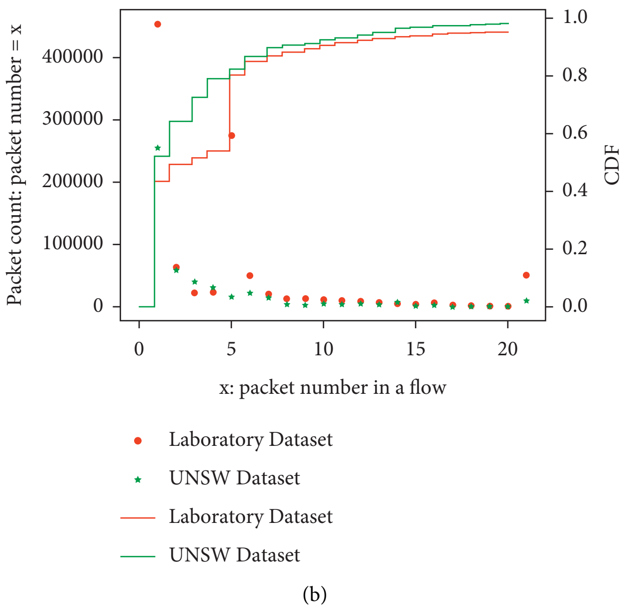


Figure 2: Statistics of the network traffic in the two datasets. (a) Statistics of the packet size. (b) Statistics of the number of data packets in the network traffic flow.

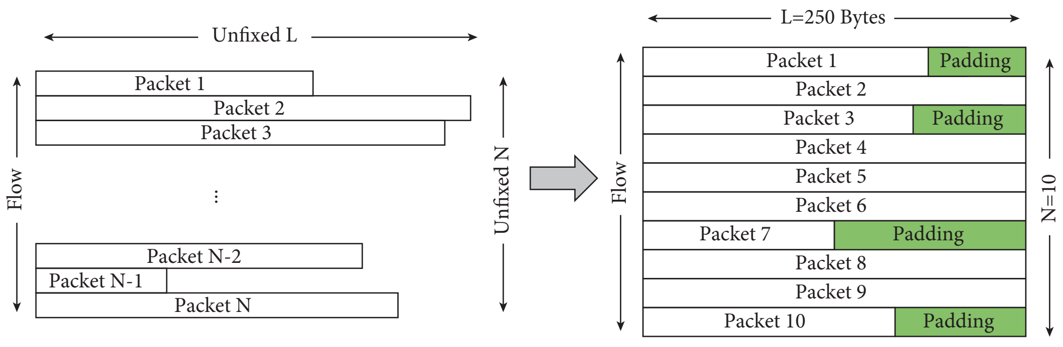


Figure 3: Representation of the network traffic flow.

(ii) Firstly, n sample data $\{x^1, x^2, \ldots, x^n\}$ are obtained from the real samples. Then, n noise samples $\{z^1, z^2, \ldots, z^n\}$ are sampled from the prior distribution noise. Secondly, n samples $\{\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^n\}$ are produced using the generator. Finally, the generator G is fixed, and the discriminator D is trained to identify the real data from the generated data as accurately as possible.

(iii) After updating the discriminator for k epochs, the parameters of the generator are updated once with a small learning rate, and the generator is trained to reduce the gap between the generated data and the real data as much as possible.

(iv) After many iterations of updates, the final ideal is for the discriminator to be unable to tell whether the sample comes from the output of the generator or the real output.

This paper designs the generator and discriminator in FGAN based on the fully connected network. Detailed information will be in Section 4.3.

3.3. *Training Conv-BiLSTM for IoT Device Classification.*

In this section, we build a deep learning model Conv-BiLSTM for identifying the IoT devices. This model is different from the traditional classification methods based on manually extracted features and statistical features. Firstly, the model can simultaneously learn the spatial and temporal features of the device traffic, which improves the accuracy of device identification. In addition, the traditional device identification method based on manual design and statistical features has some limitations. When these features are designed and selected artificially, the inherent features in the original communication flow of the device are changed, and some potential features are ignored. These potential features can

```
Input: raw network traffic pcap files
Output: Samples_Data
Samples_Data $\longleftarrow \varnothing$
Use SplitCap tool to convert raw pcap files into Flows with the same 5-tuple
information, $F1, F2, \ldots, Fm$, $m$ represents the last Flow in raw pcap
(1) for each $Fi$ do
(2)   Packets_Number$\leftarrow$Len ($Fi$), Flow_Feature $\longleftarrow \varnothing$
(3)   if Packets_Number > = 10 then
(4)     Packets_Sequence $\longleftarrow$ Get the top 10 packets in $Fi$
(5)     for each $P$ in Packets_Sequence do
(6)       if length ($P$) > = 250 bytes then
(7)         Packet_Feature $\longleftarrow$ Get the first 250 bytes in $P$
(8)       else
(9)         Packet_Feature $\longleftarrow$ Get all bytes in $P$ + "0" * (250 − length ($P$))
(10)        Set the MAC address field and IP address field in Packet_Feature to 0
(11)        Flow_Feature $\leftarrow$Flow_Feature $\cup$ Packet_Feature
(12)    end for
(13)  else
(14)    Packets_Sequence $\longleftarrow$ Get all packets in $Fi$
(15)    for each $P$ in Packets_Sequence do
(16)      if length ($P$) > = 250 bytes then
(17)        Packet_Feature $\longleftarrow$ Get the first 250 bytes in $P$
(18)      else
(19)        Packet_Feature $\longleftarrow$ Get all bytes in $P$ + "0" * (250 − length ($P$))
(20)      Set the MAC address field and IP address field in Packet_Feature to 0
(21)      Flow_Feature $\longleftarrow$Flow_Feature $\cup$ Packet_Feature
(22)    end for
(23)    for $j = 1$; $j <= 10 -$ Packets_Number; $j++$ do
(24)      Flow_Feature $\longleftarrow$Flow_Feature $\cup$ ("0" * 250)
(25)    Samples_Data.append(Flow_Feature)
(26) end for
(27) Return Samples_Data
```

Algorithm 1: The algorithm for data preprocessing.

help to improve the recognition accuracy and generalization ability of the model. In addition, artificially designed features might not fully represent the high-level semantics of the network traffic, and models trained based on these features cannot learn these high-level semantics. The Conv-BiLSTM network model can learn highly semantic features from the original communication traffic generated by the device.

The CNN [40] is widely used in the field of image classification because of its influential spatial feature learning ability. The CNN has a convolutional layer, pooling layer, and fully connected layer. The main function of the convolutional layer is to extract features. The pooling layer implements data subsampling without destroying the classification results in terms of reducing the dimensionality of the features, compressing the data, and avoiding the overfitting of parameters. The convolutional layer and the pooling layer play the role of mapping the original data to the hidden layer feature space. The fully connected layer is a fully connected neural network. The weight parameters are adjusted by weighing the proportion of each neuron's feedback. The model also uses dropout to avoid overfitting.

LSTM is a special recurrent neural network (RNN) [41]. The difference between LSTM and the standard recurrent neural network is that the LSTM overcomes the problems of gradient explosion and gradient disappearance by introducing memory units and gate mechanisms, and it performs well in extracting the long-term dependence in the sequence data. The LSTM architecture is composed of an input gate, forget gate, output gate, storage unit, hidden state, and so on. The specific calculation process of the input gate, output gate, and forget gate is as follows.

3.3.1. Forget Gate. f_t is called the forget gate, which indicates that some features of c_{t-1} are used to calculate c_t. f_t is obtained by a logical function to calculate the input x_t and the last hidden layer value h_{t-1}. The value of the forget parameter is between 0 and 1, which controls how much information is retained from c_{t-1} to c_t. Here, 1 means to retain the information completely, whereas 0 means to discard the information entirely.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right). \quad (2)$$

3.3.2. Input Gate. The input gate decides what new information to store in the "cell state." The sigmoid layer decides what value is to be updated. The tanh layer creates a new

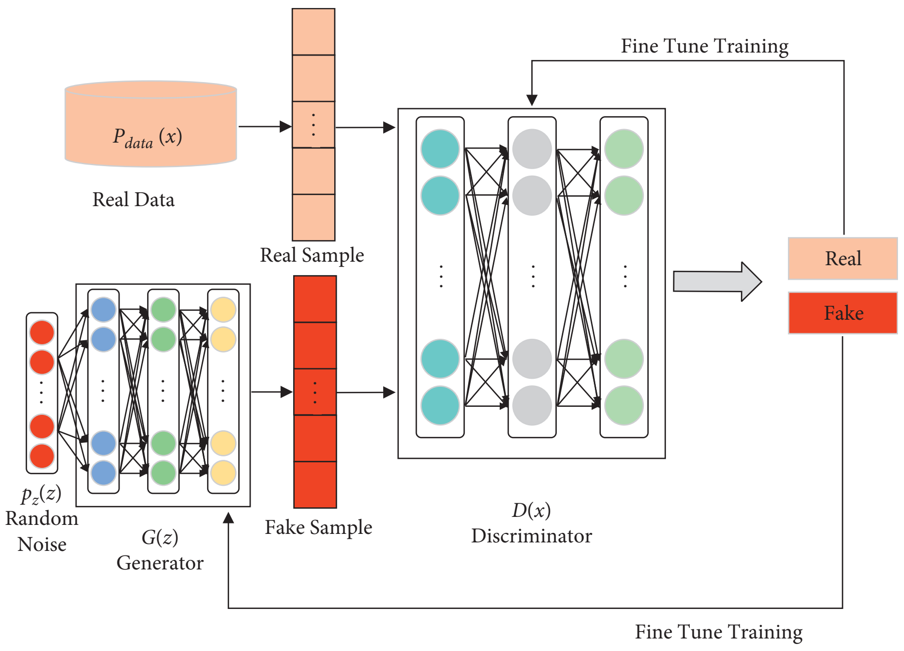


FIGURE 4: The framework of FGAN.

candidate value vector $\tilde{c}_t$, and i_t determines the part of the information to be updated. When updating c_{t-1} to $\tilde{c}_t$, we must multiply the old state with f_t, discard the information that needs to be discarded, and add $i_t * \tilde{c}_t$.

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right),\\ \tilde{c}_t &= \tan h\left(W_c \cdot \left[h_{t-1}, x_t\right] + b_c\right),\\ c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t. \end{aligned} \tag{3}$$

3.3.3. Output Gate. h_t can be considered the last output at the current moment. h_{t-1} is the output at $t-1$. o_t is a probability vector that is used to determine which part is the output. Firstly, we run a sigmoid layer to determine which part of the cell state is to be the output. Then, tanh is used to process the cell state (obtaining a value between −1 and 1). Finally, this value is multiplied with the output of the sigmoid gate to obtain the output.

$$\begin{aligned} o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right),\\ h_t &= o_t * \tan h\left(c_t\right). \end{aligned} \tag{4}$$

Unlike other types of deep neural networks, LSTM shares weights at all time steps, which reduces the number of parameters that the network must learn. The BiLSTM [42] is composed of two LSTMs: one LSTM is the input forward, whereas the other LSTM is the output backward. BiLSTM effectively increases the amount of information available to the network and improves the context available for the algorithm. BiLSTM can not only address gradient disappearance and gradient explosion, as in the LSTM, but also learn more context information from the network.

The Conv-BiLSTM network model structure is shown in Figure 5.

The convolutional neural network model used in this paper is improved on the basis of the classical lenet-5 [43]. The convolutional neural network constructed in this paper has seven layers. More detailed network structure information is provided in Section 4.3. The training process of the Conv-BiLSTM model is shown in Algorithm 2.

The feature dimension of each sample after CNN is 1600. We reshape the 1600-dimensional data into a $10 * 160$ format and input it into BiLSTM, where 10 represents the number of time steps. The vector dimension of each time point is 160. The BiLSTM consists of two layers, each with 512 hidden cells, and each layer uses the sigmoid function for nonlinear operations. The last layer of the BiLSTM network adopts the fully connected layer, and the number of neurons in the fully connected layer is equal to the number of IoT devices. Softmax is used as the activation function, which maps the output of multiple neurons to (0, 1), and the sum of each output is 1. The type with the largest probability value can be selected for multiple classifications.

4. Evaluation Setup

4.1. Computing Platform Configurations. We use Keras [44] as the neural network framework to construct the Conv-BiLSTM model. The detailed configuration information is shown in Table 2.

4.2. Dataset Description. The UNSW dataset is the traffic data generated by the IoT devices in two weeks. The dataset contains a total of 22 IoT devices. Some of these devices generate very little communication traffic. For example, Withings_Smart_Scale and Blipcare_Blood_Pressure_meter generated 8 and 13 sample data, respectively, after data preprocessing. In the experimental part, we selected 18 IoT devices with relatively large sample sizes.

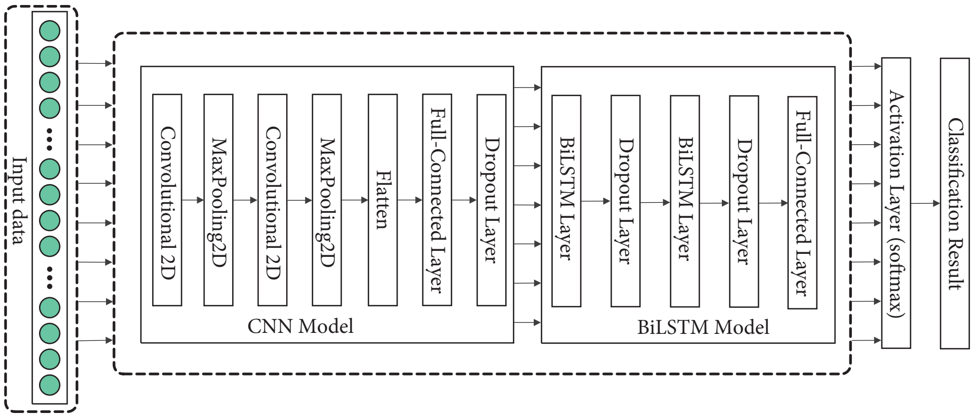


FIGURE 5: The architecture of Conv-BiLSTM.

```
Input:
    Samples_Data composed of network flows, the dimension of each network flow is 2500.
    {Epoch, Batchsize, dropout, Loss_function} represent some of the parameters during model training.
Output:
    The categories of Samples_Data
(1)  for each epoch in (1, Epoch) do
(2)    for each Batchsize data of the Samples_Data do
(3)      for each Sample_Data in batch do
(4)        Reshape Sample_Data to 50 * 50 form
(5)        Compute convolution with 6 filters
(6)        Compute the result through Relu
(7)        Max Pooling
(8)        Compute convolution with 16 filters
(9)        Compute the result through Relu
(10)       Max Pooling
(11)       Flatten the data
(12)       Run through a densely connected layer
(13)       Dropout
(14)       Reshape output data as 10 * 160
(15)       Run through the 2-layered BiLSTM with dropout
(16)       Run through a densely connected layer
(17)       Output the result referring Loss_function
(18)       Update the parameters of weight and bias
(19)     end for
(20)   end for
(21) end for
```

ALGORITHM 2: Training process of the Conv-BiLSTM model.

We built an IoT device traffic collection platform in the laboratory environment. We collected the two-week communication traffic of 23 IoT devices, covering a variety of device types and device brands, including 360, Amazon, Hikvision, Huawei, TP-Link, Xiaomi, and other common IoT device manufacturers. The device types include smart cameras, smart speakers, smart gateways, smart doorbells, and so on. Also, there are IoT devices of the same brand and type, but of different models, such as the two cameras Hikvision_DS-IPC-E22H-IW and Hikvision_DS-IPC-S12P-IWT from Hikvision and the three smart cameras TP_Link_Camera_IPC42A-4, TP_Link_Camera_IPC43AN-4,and TP_Link_Camera_IPC64C-4 from TP-Link. The data traffic generated by 23 IoT devices was processed to generate a total of 636,789 sample data. Detailed information on the UNSW dataset and the laboratory dataset is shown in Table 3.

As shown in Table 3, the number of samples of cameras in the two datasets is relatively vast. The traffic data generated by some cameras in the laboratory dataset is not very large, such as D-Link-DSH-C310, Hikvision_DS-IPC-E22H-IW, and Hikvision_DS-IPC-S12P-IWT. We checked the settings of these devices and found that they adopted the "Standard Definition" video recording method rather than the "High Definition" or "Super Definition" as the other cameras did. Some cameras also

Table 2: Experiment settings.

Category	Parameter
Operating system	Ubuntu 16.04 LTS OS
CPU	Intel(R) Xeon(R) Bronze 3106 CPU@ 1.70GZ
Deep learning platform	Keras
Deep learning backend	TensorFlow-gpu 2.2.0
GPU version	NVIDIA GP104GL(Quadro P4000)
CUDA version	10.1.243
CuDNN version	7.6.5

enabled face recognition, automatic tracking, and other modes, and hence, the communication traffic generated was enormous.

4.3. Parameter Settings. This section provides detailed information about the FGAN and Conv-BiLSTM network structures used in the experiment. Both generator and discriminator in FGAN are implemented based on a multilayer perceptron (MLP). The specific information is shown in Tables 4 and 5. The input of the generator is a 100-dimensional Gaussian noise vector, and the hidden layer contains 256, 512, 1024, and 2500 neurons. The input of the discriminator contains both real data and generated data, and its dimension is 2500. The LeakyReLU activation function, dropout, and BatchNormalization are used in FGAN to optimize the model.

Detailed information on Conv-BiLSTM is shown in Table 6, including the structural parameters of each layer of the network, the optimizer, loss function, and other hyperparameters.

4.4. Evaluation Metrics. To evaluate the performance of the neural network model, this paper selects four performance metrics: the recall, precision, accuracy, and F1-score:

$$\begin{aligned} \text{recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}, \\ \text{F1} - \text{score} &= \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}, \end{aligned} \tag{5}$$

where TP, TN, FP, and FNdenote the true positives, true negatives, false positives, and false negatives, respectively.

5. Experimental Results and Analyses

5.1. Ablation Study. To verify the effectiveness and rationality of the data augmentation module FGAN in CBBI, we performed the corresponding experiments on the UNSW dataset and the laboratory dataset. We used the precision, recall, and F1-score to evaluate the results of the experiment. Tables 7 and 8 show the experimental results on CBBI, including FGAN, on the UNSW dataset and the laboratory dataset, respectively. From the two tables, it can be seen that FGAN in CBBI has well alleviated the problem that the classification results of a small number of samples are biased toward large sample classes due to data imbalance. The small sample classes iHome, Nest_Dropcam, NEST_Procet_Smoke_Alarm, and Triby_Speaker in the UNSW dataset and D-Link-DSH-C310, Huawei_Smart_Scale, Huawei_Smart_Scale, Xiaomi_Air_Purifier, and Xiaomi_Hub in the laboratory dataset have significantly improved the performance after using FGAN. The performance of other categories has also been improved to varying degrees as the samples become more balanced. The data augmentation FGAN module in CBBI realizes the relative balance of the sample and further improves the classification accuracy of the model.

5.2. Misclassification Analysis. To analyze the misclassification of the CBBI model in the two datasets, we give the confusion matrixes of the experimental results in the two datasets, as shown in Figure 6. The classification accuracy of most of the devices in the UNSW dataset is close to 100%. The accuracy of Nest_Dropcam is 96%, and 4% of its data samples are identified as Netatmo_Welcome. These two devices are products of two different device manufacturers, however, both belong to the smart camera type, and there are certain similarities in the traffic model.

The accuracy of CBBI in the laboratory dataset reached 97.26%, which is not as high as that in the UNSW public dataset. We can determine the following reasons by analyzing the experimental data and the results: (1) the laboratory dataset contains more IoT devices than the UNSW dataset, which increases the difficulty of multiclassification of the model; (2) the number of samples generated by the devices in the laboratory dataset is more unbalanced, which affects the fitting effect of the model; (3) there are more devices from the same manufacturer and type in the laboratory dataset, and there are more similarities between the devices; the confusion matrix shows that IoT devices of the same device manufacturer and type are prone to misclassification between one another. Hikvision_DS-IPC-E22H-IW and Hikvision_DS-IPC-S12P-IWT from Hikvision, Ezviz_Camera_CS-C6CN and Ezviz_Door_CS-DB2C from Ezviz, as well as four devices from TP-Link, have all been misclassified to varying degrees. The worst classification effect in the laboratory dataset is TP_Link_WDA6332RE and Xiaomi_Air_Purifier. The sample size of these two devices is extremely small, and FGAN has improved the classification accuracy of these two devices to a certain extent.

5.3. Visualization of Spatial and Temporal Features. In this section, we input the spatial and temporal feature vectors learned by CBBI from the UNSW dataset and laboratory dataset into the t-SNE algorithm before applying softmax

Table 3: Description of the UNSW dataset and laboratory dataset.

UNSW dataset			Laboratory dataset		
No.	Device name	Sample number	No.	Device name	Sample number
0	Amazon_Echo	73780	0	360_Camera	5950
1	Belkin_Wemo_Switch	17148	1	Amazon_Echo	9584
2	HP_Printer	2794	2	D-Link-DSH-C310	836
3	Insteon_Camera	216088	3	Hikvision_DS-IPC-E22H-IW	2082
4	Light_Bulbs_LiFX_Smart_Bulb	7226	4	Hikvision_DS-IPC-S12P-IWT	2149
5	Netatmo_Weather_Station	4703	5	Ezviz_Camera_CS-C6CN	65366
6	Netatmo_Welcome	16000	6	Ezviz_Door_CS-DB2C	88273
7	PIX_STAR_Photo_Frame	12311	7	Huawei_Camera_HQ5	35742
8	Samsung_SmartCam	61815	8	Huawei_Speaker	50183
9	Smart_Things	10367	9	Huawei_Smart_Scale	378
10	TP_Link_Day_Night_Cloud_Camera	4954	10	Imou-Camera_TP1-2525	19534
11	TP_Link_Smart_Plug	2586	11	Imou-Camera_TP7C-E152	15110
12	Withings_Aura_Smart_Sleep_Sensor	13963	12	Philips_Hue	103687
13	Withings_Smart_Baby_Monitor	20229	13	TP_Link_Camera_IPC42A-4	79354
14	iHome	346	14	TP_Link_Camera_IPC43AN-4	61061
15	Nest_Dropcam	154	15	TP_Link_Camera_IPC64C-4	61815
16	Nest_Procet_Smoke_Alarm	221	16	TP_Link_WDA6332RE	221
17	Triby_Speaker	669	17	Xiaomi_Air_Purifier	133
—	—	—	18	Xiaomi_Camera_MJSXJ06CM	22120
—	—	—	19	Xiaomi_Door_LSC_M01	4203
—	—	—	20	Xiaomi_Hub	242
—	—	—	21	Xiaomi_Humidifier	1414
—	—	—	22	Xiaomi_Soundbox	7352
Total	465354		Total	636789	

Table 4: The structure of the generator.

Layer	Output shape
Dense_1(Dense)	(None, 256)
LeakyReLU	(None, 256)
BatchNormalization	(None, 256)
Dense_2(Dense)	(None, 512)
LeakyReLU	(None, 512)
BatchNormalization	(None, 512)
Dense_3(Dense)	(None, 1024)
LeakyReLU	(None, 1024)
BatchNormalization	(None, 1024)
Dense_4(Dense)	(None, 2500)
LeakyReLU	(None, 2500)

Table 5: The structure of the discriminator.

Layer	Output shape
Dense_1(Dense)	(None, 2500)
LeakyReLU	(None, 2500)
Dropout	(None, 2500)
Dense_2(Dense)	(None, 1024)
LeakyReLU	(None, 1024)
Dropout	(None, 1024)
Dense_3(Dense)	(None, 512)
LeakyReLU	(None, 512)
Dropout	(None, 512)
Dense_4(Dense)	(None, 256)
LeakyReLU	(None, 256)
Dropout	(None, 256)
Dense_5(Dense)	(None, 1)

classification to achieve dimension-reduction visualization. The dimensions of each sample input to the t-SNE algorithm in the UNSW dataset and the laboratory dataset are 18 and 23, respectively, which are consistent with the number of IoT devices in the two datasets. The visualized effect of the dimensionality reduction result is shown in Figure 7. In the laboratory dataset, some devices are not highly distinguished, especially for several devices with the same manufacturer and type. The visualization results of these two datasets are consistent with the aforementioned experimental results. The clustering effect of the two datasets is excellent, and the separation distance among the different categories is relatively obvious. In general, CBBI can learn representative spatial and temporal characteristics from device communication traffic, which can be used as the basis for device identification.

5.4. Comparison Results. The classification accuracy of CBBI on the UNSW dataset is 99.83%, which achieves a similar effect to UNSW [27]. The detailed experimental results are shown in Figure 8. As far as we know, UNSW [27] is currently the highest accuracy rate for IoT device identification-related work, reaching 99.88%. The study in [27] used 6 months of IoT device communication traffic data. In addition, the author achieves accurate identification of IoT devices based on manually extracted features combined with a multistage device identification framework. The UNWS dataset that we used contains two weeks of traffic data, and thus, the communication traffic

Table 6: The structure of Conv-BiLSTM.

Hyperparameters		Value	Activation function
Conv-BiLSTM	Conv2D	#filters = 6, filter size = 5	ReLU
	MaxPlooing2D	#pool size = 2, padding = "valid"	—
	Conv2D	#filters = 16, filter size = 5	ReLU
	MaxPlooing2D	#pool size = 2, padding = "valid"	—
	Flatten	—	—
	Dense	#neurons = 1600, dropout = 0.5	ReLU
	BiLSTM	#neurons = 512, dropout = 0.3	Sigmoid
	BiLSTM	#neurons = 512, dropout = 0.3	Sigmoid
	Dense	#neurons = 18/23	Softmax
Optimizer	—	#Adam with learning rate = 0.001	—
Loss function	—	#Categorical_crossentropy	—
Batch size	—	#512	—
Epochs	—	#50	—

Table 7: Results of the ablation study on the UNSW dataset.

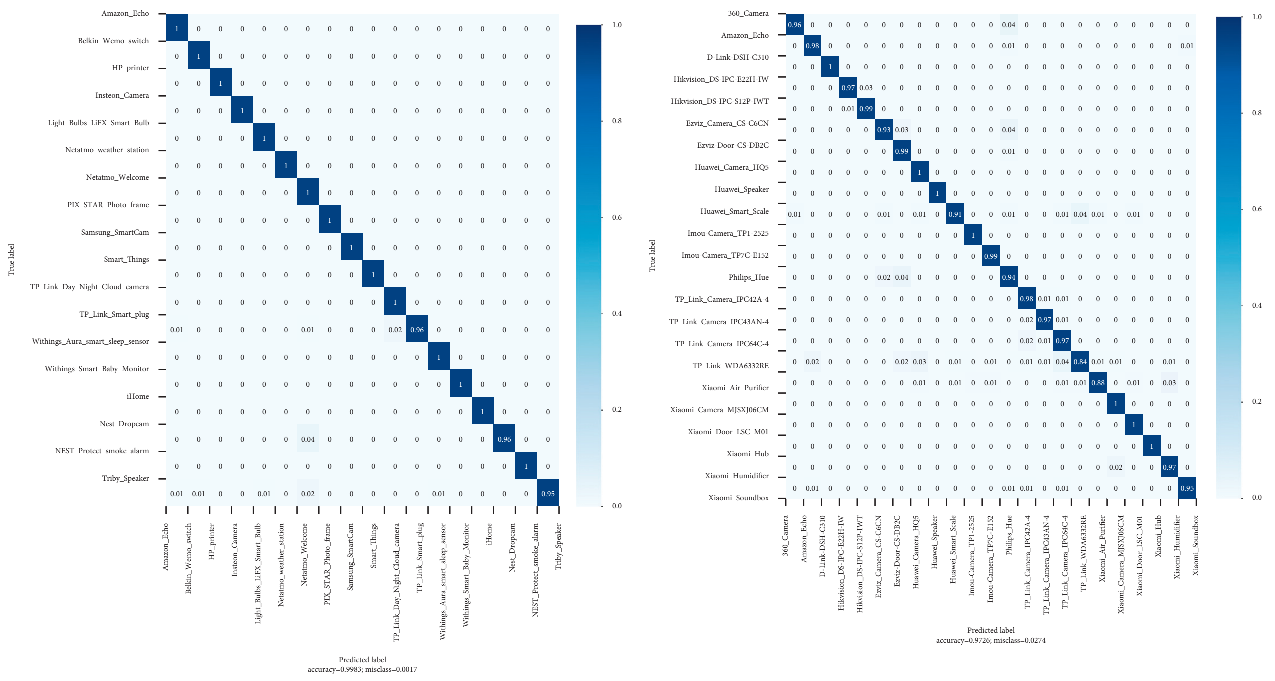
Method	CBBI w/o FGAN			CBBI		
FGAN		✗			✓	
Conv-BiLSTM		✓			✓	
Device name	Precision	Recall	F-score	Precision	Recall	F-score
Amazon_Echo	0.9878	0.9932	0.9905	0.9983	0.9988	0.9985
Belkin_Wemo_Switch	0.9932	0.9929	0.9931	0.9982	0.9988	0.9985
HP_Printer	0.9056	0.9646	0.9342	0.9908	0.9981	0.9944
Insteon_Camera	0.9931	0.9981	0.9956	0.9994	0.9994	0.9994
Light_Bulbs_LiFX_Smart_Bulb	0.9825	0.9825	0.9825	0.9979	0.9965	0.9972
Netatmo_Weather_Station	0.9865	0.9963	0.9914	0.9988	1.0000	0.9994
Netatmo_Welcome	0.9181	0.9533	0.9354	0.9937	0.9956	0.9947
PIX_STAR_Photo_Frame	0.9926	0.9887	0.9907	0.9990	0.9951	0.9971
Samsung_SmartCam	0.9978	0.9922	0.9950	0.9992	0.9989	0.9991
Smart_Things	0.9956	0.9888	0.9922	0.9985	0.9980	0.9983
TP_Link_Day_Night_Cloud_Camera	0.9898	0.9563	0.9728	0.9892	0.9989	0.9940
TP_Link_Smart_Plug	0.8638	0.8675	0.8657	0.9804	0.9615	0.9709
Withings_Aura_Smart_Sleep_Sensor	0.9929	0.9793	0.9861	0.9989	0.9985	0.9987
Withings_Smart_Baby_Monitor	0.9823	0.9961	0.9891	0.9997	0.9997	0.9997
iHome	1.0000	0.6833	0.8119	1.0000	1.0000	1.0000
Nest_Dropcam	0.3193	0.4371	0.3980	1.0000	0.9600	0.9796
NEST_Procet_Smoke_Alarm	1.0000	0.7317	0.8451	1.0000	1.0000	1.0000
Triby_Speaker	0.8841	0.4552	0.6010	0.9817	0.9469	0.9640
Macro average performance	0.9325	0.8865	0.9039	0.9958	0.9914	0.9935
Total accuracy		0.9865			0.9983	

Table 8: Results of the ablation study on the laboratory dataset.

Method	CBBI w/o FGAN			CBBI		
FGAN		✗			✓	
Conv-BiLSTM		✓			✓	
Device name	Precision	Recall	F-score	Precision	Recall	F-score
360_Camera	0.9504	0.9017	0.9254	0.9697	0.9566	0.9631
Amazon_Echo	0.9023	0.9395	0.9205	0.9435	0.9764	0.9597
D-Link-DSH-C310	0.8882	0.8988	0.8935	1.0000	1.0000	1.0000
Hikvision_DS-IPC-E22H-IW	0.9526	0.9161	0.9340	0.9902	0.9731	0.9816
Hikvision_DS-IPC-S12P-IWT	0.9255	0.9535	0.9393	0.9753	0.9907	0.9829
Ezviz_Camera_CS-C6CN	0.9251	0.9382	0.9316	0.9676	0.9264	0.9465
Ezviz_Door_CS-DB2C	0.9289	0.9596	0.9440	0.9300	0.9856	0.9570
Huawei_Camera_HQ5	0.9966	0.9968	0.9967	0.9988	0.9992	0.9990
Huawei_Speaker	0.9982	0.9936	0.9959	0.9991	0.9978	0.9985
Huawei_Smart_Scale	0.7975	0.8289	0.8129	0.9663	0.9101	0.9373
Imou-Camera_TP1-2525	0.9979	0.9962	0.9971	0.9986	0.9977	0.9982

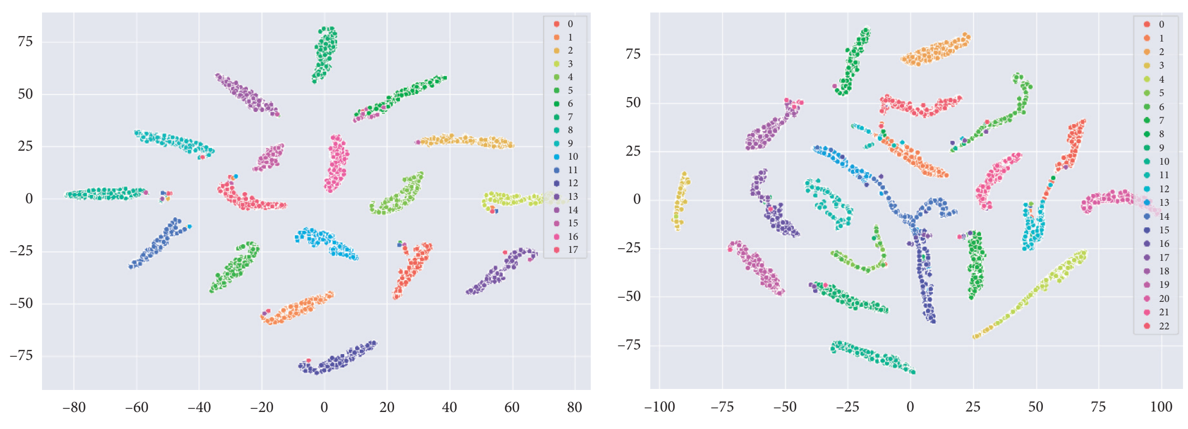
Table 8: Continued.

Method	CBBI w/o FGAN			CBBI		
Imou-Camera_TP7C-E152	0.9894	0.9894	0.9894	0.9977	0.9932	0.9955
Philips_Hue	0.9504	0.9165	0.9332	0.9606	0.9379	0.9491
TP_Link_Camera_IPC42A-4	0.9638	0.9499	0.9568	0.9806	0.9836	0.9821
TP_Link_Camera_IPC43AN-4	0.9581	0.9492	0.9536	0.9881	0.9731	0.9805
TP_Link_Camera_IPC64C-4	0.8520	0.9287	0.8887	0.9360	0.9682	0.9518
TP_Link_WDA6332RE	0.6364	0.4667	0.5385	0.7815	0.8378	0.8087
Xiaomi_Air_Purifier	0.6923	0.3333	0.4500	0.8676	0.8806	0.8741
Xiaomi_Camera_MJSXJ06CM	0.9919	0.9923	0.9921	0.9986	0.9979	0.9983
Xiaomi_Door_LSC_M01	0.9905	0.9881	0.9893	0.9952	0.9967	0.9960
Xiaomi_Hub	0.6857	0.4898	0.5714	0.9975	0.9988	0.9982
Xiaomi_Humidifier	0.8881	0.8975	0.8928	0.9730	0.9689	0.9709
Xiaomi_Soundbox	0.9135	0.9041	0.9088	0.9814	0.9483	0.9646
Macro average performance	0.9033	0.8752	0.8850	0.9651	0.9652	0.9649
Total accuracy		0.9524			0.9726	



(a) (b)

Figure 6: The confusion matrix of the two datasets. (a) The confusion matrix of the UNSW dataset. (b) The confusion matrix of the laboratory dataset.



(a) (b)

Figure 7: Visualization of the final fully connected layer based on t-SNE. (a) Visualization of the UNSW dataset. (b) Visualization of the laboratory dataset.

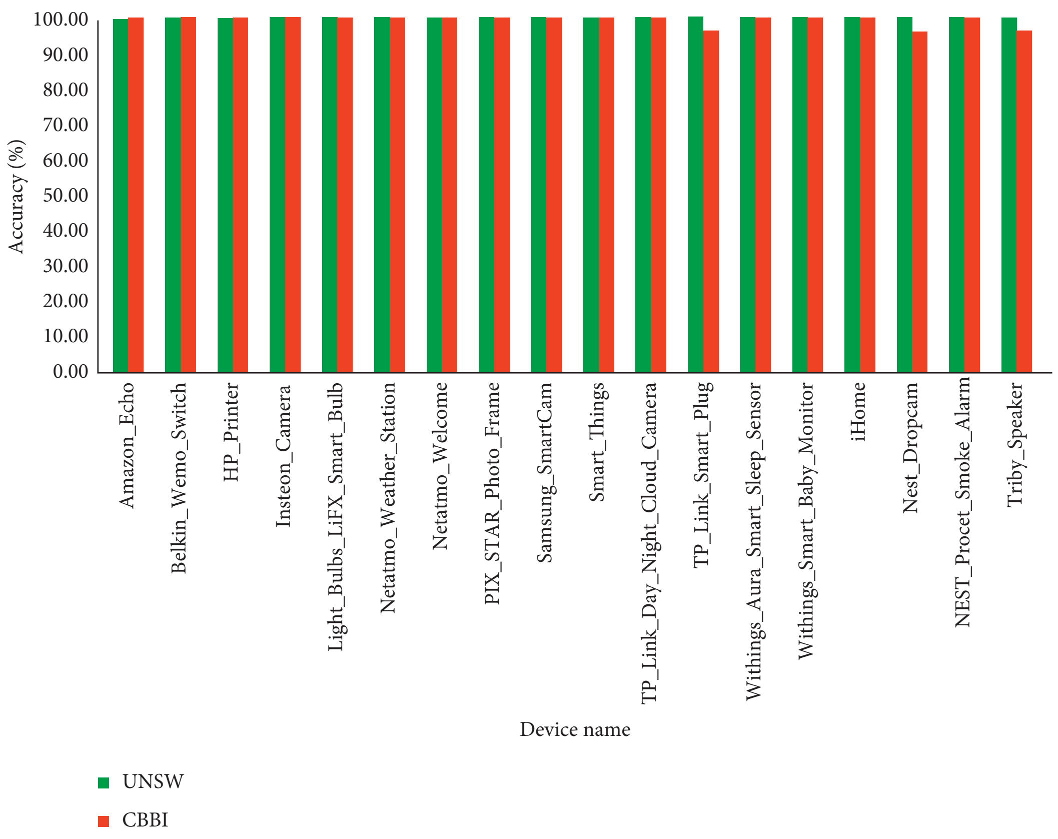


Figure 8: Comparison with related work on the UNSW dataset.

Table 9: Performance comparison of different model combinations.

Method		Accuracy	Precision	Recall	F1-score
CNN	UNSW dataset	0.9464	0.8919	0.8858	0.8872
	Laboratory dataset	0.9302	0.9026	0.8774	0.8877
FGAN + CNN	UNSW dataset	0.9517	0.8990	0.8963	0.8950
	Laboratory dataset	0.9463	0.9081	0.8925	0.8989
BiLSTM	UNSW dataset	0.9465	0.8897	0.8800	0.8812
	Laboratory dataset	0.9382	0.8875	0.8581	0.8664
FGAN + BiLSTM	UNSW dataset	0.9541	0.9059	0.8882	0.8906
	Laboratory dataset	0.9478	0.8900	0.8671	0.8754
CNN + BiLSTM	UNSW dataset	0.9865	0.9325	0.8865	0.9039
	Laboratory dataset	0.9524	0.9033	0.8752	0.8850
FGAN + CNN + BiLSTM (CBBI)	UNSW dataset	0.9983	0.9958	0.9914	0.9935
	Laboratory dataset	0.9726	0.9651	0.9652	0.9649

generated was relatively small, especially for several devices such as Nest_Dropcam, NEST_Procet_Smoke_Alarm, and Triby_Speaker. Our method achieves an accuracy rate similar to that of UNSW [27]. Additionally, CBBI does not need to manually extract features, which increases the timeliness of the device recognition.

We have implemented more comparative experiments, including CNN, FGAN + CNN, BiLSTM, FGAN + BiLSTM, CNN + BiLSTM, and CBBI. The detailed experimental results are shown in Table 9. Each method gives the accuracy, precision, recall, and F1-score values. We can conclude that FGAN and the simultaneous learning of temporal and spatial features can effectively improve the identification accuracy.

In summary, various experimental results show that our method can effectively and accurately identify the IoT devices. Compared with traditional manual feature extraction methods, this method can not only automatically learn the representative features of devices but also has good classification capabilities. On the other hand, the experimental results on the two datasets also show the effectiveness and flexibility of CBBI, which can address the complex and changeable IoT device environment.

6. Conclusions and Future Work

In this paper, we propose an IoT identification method called CBBI. This method uses the spatial and temporal features of the original network traffic generated by the IoT devices, which avoids the overhead and cumbersomeness of feature extraction in the traditional methods and reduces the complexity of the IoT device identification task. CBBI has three modules: data preprocessing, data augmentation FGAN, and Conv-BiLSTM. The main task of the data preprocessing module is to quickly process the raw network traffic generated by the IoT device and convert it into input that can be used in a deep learning model. The data augmentation module FGAN solves the problem of class imbalance in deep learning and further improves the accuracy and generalization ability of the model. The hybrid deep learning model Conv-BiLSTM can learn the spatial and temporal characteristics of the device communication traffic. In this paper, we use a public dataset and a laboratory dataset to verify the effectiveness of CBBI. The experimental results show that CBBI has good classification performance, even for some IoT devices from the same equipment manufacturers. In our future work, we will consider a combination of active and passive IoT device identification schemes to realize the identification of unknown IoT devices. [45].

Data Availability

We used the UNSW dataset, which is a publicly accessed dataset (https://iotanalytics.unsw.edu.au/iottraces). The laboratory dataset used to support the findings of this study is available from the corresponding author upon request.

Disclosure

This paper is an extended version of a conference paper, and the conference name is DSC 2021—IEEE Conference on Dependable and Secure Computing.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key Research and Development Project (2017YFB0801805) and the National Natural Science Foundation of China (62072359 and 62072352).

References

[1] "Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025," https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[2] "Internet of things (IoT) - statistics & facts," https://www.statista.com/statistics/370350/internet-of-things-installed-base-by-category/.

[3] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[4] F. Yin, L. Yang, Y. Wang, and J. Dai, "Iot ETEI: end-to-end iot device identification method," in *Proceedings of the IEEE Conference on Dependable and Secure Computing, DSC 2021*, pp. 1–8, Aizuwakamatsu, Japan, January 2021.

[5] Y. Yue, S. Li, P. Legg, and F. Li, "Deep learning-based security behaviour analysis in iot environments: a survey," *Security and Communication Networks*, vol. 2021, Article ID 8873195, 13 pages, 2021.

[6] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Proceedings of the IEEE Symposium on Security and Privacy, SP, 2016*, pp. 636–654, San Jose, CA, USA, May 2016.

[7] V. Sachidananda, J. Toh, S. Siboni, S. Asaf, and E. Yuval, "POSTER: towards exposing internet of things: a roadmap," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1820–1822, Vienna, Austria, October 2016.

[8] L. Garcia, F. Brasser, M. H. Cintuglu, R. S. Ahmed, A. M. Osama, and A. Z. Saman, "Hey, my malware knows physics! attacking plcs with physical model aware rootkit," in *Proceedings of the 24th Annual Network and Distributed System Security Symposium, NDSS 2017*, San Diego, CA, USA, February 2017.

[9] E. Ronen, A. Shamir, A. O. Weingarten, and C. O'Flynn, "Iot goes nuclear: creating a zigbee chain reaction," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy, SP2017*, pp. 195–212, San Jose, CA, USA, May 2017.

[10] Y. Jia, Y. Xiao, J. Yu, X. Cheng, Z. Liang, and Z. Wan, "A novel graph-based mechanism for identifying traffic vulnerabilities in smart home iot," in *Proceedings of the 2018 IEEE Conference on Computer Communications, INFOCOM*, pp. 1493–1501, Honolulu, HI, USA, April 2018.

[11] S. Soltan, P. Mittal, and H. V. Poor, "Blackiot: iot botnet of high wattage devices can disrupt the power grid," in *Proceedings of the 27th USENIX Security Symposium, USENIX Security 2018*, pp. 15–32, Baltimore, MD, USA, August 2018.

[12] Z. Liu, R. Wang, N. Japkowicz, Y. Cai, D. Tang, and X. Cai, "Mobile app traffic flow feature extraction and selection for improving classification robustness," *Journal of Network and Computer Applications*, vol. 125, pp. 190–208, 2019.

[13] X. Xiao, R. Li, H. T. Zheng, R. Ye, A. KumarSangaiah, and S. Xia, "Novel dynamic multiple classification system for network traffic," *Information Sciences*, vol. 479, pp. 526–541, 2019.

[14] D. L. C. Choong, C. Y. Cho, C. P. Tan, and R. S. Lee, "Identifying unique devices through wireless fingerprinting," in *Proceedings of the First ACM Conference on Wireless Network Security, WISEC 2008*, pp. 46–55, Alexandria, VA, USA, March, 2008.

[15] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," in *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MOBICOM 2008*, pp. 104–115, San Francisco, CA, USA, September 2008.

[16] 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.

[17] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz, "On the reliability of wireless fingerprinting using clock skews," in *Proceedings of the Third ACM Conference on Wireless Network Security, WISEC 2010*, pp. 169–174, Hoboken, NJ, USA, March 2010.

[18] N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng, "Device fingerprinting to enhance wireless security using

nonparametric bayesian method," in *Proceedings of the IEEE INFOCOM*, pp. 1404–1412, Shanghai, China, April 2011.

[19] S. U. Rehman, K. W. Sowerby, and C. Coghill, "Analysis of impersonation attacks on systems using rf fingerprinting and low-end receivers," *Journal of Computer and System Sciences*, vol. 80, no. 3, pp. 591–601, 2014.

[20] H. L. Yuan and A. Q. Hu, "Preamble-based detection of wi-fi transmitter rf fingerprints," *Electronics Letters*, vol. 46, no. 16, pp. 1165–1167, 2010.

[21] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MOBICOM 2008*, pp. 116–127, San Francisco, California, USA, September 2008.

[22] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, "Gtid: a technique for physical device and device type fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 519–532, 2014.

[23] M. Miettinen, S. Marchal, I. Hafeez et al., "Iot SENTINEL: automated device-type identification for security enforcement in iot," in *Proceedings of the 37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017*, pp. 2177–2184, Atlanta, GA, USA, June 2017.

[24] H. Guo and J. S. Heidemann, "Ip-based iot device detection," in *Proceedings of the 2018 Workshop on IoT Security and Privacy, IoT S&P@SIGCOMM 2018*, pp. 36–42, Budapest, Hungary, August 2018.

[25] S. Marchal, M. Miettinen, T. D. Nguyen, A. R. Sadeghi, and N. Asokan, "AuDI: toward autonomous IoT device-type identification using periodic communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

[26] V. Thangavelu, D. M. Divakaran, R. Sairam, S. B. Suman, and G. Mohan, "Deft: a distributed iot fingerprinting technique," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 940–952, 2018.

[27] A. Sivanathan, H. H. Gharakheili, F. Loi et al., "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2018.

[28] L. Yu, T. Liu, Z. Zhou, and Y. Zhu, "WDMTI: wireless device manufacturer and type identification using hierarchical dirichlet process," in *Proceedings of the 15th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2018*, pp. 19–27, Chengdu, China, October 2018.

[29] L. Yu, B. Luo, J. Ma et al., "You are what you broadcast: identification of mobile and iot devices from (public) wifi," in *Proceedings of the 29th USENIX Security Symposium, USENIX Security 2020*, pp. 55–72, August 2020.

[30] L. Deng, Y. Feng, D. Chen, and N. Rishe, "Iotspot: Identifying the iot devices using their anonymous network traffic data," in *Proceedings of the 2019 IEEE Military Communications Conference, MILCOM 2019*, pp. 1–6, Norfolk, VA, USA, November 2019.

[31] M. Antonakakis, T. April, M. Bailey et al., "Understanding the mirai botnet," in *Proceedings of the 26th USENIX Security Symposium, USENIX Security 2017*, pp. 1093–1110, Vancouver, BC, Canada, August 2017.

[32] Z. Durumeric, D. Adrian, A. Mirian et al., "A search engine backed by internet-wide scanning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 542–553, Denver, CO, USA, October 2015.

[33] Shodan, "The search engine for Internet-connected devices," https://www.shodan.io/.

[34] C. Fachkha, E. Bou-Harb, A. Keliris et al., "Internet-scale probing of CPS: inference, characterization and orchestration analysis," in *Proceedings of the 24th Annual Network and Distributed System Security Symposium, NDSS 2017*, San Diego, CA, USA, February 2017.

[35] K. Yang, Q. Li, and L. Sun, "Towards automatic fingerprinting of iot devices in the cyberspace," *Computer Networks*, vol. 148, pp. 318–327, 2019.

[36] Q. Li, X. Feng, L. Zhao, and L. Sun, "A framework for searching internet-wide devices," *IEEE Network*, vol. 31, no. 6, pp. 101–107, 2017.

[37] 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.

[38] "SplitCap tool," https://www.netresec.com/?page=SplitCap.

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[41] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "Lstm: a search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[42] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[44] A. Gulli and S. Pal, *Deep Learning with Keras*, Packt Publishing Ltd, Birmingham, United Kingdom, 2017.

[45] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*, Insecre. Com LLC (US), WA, USA, 2008.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 2610887, 10 pages
https://doi.org/10.1155/2021/2610887

Research Article

Image Speckle Denoising for Securing Internet of Smart Sensors

Wei Ma,[1,2] Zhihui Xin,[1,2] Licun Sun,[1,2] and Jun Zhang[1,2]

[1]*Yunnan Normal University, School of Physics and Electronic Information, Kunming/650500, China*
[2]*Yunnan Normal University, Yunnan Key Lab of Opto-Electronic Information Technology, Kunming/650500, China*

Correspondence should be addressed to Zhihui Xin; xinzhihui.luncky@163.com and Jun Zhang; junzhang@ynnu.edu.cn

Received 19 September 2021; Accepted 28 October 2021; Published 18 November 2021

Academic Editor: Weizhi Meng

How to improve utility performance when securing sensitive data is an important research problem in Internet of smart sensors. In this paper, we study secured image speckle denoising for networked synthetic aperture radar (SAR). Speckle noise of SAR affects image quality and has a great influence on target detection and recognition. MSTAR dataset is often used in image target recognition. In this paper, a subregion-based method is proposed in order to improve the accuracy of target recognition and better retain target information while filtering and denoising the image. The new method applies advanced encryption techniques to protect sensitive data against malicious attack. Firstly, the image is divided into marked areas and unmarked areas through edge extraction and hole filling. Secondly, we use different size windows and filtering methods to filter the image in different areas. The experimental results show that the proposed algorithm has obvious advantages over MR-NLM, SSIM-NLM, Frost, and BM3D filtering in terms of equivalent view number and preserving edge and structure.

1. Introduction

Synthetic aperture radar (SAR) is a system of continuous tracking and monitoring imaging that can transmit and receive electromagnetic waves. As a smart sensor, SAR can provide full-time and full-weather observation for targets and is often used in military and civilian applications. Radar networking is an effective method to improve the radar system in the modern war and civil scenario. In contrast to the single radar system, the radar networking system can trace objects more precisely, is more capable to anti-jamming, and has larger scale in space domain, time domain, and frequency domain. When the radar networking is designed, the security of received data and the stability of the system need to be considered. Cyber security is very important in different netting systems. Therefore, detecting and preventing cyber insider threats are necessary [1–4]. The stealing attack based on machine learning is reviewed in perspectives of three categories of targeted controlled information [5]. When the radar networking system is utilized, the data from single radar and different processing flows should be encrypted and kept secure. Malware and software vulnerability detections based on machine learning and deep neural models are studied to ensure that the Internet system is secure [6–8]. Due to the limitation of the imaging mechanism of radar, the complexity of ground environment usually makes the reflection of the surrounding environment to electromagnetic waves weaken the reflection of the target to electromagnetic waves. Therefore, speckle suppression of SAR image is extremely important for feature extraction of image and target recognition.

At present, MSTAR datasets supported by Defense Advanced Research Projects Agency are widely used for target recognition of the SAR image. Image denoising is needed to improve the efficiency of image recognition. There are two main purposes for image denoising in the MSTAR dataset: one is to smooth out the noise-polluted background area and the other is to retain more texture and detail information in the target area [9]. Commonly used SAR image denoising algorithms are divided into spatial domain filtering [10, 11], transform domain filtering [12–15], and partial differential anisotropic diffusion filtering [16, 17]. Spatial filtering is mainly based on LEE filtering [10] and Frost filtering [11]; in this method, the corresponding pixel values are processed by sliding a fixed-size window and predefined filter coefficients. However, the filtering effect is limited by the window size and has the disadvantage of insufficient filtering. In recent years, transform domain filtering such as wavelet transform [12] or BM3D and it is various improved methods have also achieved good results [13–15], but these methods are easy to introduce interference artificially. Anisotropic diffusion filtering [16, 17] takes an image as a heat field and determines whether to diffuse to the surrounding pixels according to the relationship between the surrounding pixels and the current pixel. However, this algorithm requires multiple iterations and requires a large amount of computation.

Based on bilateral filtering method, the non-local means (NLM) denoising algorithm was developed [18]. This algorithm uses image block similarity instead of traditional single-pixel similarity to construct weights. Making full use of the redundant information in the image, details of the image can be maintained to the greatest extent while denoising [19]. In recent years, scholars have carried out a large number of studies based on NLM filtering [19–25]. Through the SAR image experiment in MSTAR, it can be found that this method can achieve good denoise performance in the target area. However, the denoising performance in the background area and the edge area of the target is poor. NLM filtering effect is also related to the size of the window. Meanwhile, the methods of NLM filtering are impossible to balance the details of preserving details and smoothing noise. Due to the extremely serious noise pollution of the MSTAR data image, the variance-based segmentation algorithm proposed in paper [20] and the cluster-based segmentation algorithm proposed in paper [21] are not suitable for the MSTAR data image.

Each algorithm has its own merits and demerits. This paper proposes a subregion NLM filter method. The image is divided into marked region and unmarked region by extracting target information. Then, we comprehensively consider the noise difference between different regions. An improved NLM filter method is used for the unmarked region. The two regions need to be encrypted through encryption technology. Then, two encryption features are transferred into the next flow to decrypt and denoise with different filter weight, respectively. After this processing, different weighted functions and different sizes of filtering windows are used for NLM filtering in different regions. Experimental results show that the algorithm in this paper can greatly suppress noise while preserving the target details.

2. Proposed Method

2.1. Non-Local Mean Algorithm. The non-local mean algorithm can define the weight by measuring the similarity of the two similar blocks in the large search window when filtering the additive noise image and calculates the pixel value of the target point by weighted average. Its algorithm can be denoted as

$$U(i) = \sum_{j\in\Omega(i)} W(i,j)V(j), \tag{1}$$

where $\Omega(i)$ is the large-scale search window centering on pixel i, $U(i)$ is the pixel after filtering, and $V(j)$ is the pixel at any point in the search window. $W(i,j)$ is the weighted coefficient of pixel j to pixel i in the search window. Weight is defined as

$$W(i,j) = \frac{1}{Z(i)}\exp\left(-\frac{d(i,j)}{h^2}\right), \tag{2}$$

where h controls the rate of decay of the exponential function, and it is usually equal to 10. $Z(i)$ is the normalized factor, which can be expressed as

$$Z(i) = \sum_{j}\exp\left(-\frac{d(i,j)}{h^2}\right), \tag{3}$$

where $d(i,j)$ is the similarity of two pixels in formulas (2) and (3). When calculating $d(i,j)$, the first step is to form similar areas of fixed sizes $N(i)$ and $N(j)$ with the pixels i and j as the center. The similarity is determined by the gray value vectors $v(N_i)$ and $v(N_j)$, which can be measured by Gaussian weighted Euclidean distance which can be expressed as

$$d(i,j) = \left\|v(N_i) - v(N_j)\right\|_{2,\sigma}^{2}, \tag{4}$$

where ${}_{2,\sigma}^{2}$ denotes the L2 norm.

However, due to the special imaging mechanism of SAR image, its noise is a typical multiplicative noise. Assuming that the observed value of a SAR image at position i is $Y(i)$, its real signal is $X(i)$, and the speckle noise is $N(i)$, and then its noise model is

$$Y(i) = N(i) * X(i). \tag{5}$$

Therefore, the non-local mean algorithm is not directly applicable to SAR image denoising. On the contrary, it is suitable only when the multiplicative noise is changed into additive noise after logarithmic processing.

2.2. Subregion NLM Filtering Algorithm. Classical NLM algorithms cannot maintain the texture of the target while smoothing the noise. A new subregion filtering process and weighting function are proposed in this paper. The proposed algorithm flowchart is shown in Figure 1. The original image is divided into two areas, and different areas are processed using different filtering weights. The edge contour forms a closed area after edge extraction and hole filling. The edge contour is used to mark the original image. The original image is divided into marked region and unmarked region. For the selection of the size of search window and similar window, small window should be used to maintain details and large window should be used to smooth noise. So, the small window is used in the marked area and the large

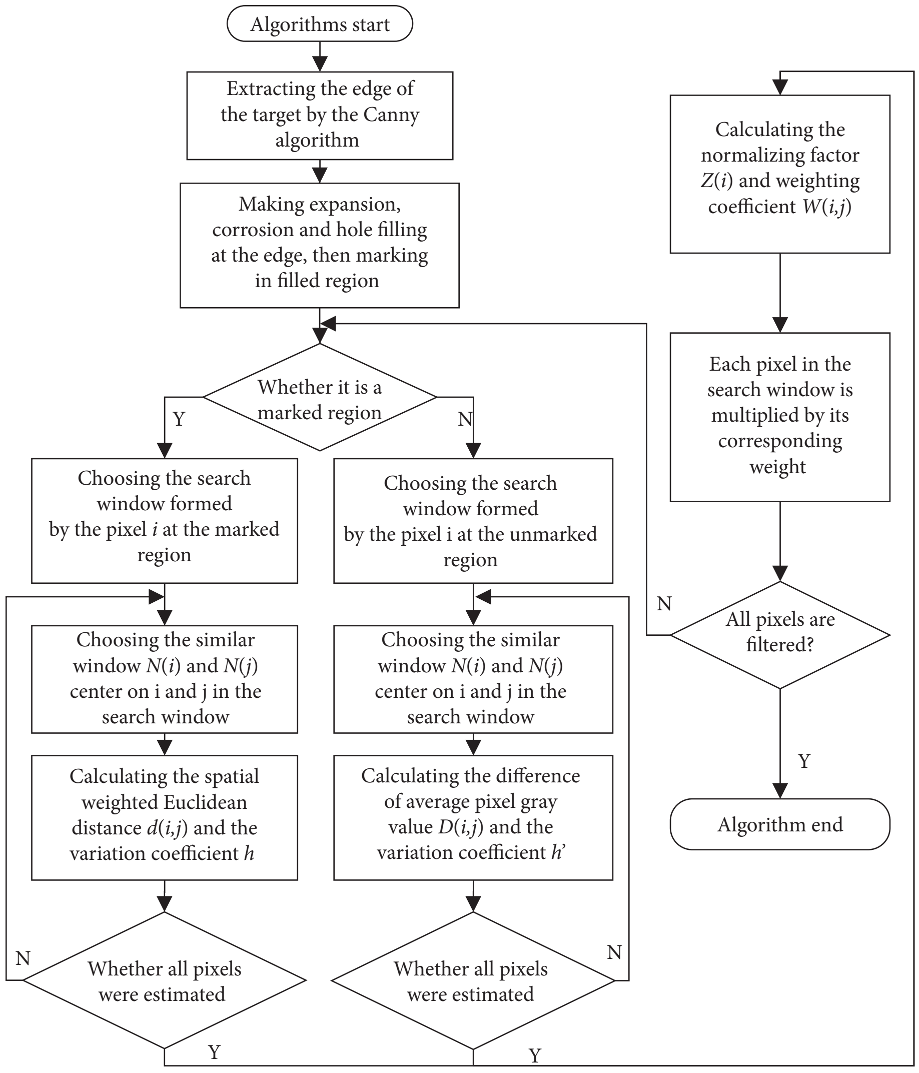

Algorithms start
Extracting the edge of the target by the Canny algorithm
Making expansion, corrosion and hole filling at the edge, then marking in filled region
Whether it is a marked region
Y
N
Choosing the search window formed by the pixel *i* at the marked region
Choosing the search window formed by the pixel i at the unmarked region
Choosing the similar window *N*(*i*) and *N*(*j*) center on i and j in the search window
Choosing the similar window *N*(*i*) and *N*(*j*) center on i and j in the search window
Calculating the spatial weighted Euclidean distance *d*(*i,j*) and the variation coefficient *h*
Calculating the difference of average pixel gray value *D*(*i,j*) and the variation coefficient *h*'
Whether all pixels were estimated
Whether all pixels were estimated
N
N
Y
Y
Calculating the normalizing factor *Z*(*i*) and weighting coefficient *W*(*i,j*)
Each pixel in the search window is multiplied by its corresponding weight
All pixels are filtered?
N
Y
Algorithm end

Figure 1: Algorithm flowchart.

window is used in the unmarked area. The marked area has a large amount of texture information. So, the similarity of windows is measured through Euclidean distance in NLM algorithm for the marked area. At the same time, the similarity of the window is measured by the difference of the average pixel for the unmarked area. When the image is filtered, the variation coefficient is chosen as the filtering parameter. The main steps of the proposed framework are summarized in Algorithm 1.

2.2.1. Edge Extraction. The Canny filtering algorithm is used to extract the image edge in the paper. The image is smoothed through Gaussian filtering. A fixed-size convolution template is used to calculate the gradient of the image. The image is divided into two parts by setting a threshold in the maximum suppression method, and then the two parts of the image are connected [26]. The commonly used convolution template includes the Soble operator, Roberts operator, and Prewitt operator. The gradient value and angle value of a certain point of the image are obtained according to the correlation operator, and the edge of the target is determined according to the method of non-maximum suppression. In this paper, the Soble operator is used to calculate the horizontal and vertical gradients respectively, which can be expressed as

$$\text{Sobel}_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix},$$
$$\text{Sobel}_y = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}. \tag{6}$$

Because the edge curve obtained by Canny filtering algorithm is not necessarily closed, it needs to be expanded and corroded [19]. The expansion process is shown in Figure 2, in which Figure 2(a) shows the original image, assuming that the pixel value of the white area is 0 and the pixel value of the black area is 1. Set a structural element

```
Input: the SAR image $A$
 (1) Extract the edge contour of the central target, carry out expansion, corrosion, and hole filling, form a closed area, and generate a
     binary image $B$;
 (2) Read the image$A$ and $B$size $[m. n]$;
 (3) for $p = 1{:}m$
 (4)   for $q = 1{:}n$
 (5)     if $B\ (i, j) = 255$
 (6)       $A\ (i, j) \in \varphi_u$; $\varphi_u$ represents the marked area;
 (7)     else
 (8)       $A\ (i, j) \in \varphi_n$; $\varphi_n$ represents the unmarked area;
 (9) end; end; end
(10) Then the marked area $\varphi_u$ and unmarked area $\varphi_n$ in the noisy image are obtained;
(11) Read the image $A$ size $[m. n]$;
(12) for $i = 1{:}m$
(13)   for $j = 1{:}n$
(14)     if current position $(i, j)$ belongs to the marker area $\varphi_u$
(15)       $D(i, j) = |\overline{v(N_i) - v(N_j)}|^2$ ;
(16)     else
(17)       $d(i, j) = \|v(N_i) - v(N_j)\|^2_{2,\sigma}$ ;
(18)     Use formulas (2) and (11) to calculate the weights;
(19) end; end; end
Output: the final denoising image
```

Algorithm 1: Secured image speckle denoising for internet of smart sensors.

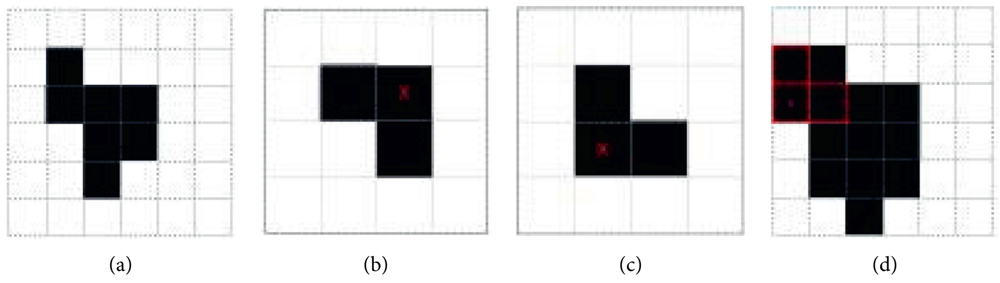


Figure 2: Image expansion process. (a) Original image. (b, c) Structural elements. (d) Expansion image.

(Figure 2(b)) and obtain the mapping (Figure 2(c)) of Figure 2(b) about the origin. The process of expansion is to make Figure 2(c) traverse every pixel in the original image. When Figures 2(c) and 2(a) have an intersection, the center pixel of Figure 2(c) is replaced by the maximum value in the structure. Figure 2(d) is the result of expansion (Figure 2(a)). The mathematical expression of expansion is

$$A \oplus B = \left\{ x \mid (\widehat{B})_X \cap A \neq \varphi \right\}, \tag{7}$$

where A is the original image and B refers to the structural elements.

The corrosion process is the inverse process of expansion. As shown in Figure 3, the aim of which is to make Figure 3(b) traverse every pixel in the original image (Figure 3(a)), when Figure 3(b) is included in Figure 3(a), the central pixel of Figure 3(b) is retained. Figure 3(d) is a corrosion diagram of Figure 3(a). The mathematical expression of corrosion is

$$A \Theta B = \{ x \mid (B)_X \subseteq A \}. \tag{8}$$

2.2.2. Hole Filling. In this algorithm, the edge of the target is extracted by the Canny operator, but the edge is not necessarily continuous. The hole is filled after the expansion and corrosion of the edge. A hole can be understood as a background region surrounded by the boundary connected by foreground pixels. The basis of hole filling is also the expansion of image. After the edge of the target is obtained, the target region is separated from the image through hole filling [27]. As shown in Figure 4, Figure 4(a) is the edge image. First of all, the edge image is inverted to get Figure 4(b) in the process. Then, use an all-white graph with a black spot in the hole (Figure 4(c)) as the initial image and fill it with Figure 4(d). If the expansion result exceeds the size of the hole, use Figure 4(b) to find the intersection of it and limit it to the inside of the hole. Figure 4(e) is the result after filling twice. Assuming that the number of expansion operations is k and the result is X_k, when X_k and X_{k-1} are the same, the filling image (Figure 4(f)) of the hole is obtained. The filling image (Figure 4(f)) is merged with Figure 4(a) to form the final result (Figure 4(g)).

The hole filling formula can be expressed as

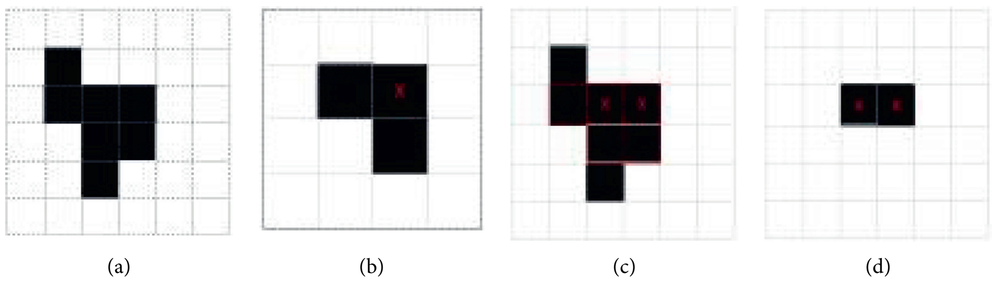


FIGURE 3: Image etching process. (a) Original image. (b) Structural element. (c) Corrosion process. (d) Corrosion image.

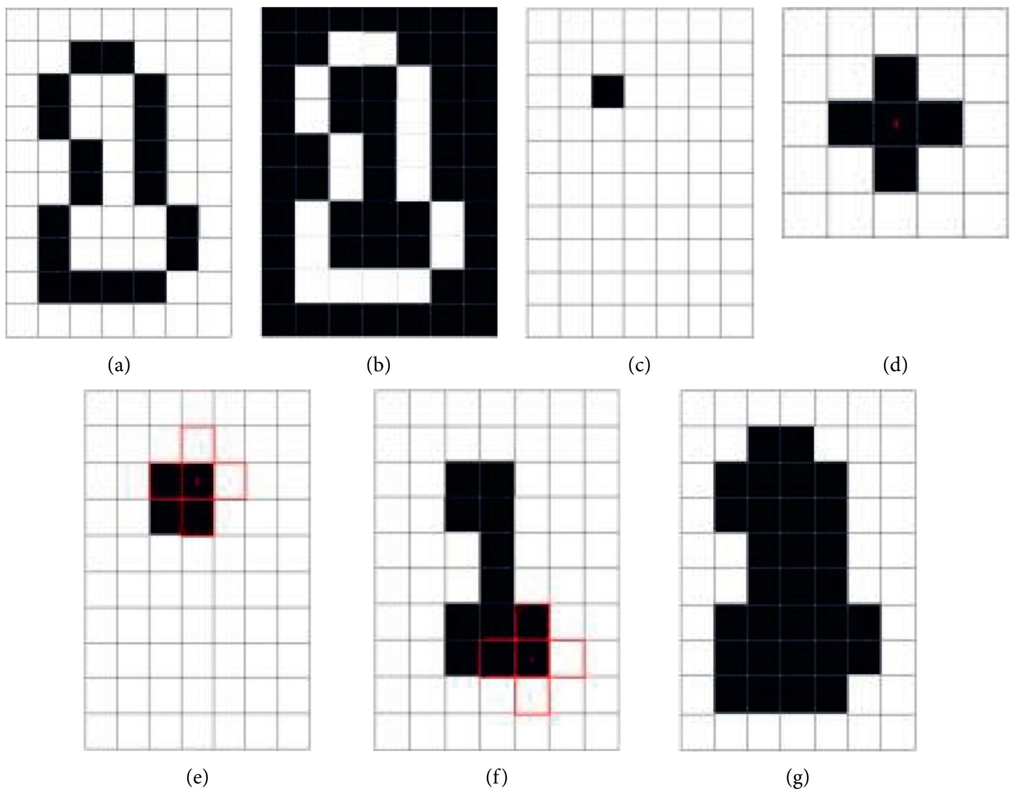


FIGURE 4: Hole filling process. (a) Edge image. (b) Edge image is inverted. (c) The first image of the hole filling. (d) Structural element. (e) The image after two fillings. (f) Filling image of the hole. (g) Final image.

$$\begin{aligned} x_k &= \left(x_{k-1} \oplus B\right) \cap A^c, \\ E &= x_k \cup A^c. \end{aligned} \tag{9}$$

2.2.3. Improved Weight Function. As shown in Figure 5, (a) is the similarity window of the center pixel of the search window, where the white pixel is noise, and (b) and (c) represent two different sliding windows, respectively. $d(a, c) > d(a, b)$ can be obtained according to formula (4), and $w(a, c) < w(a, b)$ can be obtained according to formula (2). But for (a), the white pixel at the center is noise and should be cleared, so the ideal weighting situation should be $w(a, c) < w(a, b)$. Borrowing the idea of neighborhood filtering, the difference between the gray values of the average pixels of similar blocks is used as the similarity function here so as to redefine the weighted function. When the difference

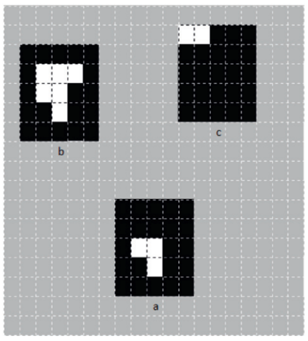


FIGURE 5: Pixel weighted weight analysis.

between the gray values of the two is smaller, a relatively large weight will be obtained. The smoothing effect of the isolated noise points is related to the size of the window. A large window will make the noise smoother better, but the amount of calculation will increase accordingly.

The new similarity function is defined as

$$D(i,j) = \left|\overline{v(N_i)} - \overline{v(N_j)}\right|^2, \tag{10}$$

where $\overline{v(N_i)}$ and $\overline{v(N_j)}$ represent the average gray value of the similar box centering on i and j, respectively, and then the weight calculation formula of the gray scale of the pixel in the center of the image block is defined as

$$W'(i,j) = \frac{1}{Z(i)} \sum\left[\exp\left(-\frac{D(i,j)}{h'^2}\right)\right], \tag{11}$$

where $Z(i)$ is expressed as a normalized factor, $D(i,j)$ denotes the grayscale distance of the mean pixels of similar blocks, and h' is the corresponding filtering parameter.

The selection of filter parameters plays an important role in the effect of filtering, the size of which directly determines the effect of denoising. The filtering parameters used in the traditional NLM filtering process are all constants, larger values are easy to lose details, and smaller values will retain more noise. Variation coefficient can well evaluate the degree of fluctuation of image pixels, so it is introduced into the filtering parameters. The coefficient of variation is defined as

$$CV(i) = \frac{\sigma(N(i))}{\overline{N(i)}}, \tag{12}$$

where $N(i)$ is a similar block with the pixel as the center of i, $\sigma(N(i))$ is the standard deviation of the similar block, and $\overline{N(i)}$ represents the mean value.

2.2.4. NLM Filtering Algorithm Based on Subregion Improved Weights. We can get the image contour information from the Canny filter algorithm, but because the outline is not closed, expansion, corrosion, and hole filling processing are needed. After edge extraction and hole filling, the original image is divided into marked area and unmarked area by marking the original image. The marked area is the central target with less noise, while the unmarked area is the target background with a large number of isolated noise points. To keep the target information and make effective denoising at the same time, this paper conducts the image in different filtering functions. First, we use the spatially weighted Euclidean distance as the similarity function in the marked area, as shown in formula (4), and appropriate filtering parameters are selected for filtering. Meanwhile, the mean difference of similar blocks is used as the similarity function for weighting in the unmarked region, as shown in equations (10) and (11), and appropriate filtering parameters are selected. According to the characteristics of MSTAR image, the new attenuation factor is defined as

$$\begin{aligned} h &= \beta * CV, \\ h' &= \alpha * CV, \end{aligned} \tag{13}$$

where h and h' are the filter parameters in the marked area and the unmarked area, respectively, and β and α are constants.

3. Experiments and Results

3.1. Experiment Settings. Experimental data were obtained from the measured SAR ground stationary target data. They were published by the MSTAR project supported by DARPA. The radar operates in the X band, and polarization mode is HH polarization. The resolution is $0.3\,\text{m}*0.3\,\text{m}$. The pixel size is $100*100$. There are 7 types of ground targets in 3 categories.

The images of armored vehicles (SAR1) and tanks (SAR2) in the MSTAR dataset are used for verification. We use Frost algorithm [11], SSIM-NLM method [22], MR-NLM method [24], and block-matching 3D (BM3D) algorithm [15] to compare with the proposed algorithm to verify its superiority. The experiment adopted Intel (R) Core (TM) i7-10700F CPU@ 2.90Ghz. During the experiment, the filter window size of MR-NLM was 1 and 2, and the filter parameters were $1/(1.5 * CV)$. The size of the filter window in the SSIM-NLM algorithm is 1 and 2, and the filter parameter is 22σ, where σ is the standard deviation obtained from paper [28] and the number of sights L in the BM3D algorithm is set to 3. In the algorithm of this paper, the search window radius in the marked area is 2, the similar window radius is 1, and the constant α is 0.25. In order to verify that the weighted algorithm proposed in this paper is more beneficial to smooth the noise, two groups of experiments are carried out. In the first group, the radius of the search window in the unmarked region is 5, and the radius of the similarity window is 2. In the second group, the radius of the search window in the unmarked region is 10, and the radius of the similarity window is 3. In the experiments, the constant β is 0.2, and the threshold of Canny edge detection is 0.8.

3.2. Denoising Quality Evaluation Index. Subjective denoising quality evaluation uses naked eyes to observe the denoising degree of images, while objective quality evaluation uses equivalent number of looks, edge preserve index, and structure similarity index to evaluate.

3.2.1. Equivalent Number of Looks (ENL). The equivalent number is an indicator used to view the smoothing effect of the image, and its definition is as follows:

$$\text{ENL} = \frac{\mu^2}{\sigma^2}, \tag{14}$$

where μ is the mean value and σ^2 is the variance. The larger the equivalent number, the better the smoothing effect of the image. In this paper, the equivalent number is calculated only in the specific area.

3.2.2. Edge Preservation Index (EPI). The edge preservation index is used to evaluate the degree of image edge preservation, and its definition is as follows:

$$\text{EPI} = \frac{\sum(|U(i,j) - U(i+1,j)| + |U(i,j) - U(i,j+1)|)}{\sum(|V(i,j) - V(i+1,j)| + |V(i,j) - V(i,j+1)|)}, \tag{15}$$

where $U(i,j)$ is the denoised image and $V(i,j)$ is the original image. The larger the edge preservation index is, the better the effect will be. In this paper, the edge preservation index is calculated only for the marked area, and the unmarked area is the noise area, which has no such index.

3.2.3. Structure Similarity Index (SSIM). The structural similarity index is used to measure the retention degree of image structure before and after denoising. The definition formula is as follows:

$$\mathrm{SSIM}(i, j) = \frac{\left(2\mu_i\mu_j + C_1\right)\left(2\sigma_{ij} + C_2\right)}{\left(\mu_i^2 + \mu_j^2 + C_1\right)\left(\sigma_i^2 + \sigma_j^2 + C_2\right)}, \tag{16}$$

where μ is the mean value, σ is standard deviation, and subscripts i and j show the similarity window, respectively, with their centers i and j. The larger the structure similarity index is, the better the effect will be. In this paper, the structure similarity index is calculated only for the marked area, and the unmarked area is the noise area, which has no such index.

3.3. Experimental Results. We apply the Frost algorithm [11], BM3D algorithm [15], SSIM-NLM algorithm [22], MR-NLM algorithm [24], and the algorithm in this paper to the SAR images of armored vehicles and tanks, respectively. Then, images and related parameters of the obtained armored vehicle are shown in Figure 6 and Table 1. The images and related parameters of the obtained tank are shown in Figure 7 and Table 2. Table 3 presents the results of running time of different algorithms in two groups of experiments, and it can be used to analyze the efficiency of the algorithm.

Armored vehicles have less edge information, while tanks have more edge information, so the robustness of the algorithm in this paper is verified. In this paper, a fixed area is selected in the image and marked by a white box in the original image, and the equivalent number of each algorithm is calculated in this fixed area. Then, the search window radius of 5 and similar block radius of 2 and the search window radius of 10 and similar block radius of 3 were defined for the unmarked area, respectively, in order to analyze the effectiveness of the new weighting function in smoothing noise.

As shown in Figure 6, the SSIM-NLM algorithm and the proposed algorithm have the best effect on noise smoothing, but the SSIM-NLM algorithm causes the image to be blurry. The contour information and texture information of the center target from Figure 6(c) are all lost. The EPI index of SSIM-NLM algorithm from Table 1 is less than 0.5, while that of the proposed method is 0.8625. EPI of the Frost algorithm from Table 1 is only 0.64, so the edge of the target is not well kept. The noise smoothing effect of Frost algorithm from Figure 6(d) is not obvious, though its ENL index from Table 1 is 6.7322. The detail texture and edge contour information of the BM3D algorithm are better preserved, but the noise suppression effect is the worst. The MR-NML algorithm has made some progress in noise suppression compared to the BM3D algorithm. According to Table 1, MR-NML algorithm and BM3D algorithm have good effect on target preservation, but the highest ENL index is only 4.2787. Compared with the proposed algorithm, the proposed algorithm has obvious advantages in noise suppression. From the analysis of the two denoising effect diagrams in this paper, when the window size is set to 5 and 2, respectively, the noise is not smoothed completely, and there are still some white patches. When the window is set to 10 and 3, the white patches are also smoothed out, which proves that the weighting function proposed in this paper has obvious advantages in noise smoothing. Because the same parameters and algorithms are used in the target area, the relevant parameters of the target area have not changed. To sum up, the algorithm in this paper has the optimal effect on noise smoothing, target contour, and detail preservation.

As shown in Figure 7, the central target of the original image includes a square body area and a protruding barrel area, especially an isolated white area directly below the body, which represents the vehicle wheel of the tank. The smoothing effect of the SSIM-NLM algorithm is too serious. While smoothing the noise, it also oversmooths the target. So, the target information is almost completely lost. The images of the vehicle wheel area the barrel area are completely erased from Figure 7(c). The denoising image does not retain any detailed texture information of the target, so its EPI and SSIM are the lowest among various denoising algorithms. The Frost algorithm makes the detailed information of the target lost from Figure 7(d), so EPI of the algorithm is relatively low. The denoising effect of the background of Frost algorithm is not ideal though ENL is 7.2377 in Table 2. The MR-NLM algorithm is excessively smooth in the target area, resulting in a decrease in the SSIM index, and its ENL index is the lowest. Compared with the MR-NLM algorithm, the BM3D algorithm has made certain progress in noise suppression and target retention. However, the BM3D algorithm with the highest EPI index has also reduced by 10% compared with EPI in this paper. The two denoising images in this paper indicate that window will smooth the noise to a greater extent without affecting the center target. The algorithm in this paper can control the ENL index by adjusting the window while maintaining a high level of EPI and SSIM indexes. To sum up, the algorithm in this paper has the optimal effect on noise smoothing, target contour, and detail preservation. The window size can be chosen to obtain an ideal denoising image using the proposed algorithm to recognize the target better.

Table 3 records the running times of the various algorithms in Figures 6 and 7. Since the window size of the algorithm in this paper is set to 2 and 5, the MR-NLM algorithm and the SSIM-NLM algorithm also adopt the same size window as that in this paper. As can be seen from this table, Frost algorithm has the shortest running time. BM3D algorithm is a block method of 3-dimension algorithm based on the similarity between image blocks; although it has shorter running time, the denoising is not good. Among the various improved algorithms of NLM, when MR-NLM adopts a smaller window, the computation time will be shorter, while a larger window will make the computation time longer. When a window of the same size as the one in this paper is used, the running time difference between the algorithm in this paper and the MR-NLM algorithm is less, but the denoising effect in this paper is

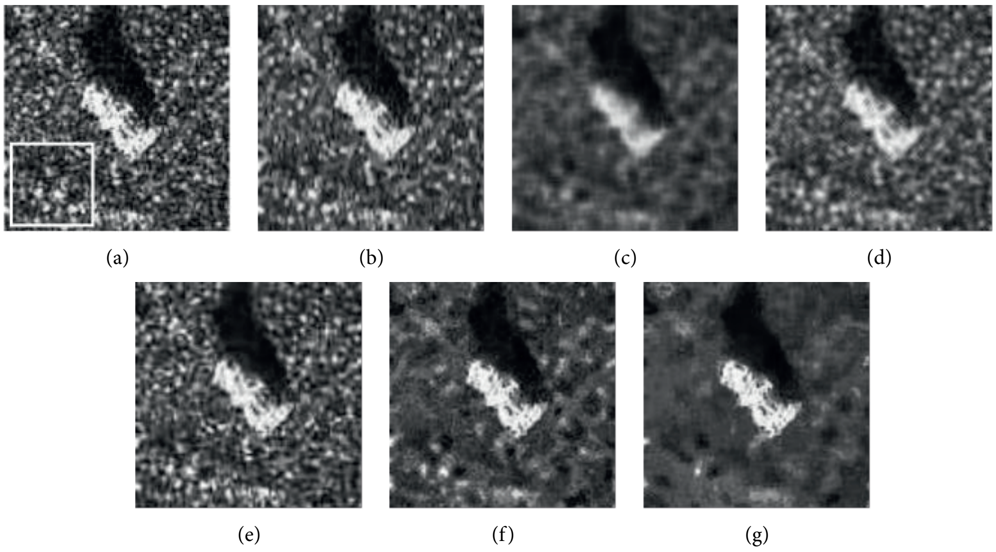


Figure 6: SAR images of armored vehicle. (a) Original. (b) MR-NLM. (c) SSIM-NLM. (d) Frost. (e) BM3D. (f) This paper (search window radius and the similarity window radius are set to 5 and 2). (g) This paper (search window radius and the similarity window radius are set to 10 and 3).

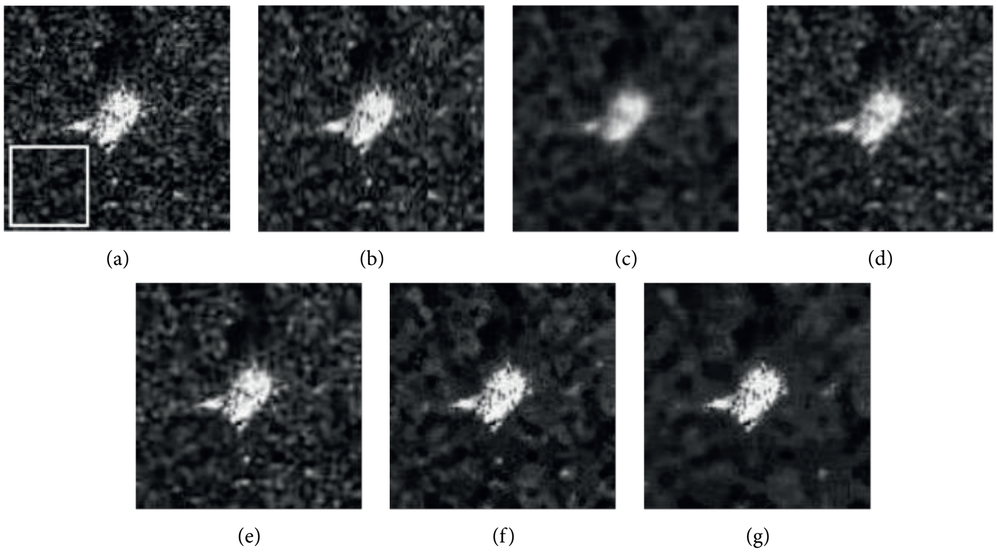


Figure 7: SAR images of tank. (a) Original. (b) MR-NLM. (c) SSIM-NLM. (d) Frost. (e) BM3D. (f) This paper (search window radius and the similarity window radius are set to 5 and 2). (g) This paper (search window radius and the similarity window radius are set to 10 and 3).

Table 1: Performance comparison of several algorithms in Figure 6.

Algorithms	EPI	SSIM	ENL
MR-NLM	0.7134	0.9799	4.2787
SSIM-NLM	0.4646	0.9546	13.7213
Frost	0.6442	0.9740	6.7322
BM3D	0.7955	0.9995	3.6986
This paper (2/5)	0.8625	0.9908	6.0072
This paper (3/10)	0.8625	0.9908	16.1660

better. SSIM-NLM algorithm introduces the structural similarity index, and the increase of parameters in the operation process leads to the increase of operation time and low efficiency. Since the NLM algorithm itself has a high amount of computation, how to improve the computing efficiency of the algorithm will be the focus of future research.

Table 2: Performance comparison of several algorithms in Figure 7.

Algorithms	EPI	SSIM	ENL
MR-NLM	0.7169	0.9827	5.3808
SSIM-NLM	0.4147	0.9504	14.1114
Frost	0.6153	0.9741	7.2377
BM3D	0.7676	0.9994	5.9258
This paper (2/5)	0.8679	0.9918	7.3157
This paper (3/10)	0.8679	0.9918	13.6220

Table 3: Efficiency comparison of several algorithms in Figures 6 and 7.

	Frost (s)	BM3D (*s*)	MR-NLM		SSIM-NLM		This paper
			(1/2) (*s*)	(2/5) (*s*)	(1/2) (*s*)	(2/5) (*s*)	(2/5) (*s*)
Figure 6	0.3	4	74	129	1098	6389	118
Figure 7	0.3	3	72	129	1241	6369	117

This paper adopts different weighting methods in different regions through regional processing. From the analysis of SAR denoising images and relevant indicators of the three targets, the proposed method can achieve good denoising performance compared with the other methods. The texture information of the image is retained well and the noise is thoroughly smoothed. ENL, EPI, and SSIM of the proposed algorithm have great improvement compared with other algorithms.

4. Conclusions

In this paper, a new segmentation filtering algorithm based on edge extraction was proposed, in which edge extraction is applied to SAR image segmentation algorithm. The edge of the target is first determined by edge extraction, and then the image is divided into different areas by expansion corrosion and hole filling. The separated regions are both encrypted before they are transferred to next procedure for security. In the process of filtering, each area is decrypted and filtered separately. Meanwhile, a new NLM filter weighting algorithm is proposed for the background areas with serious noise. The new segmentation filtering algorithm can preserve the characteristics of central target and smooth the background noise well. Performance indexes such as ENL, EPI, and SSIM are obtained to evaluate the algorithm compared with existing methods. Finally, experimental results and simulation data demonstrate the effectiveness of the proposed algorithm.

Data Availability

The MSTAR data used to support the findings of this study have been deposited in the following website: https://download.csdn.net/download/a1367666195/12302537?utm_source=iteye_new.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (61801419) and Natural Science Foundation of Yunnan Province (2019FD114).

References

[1] R. Coulter, Q.-L. Han, L. Pan, Z. Jun, and X. Yang, “Data driven cyber security in perspective - intelligent traffic analysis,” *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3081–3093, 2020.

[2] N. Sun, J. Zhang, P. Rimba, Y. Z. Leo, G. Shang, and X. Yang, “Data-driven cybersecurity incident prediction: a survey,” *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1744–1772, 2019.

[3] L. Liu, O. D. Vel, Q. L. Han, Z. Jun, and X. Yang, “Detecting and preventing cyber insider threats: a survey,” *IEEE Communications Surveys and Tutorials*, vol. 20, no. 2, pp. 1397–1417, 2018.

[4] M. H. Wang, T. Q. Zhu, T. Zhang, Z. Wanlei, S. Yu, and Z. Jun, “Security and privacy in 6G networks: new areas and new challenges,” *Digital Communications and Networks*, vol. 6, no. 3, pp. 281–291, 2020.

[5] Y. T. Miao, C. Chen, L. Pan, Q. H. Long, Z. Jun, and X. Yang, “Machine learning based cyber attacks targeting on controlled information: a survey,” *ACM Computing Surveys*, vol. 54, no. 7, 2021.

[6] J. Y. Qiu, J. Zhang, L. Pan, L. Wei, N. Surya, and X. Yang, “A survey of android malware detection with deep neural models,” *ACM Computing Surveys*, vol. 53, no. 6, pp. 1–36, 2021.

[7] X. Chen, C. Li, D. Wang et al., “Android HIV: a study of repackaging malware for evading machine-learning detection,” *IEEE Transactions on Information Forensics and Security*, vol. 15, no. 1, pp. 987–1001, 2020.

[8] G. J. Lin, S. Wen, Q. L. Han, J. Zhang, and Y. Xiang, “Software vulnerability detection using deep neural networks: a survey,” *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1825–1848, 2020.

[9] C. Liu, R. Szeliski, S. B. Kang, C. L. Zitnick, and W. T. Freeman, “Automatic estimation and removal of noise

from a single image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 299–314, 2008.

[10] J. S. Lee, "Digital image enhancement and noise filtering by using local statistics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 2, pp. 165–168, 1980.

[11] Y. Pan, Y. Meng, and L. Zhu, "SAR image despeckling method based on improved Frost filtering," *Signal Image and Video Processing*, vol. 15, pp. 1–8, 2020.

[12] X. Q. Yang, Z. H. Jia, J. Yang, and K. Nikola, "Change detection of optical remote sensing image disturbed by thin cloud using wavelet coefficient substitution algorithm," *Sensors*, vol. 19, no. 9, pp. 1972–1986, 2019.

[13] M. Lebrun, "An analysis and implementation of the BM3D image denoising method," *Image Processing On Line*, vol. 2, no. 25, pp. 175–213, 2012.

[14] M. M. Hasan, *Adaptive Edge-Guided Block-Matching and 3D Filtering (BM3D) Image Denoising Algorithm*, University of Western Ontario, London, United Kingdom, 2014.

[15] Y. Makinen, L. Azzari, and A. Foi, "Collaborative filtering of correlated noise: exact transform-domain variance for improved shrinkage and patch matching," *IEEE Transactions on Image Processing*, vol. 29, pp. 8339–8354, 2020.

[16] Y. J. Yu and S. T. Acton, "Speckle reducing anisotropic diffusion," *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, vol. 11, no. 11, pp. 1260–1270, 2002.

[17] E. Cuevas, H. Becerra, and A. Luque, "Anisotropic diffusion filtering through multi-objective optimization," *Mathematics and Computers in Simulation*, vol. 183, no. 2, pp. 1410–1429, 2021.

[18] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 60–65, San Diego, CA, USA, June 2005.

[19] S. Y. Chen and X. J. Li, "SAR image despeckling based on adaptive non-local means," *Systems Engineering and Electronics*, vol. 39, no. 12, pp. 2683–2690, 2017.

[20] S. J. Liu, G. Q. Wu, X. Z. Zhang, T. Yang, and Y. M. Li, "SAR image despeckling via the classification-based non-local clustering," *Systems Engineering and Electronics*, vol. 38, no. 3, pp. 551–556, 2016.

[21] J. M. Li, L. Zhu, B. Zhang, and Y. Pan, "A non-localmeans speckle suppression algorithm with restrained search window," *Journal of Xi'an Jiaotong University*, vol. 54, no. 10, pp. 54–62, 2020.

[22] Y. J. Wang, *The Study of Image Denoising Methods Based on the Non-local Means*, Harbin University of Science and Technology, Harbin, China, 2019.

[23] B. G. Kim, S. H. Kang, R. P. Chan, H. W. Jeong, and Y. Lee, "Noise level and similarity analysis for computed tomographic thoracic image with fast non-local means denoising algorithm," *Applied Sciences*, vol. 10, no. 21, 2020.

[24] L. Zhu, F. F. Cai, Y. N. Wang, and L. Y. Guo, "A non-local means filtering algorithm for despeckling of SAR images," *Journal of Xi'an Jiaotong University*, vol. 52, no. 4, pp. 98–104, 2018.

[25] D. Devapal, S. S. Kumar, and C. Jojy, "A novel approach of despeckling SAR images using nonlocal means filtering," *Journal of the Indian Society of Remote Sensing*, vol. 45, no. 3, pp. 443–450, 2016.

[26] C. Huang, W. Jin, Q. Xu, Z. Q. Liu, and Z. L. Xu, "Sub-pixel edge detection algorithm based on canny–zernike moment method," *Journal of Circuits, Systems, and Computers*, vol. 29, no. 15, pp. 1–18, 2020.

[27] X. Liu and X. Ji, "Weld pool image processing and feature extraction based on the vision of the co2 welding,"vol. 355, pp. 625–633, in *Proceedings of the 4th International Conference on Computer Engineering and Networks*, vol. 355, Springer International Publishing, Harbin, China, December 2015.

[28] A. F. Santiago, V. S. F. Gonzalo, M. F. Marcos, and A. L. Carlos, "Automatic noise estimation in images using local statistics. Additive and multiplicative cases," *Image and Vision Computing*, vol. 27, no. 6, pp. 756–770, 2009.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 6234409, 10 pages
https://doi.org/10.1155/2021/6234409

Research Article

EPCT: An Efficient Privacy-Preserving and Collusion-Resisting Top-k Query Processing in WSNs

Qian Zhou,[1] Hua Dai,[1,2] Jianguo Zhou,[1] Rongqi Qi,[1] Geng Yang,[1,2] and Xun Yi[3]

[1]*Nanjing University of Post and Telecommunication, Nanjing 210023, China*
[2]*Jiangsu Security and Intelligent Processing Lab of Big Data, Nanjing 210023, China*
[3]*Royal Melbourne Institute of Technology University, Melbourne 3001, Australia*

Correspondence should be addressed to Qian Zhou; zhouqian@njupt.edu.cn

Received 8 September 2021; Revised 28 September 2021; Accepted 7 October 2021; Published 15 November 2021

Academic Editor: Weizhi Meng

Data privacy threat arises during providing top-k query processing in the wireless sensor networks. This article presents an efficient privacy-preserving and collusion-resisting top-k(EPCT) query processing protocol. A minimized candidate encrypted dataset determination model is first designed, which is the foundation of EPCT. The model guides the idea of query processing and guarantees the correctness of the protocol. The symmetric encryption with different private key in each sensor is deployed to protect the privacy of sensory data even a few sensors in the networks have been colluding with adversaries. Based on the above model and security setting, two phases of interactions between the interested sensors and the sink are designed to implement the secure query processing protocol. The security analysis shows that the proposed protocol is capable of providing secure top-k queries in the manner of privacy protection and anticollusion, whereas the experimental result indicates that the protocol outperforms the existing works on communication overhead.

1. Introduction

Wireless sensor networks (WSNs), as one of the important technologies in the Internet of Things (IoT), have been widely deployed to provide practical solutions in various applications, such as environment monitoring, military target sensing, and smart home application. Meanwhile, data privacy leakage in WSNs is becoming the main obstruction, which slows down its further development. For example, in the scenario of a smart home application, videos or pictures collected by wireless IP-cameras could be eavesdropped for illegal profit. As a result, privacy protection on sensitive data is a critical issue that must be addressed in WSNs.

In WSNs, the top-k query is one of the critical operations in data aggregation for sensor monitoring process. The top-k query requests the k lowest or highest data items collected from IoT sensors in WSNs. For example, "collecting the 10 lowest humidity data in forest area A-Z in last 2 hours" is an example of top-k query, which can be performed for fire monitoring. Our aim of this work is to design a secure top-k query approach with privacy-preserving and collusion-resisting manners.

This article presents an efficient privacy-preserving and collusion-resisting top-k query processing protocol (EPCT) in WSNs. We first propose a minimized candidate encrypted dataset determination model, which is the foundation of our proposed protocol. It guides the idea of query processing and guarantees the correctness of the protocol. There are two phases of interactions between the queried sensors and the sink in EPCT. In the first phase, when the queried sensors receive a top-k query from the sink, they first use their own private keys to encode the maximum of the collected data in the interested time slot, respectively, and then, they submit the encrypted data to the sink. In the second phase, the sink decrypts the received ciphertext and calculates the candidate sensors; after that, it unicastly informs the candidate sensors to submit the rest candidate data. Once the sink obtains enough data from the candidate sensors, the final result of the query is determined. The security analysis and performance evaluation indicate that the proposed approach

EPCT has the ability of protecting data privacy and performing efficiently in transmission overhead.

The main contributions of this article are listed as follows:

(i) We present a minimized candidate encrypted dataset determination model, which is the foundation of our proposed scheme. It guides the idea of query processing and guarantees the correctness of the protocol.

(ii) We present a novel privacy-preserving and collusion-resisting top-k query processing protocol, which consists of two phases of secure interactions between the queried nodes and the sink. We also analyse the correctness, security, and transmission overhead of the proposed method.

(iii) We perform evaluations on the transmission overhead of the proposed protocol and the existing works. The experimental result shows the advantages of the proposed scheme in transmission overhead.

The remainder of this article is organized as follows. Section 2 discusses the related work. Section 3 introduces the network model, query model, threat model, and the problem description. Section 4 proposes the minimized candidate encrypted dataset determination model. Section 5 presents the top-k query processing protocol and the analysis of this protocol. Section 6 presents the performance evaluation of query protocols on communication cost, and Section 7 gives a conclusion of this article.

2. Related Work

Secure data queries (such as top-k query, range query, and MAX/MIN query) are critical operations for sensor monitoring and data collection in security-sensitive environment. There are a lot of works [1–24] focusing confidentiality, integrity, and completeness when performing data queries.

Kui et al. [3] utilize the pairwise-key and order-preserving symmetric encryption and the together to protect the privacy of data in top-k queries in two-tiered WSNs. Peng et al. [4] encoded both sensory data and top-k query commands, and storage nodes are designed to be able to correctly perform top-k queries over those encoded data. Li et al. [6] use pseudorandom hash function with bloom filter and partition algorithm to protect data privacy and integrity for top-k queries, respectively. Tsou et al. [7] constructed a layered authentication tree by an order-preserving symmetric encryption and used it to verify the completeness of query results. Zhang et al. [12] designed a renormalized arithmetic coding method such that storage nodes can calculate exact top-k query results without knowing real values of data, and they proposed a verification scheme to detect compromised storage nodes. Peng et al. [13] encoded top-k queries by threshold-based scheme and proposed a secure protocol that storage nodes can calculate query results over encrypted sensory data. Xingpo et al. [14] proposed secure top-k query protocol with privacy and integrity preservation by deploying the secure data preprocessing in sensor nodes. Wu and Wang [17] bound the collected sensory data with the corresponding locations to achieve secure top-k query processing on hybrid sensory data. Liu et al. [18] proposed a verifiable top-k query protocol on two-tiered mobile sensor network, which adopts the distinct symmetric data encryption and maps real nodes into virtual nodes. These methods are designed for two-tiered WSNs, which adopt resource-rich storage nodes in traditional multihop WSNs. The different network architecture makes them not suitable for addressing secure top-k queries in traditional multihop WSNs.

In traditional multihop WSNs, the earlier studies [19, 25–29] proposed various top-k query schemes but without concerning any security issues. Huang et al. [30] designed a privacy-protection top-k query algorithm using a filter and a data distribution table. The algorithm adopts conic section function to protect the privacy of the sensory data. But, the algorithm is vulnerable when collusion attacks happen. It is because all sensor nodes share the same secure keys and functions. If a sensor node colludes with adversaries, these secure keys and functions will be disclosed, and the adversaries could obtain the private data of other innocent sensors. In our previous work [31], we gave the first solution providing the privacy-protecting and anticollusion top-k query processing scheme in wireless sensor networks. It adopts the bloom filter and HMAC when performing interactions between nodes and the sink to achieve secure top-k query processing. However, there is some space for transmission overhead saving because of the redundant data submission and the false positive of bloom filter. This article presents an efficient and secure top-k query processing protocol, which can address the above problems.

Additionally, some previous studies have focused on the privacy-preserving range queries [8, 11, 32] and MAX/MIN queries [15, 16] in WSNs. Because the query types are different, the ideas of these works cannot be applied to achieve the secure top-k queries in WSNs.

3. Problem Description

3.1. Network Model. The architecture adopted is shown in Figure 1. The network routing topology is structured as a tree, which is following TAG protocol [33]. Assuming that in our scenario, n sensors $S = \{s_1, s_2, \ldots, s_n\}$ are deployed and a sink. Sensor nodes are sensory devices with limited resources in energy, storage, and computation. They are in charge of collecting data items from their neighboring areas and then submitting the collected data to the sink through the tree route. The sink is a resourceful device, which executes query commands from users and returns query results to users. When receiving a query command, the sink cooperates with those queried sensors in S to process queries according to predeployed protocols. After the sink obtains the query result, it returns the result to the upper-level users.

3.2. Top-k Query Model. A top-k query is a data aggregation operation to get k highest or lowest sensory data from queried sensors. It is denoted as a triple $\text{Query}_t = (t, S, k)$

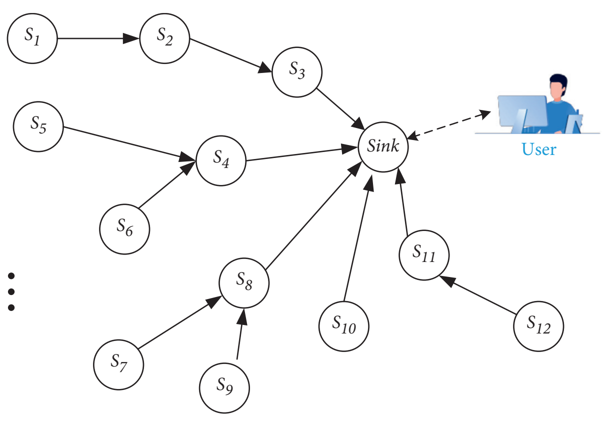


FIGURE 1: A example of tree routing topology.

where t is the queried time slot identity, S is the set of interested sensors, and k is the number of interested data items. For example, $(t, \{s_1, s_2, \ldots, s_{12}\}, 3)$ is a top-3 query to obtain the 3 highest or lowest data items during sensors $\{s_1, s_2, \ldots, s_{12}\}$ in the time slot t.

Each sensor $s_i \in S$ is assumed to collect N data items in a time slot, which is denoted as $D_i = \{d_{i,1}, d_{i,2}, \ldots, d_{i,N}\}$, and each data item collected by a sensor is assumed to have an unique score. The uniqueness of collected data items can be achieved by integrating the data collecting time and the sensor identity into the data item score calculation. It ensures the uniqueness and correctness of a top-k query result.

3.3. Threat Model. The honest-but-curious threat model [9] is adopted in this article. The sink is trustful while sensors could collude with adversaries to leak out their collected or forwarded data. But the sensors that has been attacked still perform the pre-deployed protocols and cooperate with other innocent (noncompromised) sensors to process query commands. We have to note that the innocent sensors are the majority in WSNs; otherwise, the network will be useless.

The goal of the proposed secure top-k query protocol is described as follows:

(1) A sensor only owns the data collected by itself, and the data can be shared with the sink. It has no idea of the data collected by other sensors even when they are colluding with the adversaries.

(2) Query results can only be obtained by the sink, but the adversaries have no idea of them even when there are a few compromised sensors colluding with the adversaries.

(3) The k data items obtained by the sink are the k highest or lowest data items collected by the queried sensors, which means that the query result is correct.

Because sensors have limited energy, the network lifetime is usually determined by the energy consumption of the sensors. Reference [33] shows that sensors consume most energy in data transmission. Thus, the transmission overhead of network is an important metric for performance evaluation. We will perform the evaluation on this metric in Section 6.

4. Minimized Candidate Encrypted Dataset Determination Model

Based on the idea making, the proposed protocol efficient in transmission overhead. We propose the minimized candidate encrypted dataset determination model in this session.

4.1. Minimized Candidate Sensor Set. Let $\text{Query}_t = (t, S, k)$ be a query command, and each sensor $s_i \in S$ collects N data items in a time slot, the set of collected data of all sensors in S is $D = \{d_{i,j} | s_i \in S \wedge 1 \leq j \leq N\}$.

Definition 1. For a top-k query, the query result R_t is a dataset having the k largest data items of D. $L(R_t)$ is denoted as the lower bound of R_t, which is the minimum of R_t.

Definition 2. For a sensor $s_i \in S$, the in-node-maximum of s_i is the maximum data item.

For example, if the collected data of s_i are $D_i = \{d_{i,1}, d_{i,2}, \ldots, d_{i,N}\}$ and $d_{i,1} > d_{i,2} > \cdots > d_{i,N}$, then $d_{i,1}$ is the in-node-maximum of s_i.

Definition 3. For a top-k query, we define Φ is a sensor set consisting of k sensors whose in-node-maximums are the k largest in-node-maximums of sensors in S, that is

$$\Phi \subseteq S \wedge |\Phi| = k \wedge \left(\forall s_i \in \Phi, s_j \in S - \Phi \longrightarrow d_{i,1} > d_{j,1}\right). \quad (1)$$

Lemma 1. $L(R_t) \geq \min(\{d_{i,1} | s_i \subseteq \Phi\})$

Proof. According to Definition 1, $L(R_t)$ is the lower bound of R_t, which is the kth largest data of D. Because $|\Phi| = k$, there are k in-node-maximums of sensors of Φ, that is, $|\{d_{i,1} | s_i \in \Phi\}| = k$. Thus, $\min(d_{i,1} | s_i \in \Phi\})$ is the kth largest data of $\{d_{i,1} | s_i \in \Phi\}$, where $\min(*)$ represents the minimum of a dataset. Because $\{d_{i,1} | s_i \in \Phi\} \subseteq D$, we have that $L(R_t) \geq \min(\{d_{i,1} | s_i \in \Phi\})$ holds. □

Lemma 2. *Φ is the candidate sensor set of a query, which means that all data in the query result R_t are contributed by sensors of Φ, that is,*

$$R_t \subseteq \bigcup_{s_i \in \Phi} D_i. \quad (2)$$

Proof. We give the proof by contradiction. We are assuming that there is at least one data of R_t, which is not contributed by a sensor of Φ. It means that $\exists x (x \in R_t \wedge x \in D_j)$, where x is collected by s_j and s_j is not in Φ, i.e., $s_j \in S - \Phi$. We are assuming that x is the lth largest data of D. Then, we can deduce two results: 1. $1 \leq l \leq k$ holds because of $x \in R_t$ and $|R_t| = k$. 2. According to the definition of Φ, for $\forall y \in \{d_{i,1} | s_i \in \Phi\}$, because x is assumed to be collected by $s_j \in S - \Phi$, we have $y > d_{j,1} \geq x$, where $d_{j,1}$ is the in-node-maximum of s_j. Additionally, there are k in-node-maximums contributed by sensors in Φ, i.e., $|d_{i,1} | s_i \in \Phi\}| = k$. Therefore, we can deduce that $l > k$ holds.

Obviously, there are contradictions between 1 and 2. As a result, we deduce that Lemma 2 holds.

Lemma 2 It indicates that all sensors in Φ are candidate sensors, which contribute the query result. In addition, Φ is also the minimized candidate sensor set, and we prove it in Lemma 3. □

Lemma 3. *Φ is the minimized candidate sensor set that contribute the query result R_t.*

Proof. To prove this lemma, we have to prove the following two observations. □

Observation 1. For $\forall d_{j,h} \in D_j$ where $\forall s_j \in S - \Phi$, $L(R_t) > d_{j,h}$ holds.

Observation 2. Any sensor deletion from Φ could incur the incompleteness of query result. If and only if the two observations hold simultaneously, then we can deduce that Φ is the minimized candidate sensor set that contribute the query result.

Proof to Observation 1. According to Definition 3, for $\forall s_i \in \Phi$ and $\forall s_j \in S - \Phi$, $d_{i,1}$ and $d_{j,1}$ are their in-node-maximums and $d_{i,1} > d_{j,1}$ holds. Because $d_{j,1}$ is the in-node-maximum of s_j, $d_{i,1} \geq d_{j,h}$ holds where $d_{j,h} \in D_j$. Thus, $d_{i,1} > d_{j,h}$ holds. In addition, because s_i could be any sensor of Φ, we can deduce that $\min(\{d_{i,1} | s_i \in \Phi\}) > d_{j,h}$. At last, Lemma 1 indicates that $L(R_t) \geq \min(\{d_{i,1} | s_i \in \Phi\})$; therefore, $L(R_t) > d_{j,h}$ holds, and the first observation is proved. □

Proof to Observation 2. To prove the second observation, we just need to prove that, for any sensor of Φ, its collected data could belong to the query result R_t. If it is true, then deleting any sensor from Φ could cause the incompleteness of R_t. We are assuming that the collected data of sensors of Φ satisfy: $\forall d_{i,j} (s_i \in \Phi \wedge 2 \leq j \leq N) \longrightarrow d_{i,j} < \min(d_{p,1} \mid s_p \in \Phi)$. Because $|\Phi| = k$, the top-k query result R_t is determined and $R_t = \{d_{p,1} | s_p \in \Phi\}$. It means that the in-node-maximums of all sensors of Φ are just the elements of R_t. It is obvious that, in such circumstance, deleting any sensor from Φ will incur the incompleteness of R_t. Therefore, the second observation is proved.

According to the proofs, the above two observations both hold. Thus, Φ is the minimized candidate sensor set that contribute the query result. □

4.2. Minimized Candidate Encrypted Dataset. To protect data privacy, each sensor owns its private key only by itself. When a query is started, sensors first encrypt the qualified data by their keys and then submit the encrypted data to sink. For sensor s_i, we are assuming its key is g_i, which is only shared by s_i and sink. The encrypted data of $d_{i,j}$ is denoted as $(d_{i,j})_{g_i}$.

Definition 4 (minimized candidate encrypted dataset). For a top-k query, the minimized candidate encrypted dataset, denoted as Γ, is contributed by sensors of Φ and consists of the minimum number of encrypted data that have the encrypted query result in it.

We are assuming that the candidate sensors are $\Phi = \{s_1, s_2, \ldots, s_k\}$ and their in-node-maximums are $\{d_{1,1}, d_{2,1}, \ldots, d_{k,1}\}$, respectively, where $d_{1,1} > d_{2,1} > \cdots > d_{k,1}$. For any sensor $s_i \in \Phi$, its collected data items are $\{d_{i,1}, d_{i,2}, \ldots, d_{i,N}\}$, where $d_{i,1} > d_{i,2} > \cdots > d_{i,N}$. Thus, the calculation of Γ is given as follows:

$$\Gamma = \bigcup_{s_i \in \Phi} \Gamma_i, \tag{3}$$

where

$$\Gamma_i = \begin{cases} \left\{ \left(d_{i,j}\right)_{g_i} | 1 \leq j \leq k - i + 1 \right\}, & N \geq k - i + 1, \\ \left\{ \left(d_{i,j}\right)_{g_i} | 1 \leq j \leq N \right\}, & N < k - i + 1. \end{cases} \tag{4}$$

We give an example to describe the minimized candidate encrypted dataset. As shown in Figure 2, we are assuming that there are 5 nodes $\{s_1, s_2, s_3, s_4, s_5\}$, and each sensor has collected 4 data items. Their in-node-maximums satisfy $d_{1,1} > d_{2,1} > \cdots > d_{5,1}$. For sensor s_i, its collected data satisfy $d_{i,1} > d_{i,2} > d_{i,3} > d_{i,4}$. According to Definition 4, the minimized candidate encrypted datasets when $k = 3$ and $k = 5$ are shown in the dotted-lined area and solid-lined area, respectively.

Lemma 4. *Γ is the minimized candidate encrypted dataset that has the encrypted query result.*

Proof. To prove this lemma, the following two observations need to be proved. □

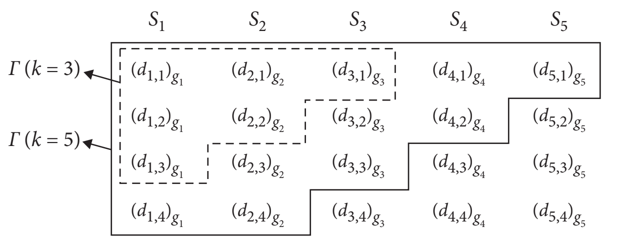
Observation 3. For any $(d_{i,j})_{g_i} \notin \Gamma$, which is generated by s_i, $L(R_t) > d_{i,j}$ holds.

Observation 4. Any encrypted data deletion from Γ could incur the incompleteness of query result. If and only if the two observations hold simultaneously, then we can deduce that Γ is the minimized candidate encrypted dataset that has the encrypted query result.

Proof of Observation 1. For sensor s_i, it has two alternative cases, which are $s_i \notin \Phi$ or $s_i \in \Phi$. We give the proofs in such two cases:

(i) Case I: $s_i \notin \Phi$. According to Lemma 3, Φ is the minimized candidate sensor set that contribute the query result R_t. Because $s_i \notin \Phi$, we have $d_{i,j} \notin R_t$, where $d_{i,j} \in D_j$ and then $L(R_t) > d_{i,j}$ is deduced.

(ii) Case II: $s_i \in \Phi$. Because $(d_{i,j})_{g_i} \notin \Gamma$, $k - i + 2 \leq j \leq N$ is deduced according to equation (4). In the calculation of Γ, $d_{1,1} > d_{2,1} > \cdots > d_{i,1} > \cdots > d_{k,1}$ and $d_{i,1} > d_{i,2} > \cdots > d_{i,j} > \cdots > d_{i,N}$ are the given assumption. Thus, there are at least $k' = i + j - 2$ data



s_1 s_2 s_3 s_4 s_5
$\Gamma\,(k=3)$
$\Gamma\,(k=5)$
$(d_{1,1})_{g_1}$ $(d_{2,1})_{g_2}$ $(d_{3,1})_{g_3}$ $(d_{4,1})_{g_4}$ $(d_{5,1})_{g_5}$
$(d_{1,2})_{g_1}$ $(d_{2,2})_{g_2}$ $(d_{3,2})_{g_3}$ $(d_{4,2})_{g_4}$ $(d_{5,2})_{g_5}$
$(d_{1,3})_{g_1}$ $(d_{2,3})_{g_2}$ $(d_{3,3})_{g_3}$ $(d_{4,3})_{g_4}$ $(d_{5,3})_{g_5}$
$(d_{1,4})_{g_1}$ $(d_{2,4})_{g_2}$ $(d_{3,4})_{g_3}$ $(d_{4,4})_{g_4}$ $(d_{5,4})_{g_5}$

FIGURE 2: Minimized candidate encrypted datasets when $k = 3$ and 5.

larger than $d_{i,j}$. According to $k - i + 2 \leq j \leq N$ and $k\prime = i + j - 2$, then we have $k \leq k\prime \leq i + N - 2$. It means that there are at least k data larger than $d_{i,j}$. Definition 1 shows that the query result R_t has the k largest data, so the minimum of R_t is obviously larger than $d_{i,j}$, that is, $L(R_t) > d_{i,j}$. The deductions in two cases both lead to the same result $L(R_t) > d_{i,j}$, and the first observation is proved. □

Proof of Observation 2. To prove the second observation, we just need to prove that, for any $(d_{i,j})_{g_i} \in \Gamma$, the corresponding plaintext data $d_{i,j}$ could belong to R_t. If it is true, then deleting any encrypted data from Γ could cause the incompleteness of R_t. According to the assumptions of the calculation of Γ that the minimized candidate sensor set is $\Phi = \{s_1, s_2, \ldots, s_k\}$, where their in-node-maximums satisfy $d_{1,1} > d_{2,1} > \cdots > d_{k,1}$ and the collected data of any $s_i \in \Phi$ satisfy $d_{i,1} > d_{i,2} > \cdots > d_{i,N}$, for the data $d_{i,j}$ and $C = \{d_{1,1}, d_{2,1}, \ldots, d_{i-1,1}, d_{i,1}, d_{i,2}, \ldots, d_{i,j-1}\}$, each data in C is larger than $d_{i,j}$ and $|C| = i + j - 2$. If the following equation holds, then $d_{i,j}$ is the $(i + j - 2)$th largest data.

$$\forall d_{p,q}\Big(\big(d_{p,q}\big)_{g_p} \in \Gamma \wedge d_{p,q} \notin C\Big) \longrightarrow d_{p,q} < d_{i,j}. \quad (5)$$

According to the calculation of Γ in equations (3) and (4), we have $j \leq k - i + 1$, then $|C| \leq k - 1$ holds. It means that $d_{i,j}$ is at least the kth largest data when equation (5) hold. In such scenario, $d_{i,j}$ always belongs to R_t. Therefore, we have that deleting any encrypted data from Γ could cause the incompleteness of R_t. Observation 2 is proved.

According to the above proofs, two observations both hold. Thus, Γ is the minimized candidate encrypted dataset that has the encrypted query result. Lemma 4 is proved.

Lemma 4 It indicates that Γ is the minimized candidate encrypted dataset that has the encrypted query result. It is a key to achieve efficient privacy-preserving query processing method. □

5. Top-k Query Processing

At first, an efficient privacy-preserving and collusion-resisting top-k (EPCT) query scheme is introduced here. Then, the correctness and security analysis, and performance of the proposed EPCT protocol will be presented.

5.1. Query Processing Protocol. The queried nodes and the sink are involved as the cooperators in this EPCT protocol. To perform the protocol, sensors and the sink are firstly settled with keys in the network deployment. Each sensor is deployed a private key, and it only shares the key with the sink. The sink owns keys of all sensors, whereas sensors have no idea of each other's keys. The protocol has two phases, shown in Figure 3. The command is broadcasted to sensors in S, before the sink receives a top-k query $\text{Query}_t = (t, S, k)$ in the first phase from the user. Once the sensor s_i gets Query_t, it transmits the encrypted in-node-maximum in the queried time slot t to the sink. As the first phase ends, the second phase begins. In the second phase, the minimized candidate sensor set is determined according to the maximum values of the queried sensors. Then, the sink transmits the second phase data request command to those candidate sensors. After each candidate sensor submits the qualified encrypted data, the sink obtains the minimized candidate encrypted dataset, and then, it will get the final query result after decryption. The processing of the top-k query Query_t is finished.

The detailed procedures of the query processing protocol are shown in Protocol 1.

Protocol 1. EPCT protocol is shown as follows:

(1) Phase 1:

  (1) As a query $\text{Query}_t = (t, S, k)$ is running, the first phase starts to process. Sink broadcasts Query_t through all the networks and initials the dataset $\Gamma = \varnothing$. Then, it waits till the first phase responses from the queried nodes in the networks.

  (2) For each node $s_i \in S$, s_i encrypts its in-node-maximum $d_{i,1}$ by using its private key g_i, after s_i gets the Query_t. Then, s_i generates the encrypted data $(d_{i,1})_{g_i}$, submitting the message as follows to the sink.

$$s_i \longrightarrow \text{sink: } \langle t, i\ d(s_i), \big(d_{i,1}\big)_{g_i}\rangle \quad (6)$$

(2) Phase 2:

  (1) As the submitted message from a queried sensor $s_i \in S$ arrives, $\langle t, i\ d(s_i), (d_{i,1})_{g_i}\rangle$, the sink decrypts $(d_{i,1})_{g_i}$ with the shared private key g_i and gets the plaintext in-node-maximum of s_i. s_i obtains all the decrypted in-node-maximums of the nodes in S, $\{d_{i,1} | s_i \in S\}$, before it determines the top-k data. If the determined top-k data are $\{d_{1,1}, d_{2,1}, \ldots, d_{k,1}\}$ where $d_{1,1} > d_{2,1} > \cdots > d_{k,1}$ and the corresponding sensor list according to the decent sequence of data are $\Phi = \{s_1, s_2, \ldots, s_k\}$. According to Lemma 3, Φ are the set of minimized candidate sensors. Then, the sink appends $\{d_{1,1}, d_{2,1}, \ldots, d_{k,1}\}$ into Γ and transmits the following messages to the k-1 candidate nodes in $\Phi - \{s_k\}$ in unicast mode.

$$\text{sink} \longrightarrow s_i\text{: } \langle t, (k-i)_{g_i}\rangle, \quad \forall s_i \in \Phi - \{s_k\}. \quad (7)$$

  (2) For each candidate node $s_i \in \Phi - \{s_k\}$, as the message $\langle t, (k-i)_{g_i}\rangle$ arrives, s_i decrypts the ciphertext and gets the plaintext number $k - i$.

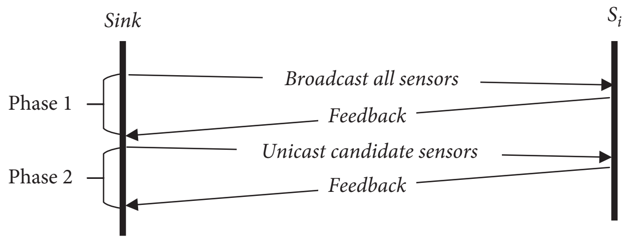


FIGURE 3: EPCT protocol query process.

Then, s_i encrypts $k - i$ collected data items and sends them to the sink, e.g.,

$$s_i \longrightarrow \text{sink: } \langle t, i\ d(s_i), LR_i \rangle, \quad (8)$$

where

$$LR_i = \begin{cases} \left\{ \left(d_{i,j}\right)_{g_i} | 2 \le j \le k - i + 1 \right\}, & N \ge k - i + 1, \\ \left\{ \left(d_{i,j}\right)_{g_i} | 2 \le j \le N \right\}, & N < k - i + 1. \end{cases} \quad (9)$$

(3) The sink obtains the message $\langle t, i, LR_i \rangle$ transmitted by the candidate node $s_i \in \Phi - \{s_k\}$ in the second phase, before the ciphertext of the message is decrypted. The plaintext data after decryption are denoted as $\text{Dec}(LR_i, g_i)$ and appended into Γ. After all messages submitted from the candidate nodes are processed, the minimized candidate encrypted dataset Γ is determined, where

$$\Gamma = \bigcup_{s_i \in \Phi} \left(\left\{ d_{i,1} \right\} \cup \text{Dec}\left(LR_i, g_i\right) \right). \quad (10)$$

(4) The sink gets the top-k data of Γ, which is the exact query result R_t.

$$R_t \subseteq \Gamma \wedge \left|R_t\right| = k \wedge \left(\forall x \in R_t, y \in \left(\Gamma - R_t\right) \longrightarrow x > y\right). \quad (11)$$

As presented in Protocol 1, the query command Query$_t$ arrives from the user in the first phase, before the sink broadcasts it through the whole network. As a queried sensor knows Query$_t$, it encodes the in-node-maximum before transmitting the encrypted data to the sink, where the received ciphertext is decrypted to obtain the in-node-maximums of the queried sensors in the second phase. Afterwards, the sink uses the in-node-maximums to determine the candidate sensor set $\Phi - \{s_k\}$, and then, it unicasts each candidate sensor in $\Phi - \{s_k\}$ to start the second phase. Once a candidate sensor receives the unicast message, it submits the rest data in ciphertext according to the request to the sink. As the sink obtains all the needed data from candidate nodes, the query result is determined in the end.

5.2. Protocol Analysis

5.2.1. Correctness Analysis. In the proposed EPCT protocol, when a user starts a query command Query$_t$, the sink will know the minimized candidate encrypted dataset Γ after interactions of the sink and sensors within two phases. Γ is consisting of the coded data items of query result. According to Lemma 4, for any $(d_{i,j})_{g_i} \notin \Gamma$, $(d_{i,j})_{g_i}$ does not belong to the query result R_t, definitely. Additionally, Γ is the minimized candidate encrypted dataset that contains the encrypted query result. Any encrypted data deletion from Γ could incur the incompleteness of query result. As Γ received by the sink, it can get the query result by obtaining the top-k data from Γ. Therefore, our proposed scheme is capable of guaranteeing the correctness of top-k query result.

5.2.2. Security Analysis. The security analysis is conducted here for the privacy of the collected data and the query results. With the cooperation of the sink and the sensors in EPCT in these two phases, each node is deployed with a private key, which is only shared with the sink. The collected data of sensors only exists in data submission from sensors to the sink. When a top-k query is started, two phases of query processing are performed. In the first phase, each sensor performs a symmetric encryption to encrypt its in-node-maximum and then transmits it to the sink. Secondly, candidate nodes are unicastly informed by the sink. They encrypted a fixed number of collected data according to the request and then sends the enciphered date to the sink node. Clearly, the data collected and transmitted through the network are all in the form of ciphertext. Every node in WSN owns a unique private key, so it can only get access to the data it collected. However, it fails to know the data collected by other sensors because of the computational infeasibility of symmetric encryption. Even a few nodes probably are attacked and colluded with adversaries, they can only snoop the collected data of those colluded sensors, but they have no idea of the collected data of innocent sensors. Besides, due to the query result is decrypted and computed in the sink and sensor nodes only process the encrypted data for the query, it is hard for the attackers to know the plaintext query result even if a few compromised sensors are colluded with them. Therefore, this proposed EPCT is a privacy-preserving and anticollusion top-k query processing protocol, which can protect the privacy of collected data of sensors even a few compromised sensors are in collusion with the adversaries, which can protect the privacy of collected data from adversaries even a few compromised sensors are in collusion with the adversaries.

5.2.3. Communication Cost Analysis. In WSNs, sensors have limited energy resource, and the energy are mainly consumed by communication. During the top-k query procedures, the communication cost of the network is mainly caused by transmission overhead of sensors. The parameters used in sensor networks are introduced in Table 1.

We are assuming that the transmission overhead of phase 1 and phase 2 are C_1 and C_2, respectively. According to the proposed EPCT protocol, all sensors participate in phase 1, whereas only the candidate sensors participate in phase 2. Then, we obtain

Table 1: Parameters description.

Para	Description
l_{id}	The space size of a sensor ID
l_t	The space size of a time-slot
l_c	The space size of a coded data item
l_q	The space size of a query command
L	The average path length from sensors to the sink

$$\begin{aligned} C_1 &= n \cdot l_q + n \cdot \left(l_{id} + l_t + l_c\right) \cdot L, \\ C_2 &= (k-1) \cdot \left(l_t + l_c\right) \cdot L + \sum_{i=1}^{k-1} \left(l_{id} + l_t + i \cdot l_c\right) \cdot L \\ &= (k-1) \cdot l_{id} \cdot L + (k-1) \cdot \left(l_{id} + l_t\right) \cdot L + \frac{k \cdot (k-1)}{2} \cdot l_c \cdot L \\ &= (k-1) \cdot \left(l_{id} + 2l_t + l_c\right) \cdot L + \frac{k \cdot (k-1)}{2} \cdot l_c \cdot L \end{aligned} \tag{12}$$

The total communication overhead of the whole network is computed as follows:

$$\begin{aligned} C_{\text{total}} = C_1 + C_2 &= n \cdot l_q + (n + k - 1) \cdot \left(l_{id} + l_t + l_c\right) \cdot L \\ &\quad + (k-1) \cdot l_t \cdot L + \frac{k \cdot (k-1)}{2} \cdot l_c \cdot L. \end{aligned} \tag{13}$$

6. Performance Evaluation

Based on the improved simulator of [34], we implement three protocols, EPCT, PCTQ [31], and a naive protocol (Naive). For Naive scheme, each node queried firstly encodes its k highest data items and then submits them to the sink. After the sink gets all the ciphertext from sensors, it decrypts them to obtain the final query result. The performance is evaluated by the communication overhead in WSNs.

This experiment is conducted on a PC with an AMD R5-3600 (6 cores 12 threads 4.2 Ghz) CPU and 32 GB RAM, running 64-bit win 10 professional OS and Java JDK 1.8. In the simulation, we generate 10 networks with random topologies, and each network is distinguished by different network IDs. In each network, sensors are randomly distributed in area covering a $200 \times 200\,\text{m}^2$, and the communicating radius of a sensor is 6 m. The collected data of sensors are randomly generated in each time slot. The network communication cost C_{total} is measured by computing the average result of these 10 networks. The default settings of other parameters are shown in Table 2.

(1) C_{total} versus Network ID. Figure 4 presents that the transmission overhead of these methods are distributed uniformly in different networks. Naive has much higher cost compared with PCTQ and EPCT. Statistically, the communication overhead of EPCT is averagely 89.06% and 43.23% lower than that of Naive and PCTQ, respectively.

Table 2: Default settings of parameters.

Parameter	n	k	l_{id}	l_q	l_t	l_c
Value	500	10	4 byte	8 byte	4 byte	16 byte

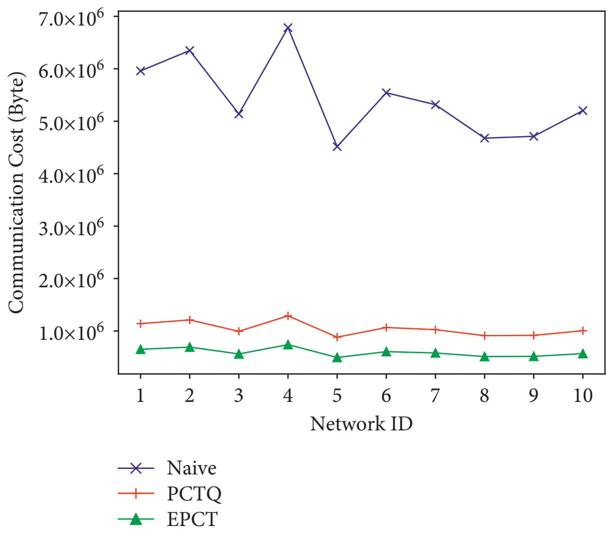


Figure 4: C_{total} vs. Network ID.

(2) C_{total} versus l_c. Figure 5 shows that the communication overhead of EPCT, PCTQ, and Naive increases as the space size of an encrypted data item l_c increases. The reason is that the transmission overhead of three approaches are all in proportion to the space size of an encrypted data item. The growth rates of communication overhead in EPCT and PCTQ are smaller than that in Naive. Statistically, EPCT reduces about 89.14% and 38.32% transmission overhead than Naive and PCTQ, respectively.

(3) C_{total} versus n. Figure 6 presents that the communication overhead of three schemes grows as the number of sensors n increases. The reason is that the more sensors are queried, the more data are transmitted in the network, i.e., the higher communication costs. Moreover, the curves in Figure 6 tell that the growth rate of transmission overhead in Naive is significantly higher than that in PCTQ and EPCT. Statistically, EPCT saves about 89.51% and 42.00% communication overhead than Naive and PCTQ, respectively.

(4) C_{total} versus k. As shown in Figure 7, the transmission overhead of three methods all increases as the number of requested data items k increases. It is that when k increases, more data items are requested in all three protocols. The growth rates of communication cost in PCTQ and EPCT are both lower than that in Naive. Specifically, EPCT saves about 93.44% and 44.57% on average than Naive and PCTQ in communication cost.

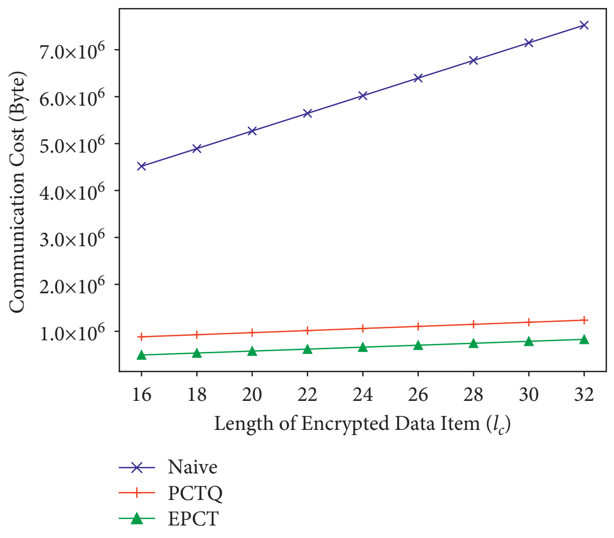


FIGURE 5: C_{total} vs. l_c.

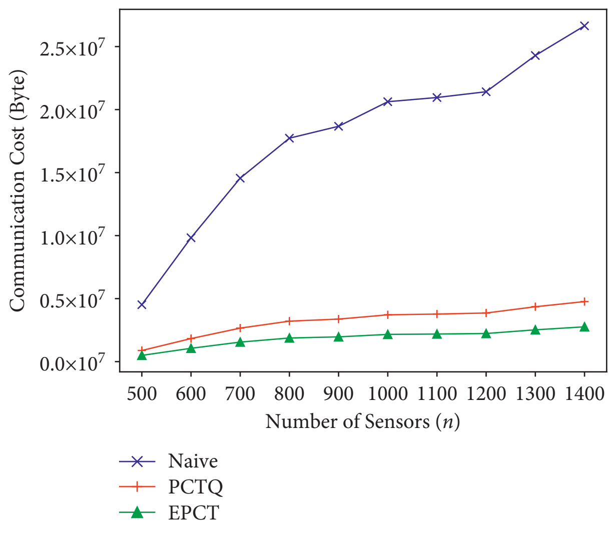


FIGURE 6: C_{total} vs. n.

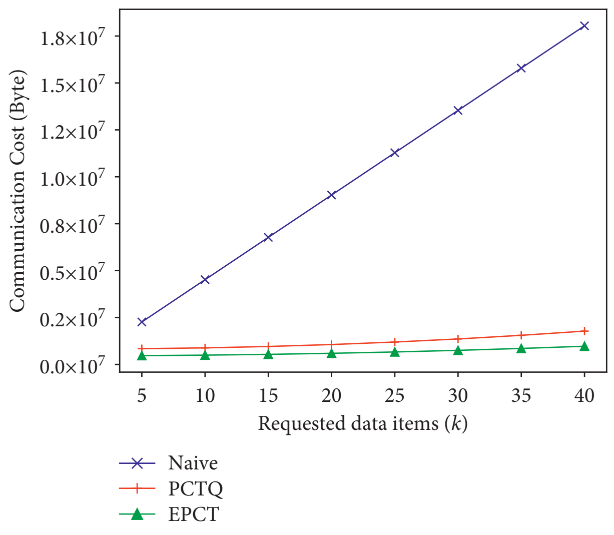


FIGURE 7: C_{total} vs. k.

According to the results of Figures 4–7, the transmission overhead of EPCT is the lowest in three protocols, whereas the overhead of Naive is much higher than the others. Because in EPCT and PCTQ, transmission only caused by candidate sensors need to, whereas in Naive scheme, all sensors are participated in transmission. Specifically, there are $k \cdot (k+1)/2$, at least k^2, and $n \cdot k$ encrypted data items are submitted from sensors to the sink in EPCT, PCTQ, and Naive, respectively. As a result, according to the above evaluations, compared with the PCTQ and Naive protocol, it has been shown that the proposed EPCT has less network communication cost and more efficient.

7. Conclusion

Data privacy threat arises during providing top-k query processing in the wireless sensor networks. To address this issue, we proposed a novel and efficient top-k query processing approach, which is capable of privacy protection and anticollusion. We fist present a minimized candidate encrypted dataset determination model, which is the foundation of the protocol. The model guides the idea of query processing and guarantees the correctness of the protocol. The symmetric encryption with different private keys in each node is employed for data privacy and even to prevent the attackers from colluding with a few nodes. Based on the above model and security setting, two phases of secure interactions between queried nodes and the sink are designed to implement the query processing protocol. The security analysis shows that our scheme is capable of providing privacy-protecting and collusion-resisting top-k queries, whereas the experimental result indicates that our approach is efficient by evaluating the network communication.

Data Availability

The data generated randomly in WSN and 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 sponsored by the Natural Science Foundation of China (Grant nos. 61902199, 61872197, 61972209, and 61872193), the Postdoctoral Science Foundation of China (Grant no. 2019M651919), and the Natural Research Foundation of Nanjing University of Posts and Telecommunications (Grant nos. NY219142 and NY217119).

References

[1] Z. H. Liang Liu and L. Wang, "Energy-efficient and privacy-preserving spatial range aggregation query processing in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 15, 2019.

[2] P. Li, Y. Liu, X. Gao, H. Li, and P. Gong, "Energy-efficient time and energy resource allocation in non-selfish symbiotic cognitive relaying sensor network with privacy preserving for smart city," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, pp. 1687–1499, 2021.

[3] X. Kui, J. Feng, X. Zhou et al., "Securing top-k query processing in two-tiered sensor networks," *Connection Science*, vol. 33, no. 1, pp. 62–80, 2021.

[4] H. Peng, X. Zhang, H. Chen, Y. Wu, Y. Wu, and J. Zeng, "Enable privacy preservation and result verification for top-k query in two-tiered sensor networks," *IEEE Trustcom/BigDataSE/ISPA*, vol. 1, pp. 555–562, 2015.

[5] X. Liao and J. Li, "Privacy-preserving and secure top-k query in two-tier wireless sensor network," in *Proceedings of the 2012 IEEE Global Communications Conference (GLOBECOM)*, pp. 335–341, Anaheim, CA, USA, December 2012.

[6] R. Li, A. X. Liu, S. Xiao, H. Xu, B. Bruhadeshwar, and A. L. Wang, "Privacy and integrity preserving top- k query processing for two-tiered sensor networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2334–2346, 2017.

[7] Y. T. Tsou, Y. L. Hu, Y. Huang, and S. Y. Kuo, "PCTopk: privacy-and correctness-preserving functional top-k query on un-trusted data storage in two-tiered sensor networks," in *Proceedings of the 2014 IEEE 33rd International Symposium on Reliable Distributed Systems*, pp. 191–200, Nara, Japan, October 2014.

[8] H. Dai, Q. Ye, X. Yi, R. He, G. Yang, and J. Pan, "VP2RQ: efficient verifiable privacy-preserving range query processing in two-tiered wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 12, 2016.

[9] L. Dong, X. Chen, J. Zhu, H. Chen, K. Wang, and C. Li, "A secure collusion-aware and probability-aware range query processing in tiered sensor networks," in *Proceedings of the 2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*, pp. 110–119, Montreal, Canada, October 2015.

[10] Y.-T. Tsou, C.-S. Lu, and S.-Y. Kuo, "SER: secure and efficient retrieval for anonymous range query in wireless sensor networks," *Computer Communications*, vol. 108, pp. 1–16, 2017.

[11] J. Zeng, L. Dong, Y. Wu, H. Chen, C. Li, and S. Wang, "Privacy-preserving and multi-dimensional range query in two-tiered wireless sensor networks," in *Proceedings of the GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–7, Singapore, December 2017.

[12] X. Zhang, H. Peng, L. Dong, H. Chen, and H. Sun, "SET: secure and efficient top-k query in two-tiered wireless sensor networks," in *Proceedings of the Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*, pp. 495–510, Beijing, China, August 2017.

[13] H. Peng, B. Liu, J. Liu, D. Li, and L. Yun, "Dp2T: preserving data privacy for top-K query in wireless sensor networks," in *Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pp. 885–888, Beijing, China, November 2018.

[14] M. Xingpo, L. Junbin, M. Wenpeng, L. Yin, L. Ran, and K. Xiaoyan, "A secure top-k query processing protocol for two-tiered wireless sensor networks," *Journal of Computer Research and Development*, vol. 55, p. 2490, 2018.

[15] H. Dai, M. Wang, X. Yi, G. Yang, and J. Bao, "Secure max/min queries in two-tiered wireless sensor networks," *IEEE Access*, vol. 5, pp. 14478–14489, 2017.

[16] H. Dai, T. Wei, Y. Huang, J. Xu, and G. Yang, "Random secure comparator selection based privacy-preserving MAX/MIN query processing in two-tiered sensor networks," *Journal of Sensors*, vol. 2016, Article ID 6301404, 13 pages, 2016.

[17] H. Wu and L. Wang, "Efficient and secure top-k query processing on hybrid sensed data," *Mobile Information Systems*, vol. 2016, Article ID 1685054, 10 pages, 2016.

[18] F. Liu, X. Ma, J. Liang, and M. Lin, "Verifiable top-k query processing in tiered mobile sensor networks," *International Journal of Distributed Sensor Networks*, vol. 11, pp. 437678–437678, 2015.

[19] J. Tang and Z. Zhou, "A priority-aware multidimensional top-k query processing in wireless sensor networks," *Procedia Computer Science*, vol. 129, pp. 149–158, 2018.

[20] J. Shiraishi, H. Yomo, and K. Huang, "Content-based wake-up for top-k query in wireless sensor networks," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 1, pp. 362–377, 2021.

[21] A. F. Baig and S. S. Eskeland, "Privacy, and usability in continuous authentication: a survey," *Sensors*, vol. 21, 2021.

[22] Q. Xie, K. Li, X. Tan, L. Han, and W. T. Bin Hu, "A secure and privacy-preserving authentication protocol for wireless sensor networks in smart city," *EURASIP Journal on Wireless Communications and Networking*, vol. 12, 2021.

[23] K. A. Shah and D. Jinwala, "Privacy preserving secure expansive aggregation with malicious node identification in linear wireless sensor networks," *Frontiers of Computer Science*, vol. 15, 2021.

[24] S. Hu, L. Liu, L. Fang, F. Zhou, and R. Ye, "A novel energy-efficient and privacy-preserving data aggregation for WSNs," *IEEE Access*, vol. 8, pp. 802–813, 2020.

[25] J. Zheng, B. Song, Y. Wang, and H. Wang, "Adaptive filter updating for energy-efficient top-k queries in wireless sensor networks using Gaussian process regression," *International Journal of Distributed Sensor Networks*, vol. 11, 2015.

[26] G. Li, X. Gao, M. Liao, and B. Han, "An iterative algorithm to process the top-k query for the wireless sensor networks," *International Journal of Embedded Systems*, vol. 7, no. 1, pp. 26–33, 2015.

[27] Z. Chen, M. He, W. Liang, and K. Chen, "Trust-aware and low energy consumption security topology protocol of wireless sensor network," *Journal of Sensors*, 2015.

[28] J. Tang, Z. Wang, Y. Sun, C. Du, and Z. Zhou, "Top-k queries in wireless sensor networks leveraging hierarchical grid

index," in *Proceedings of the 2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 381–386, Birmingham, United Kingdom, July 2014.

[29] C. Zhu, L. T. Yang, L. Shu, V. C. Leung, T. Hara, and S. Nishio, "Insights of top-k query in duty-cycled wireless sensor networks," *IEEE Transactions on Industrial Electronics*, vol. 62, pp. 1317–1328, 2014.

[30] H. Haiping, F. Juan, W. Ruchuan, and Q. XiaoLin, "An exact top-k query algorithm with privacy protection in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 10, 2014.

[31] J. Zhou, H. Dai, J. Zhu, R. Qi, G. Yang, and J. Xu, "A privacy-preserving and collusion-resisting top-K query processing in WSNs," in *Proceedings of the 2020 16th International Conference on Mobility, Sensing and Networking (MSN)*, pp. 677–682, IEEE, Yokyo, Japan, December 2020.

[32] L. Wang, M. Zhao, J. Chen et al., "A novel privacy-and integrity-preserving approach for multidimensional data range queries in two-tiered wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 15, 2019.

[33] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS - Operating Systems Review*, vol. 36, no. SI, pp. 131–146, 2002.

[34] A. Coman, M. A. Nascimento, and J. Sander, "A framework for spatio-temporal query processing over wireless sensor networks," in *Proceedings of the 1st International Workshop on Data Management for Sensor Networks: in conjunction with VLDB 2004*, pp. 104–110, Toronto, Canada, August 2004.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9430132, 10 pages
https://doi.org/10.1155/2021/9430132

Research Article

BCEAD: A Blockchain-Empowered Ensemble Anomaly Detection for Wireless Sensor Network via Isolation Forest

Xiong Yang[1,2] **Yuling Chen**[1] **Xiaobin Qian**[3] **Tao Li**[1] **and Xiao Lv**[4]

[1]*State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, China*
[2]*Guangxi Key Laboratory of Cryptography and Information Security, Guilin, China*
[3]*Guizhou CoVision Science & Technology Co., Ltd., Guiyang, China*
[4]*Guizhou Shuanhui Big Data Industry Development Co., Ltd., Guiyang, China*

Correspondence should be addressed to Yuling Chen; ylchen3@gzu.edu.cn

Received 27 September 2021; Revised 18 October 2021; Accepted 27 October 2021; Published 10 November 2021

Academic Editor: Weizhi Meng

The distributed deployment of wireless sensor networks (WSNs) makes the network more convenient, but it also causes more hidden security hazards that are difficult to be solved. For example, the unprotected deployment of sensors makes distributed anomaly detection systems for WSNs more vulnerable to internal attacks, and the limited computing resources of WSNs hinder the construction of a trusted environment. In recent years, the widely observed blockchain technology has shown the potential to strengthen the security of the Internet of Things. Therefore, we propose a blockchain-based ensemble anomaly detection (BCEAD), which stores the model of a typical anomaly detection algorithm (isolated forest) in the blockchain for distributed anomaly detection in WSNs. By constructing a suitable block structure and consensus mechanism, the global model for detection can iteratively update to enhance detection performance. Moreover, the blockchain guarantees the trust environment of the network, making the detection algorithm resistant to internal attacks. Finally, compared with similar schemes, in terms of performance, cost, etc., the results prove that BCEAD performs better.

1. Introduction

In recent years, the booming Internet of Things is revolutionizing the world. As its supporting technology, wireless sensor networks have also received extensive attention [1, 2]. WSNs are a multihop self-organizing network formed by many sensor nodes deployed in the monitoring area to communicate. It gets rid of the limitation of the cable, realizes the wireless communication of the network, and has a wide range of application scenarios. However, due to the backwardness of WSNs security technology, various security issues limit the practical application of WSNs [3, 4].

Various security technologies and strategies have emerged for protecting network security. Intrusion detection is a classic network security technology [5]. Early intrusion detection systems (IDS) mostly utilize misuse detection. Misuse detection record the attacks by a signature database, then judge an intrusion with the events or data matching the signatures. However, misuse detection is not practical enough because it cannot detect unrecorded attacks. Nowadays, anomaly detection has been more widely used [6] with the development of machine learning. Anomaly detection comes from the statistical community [7]. It establishes a standard model and judges the events or data that do not match with the model as an intrusion. Although anomaly detection requires some model training time and produces a higher false alarm rate, it can detect new unknown intrusions. The performance of anomaly detection will continue to increase and make outstanding contributions to protecting network security with the optimization of modeling algorithms in anomaly detection.

The structure of intrusion detection systems has become richer for stronger practicability and applicability. For example, the proposal of distributed intrusion detection systems (DIDSs) eases the pressure of detecting heterogeneous networks. The DIDS is similar to ensemble learning [8], and the system builds multiple detection models in a large-scale network. Therefore, the DIDS not only balances network energy consumption but also improves detection performance. However, distributed anomaly detection must ensure trust between nodes to prevent internal attacks. It is the prerequisite for its further application.

In recent years, the emergence of blockchain technology has pointed out a path worth trying to solve the distributed trust problem in the Internet of Things environment. Blockchain is a peer-to-peer distributed network with features such as non-tempera, decentralization, transparency, and system autonomy [9], which can effectively enhance device security and network collaboration in the Internet of Things. Nowadays, some distributed intrusion detection based on blockchain has been proposed [10]. The system packs the detection results or alarms into blocks and shares them in the network to manage trust among domains. However, due to the limitation of the data shared by the blockchain, the above scheme still has its limitations.

Therefore, we propose a blockchain-based ensemble anomaly detection scheme. The scheme stores the model of a typical anomaly detection algorithm (isolated forest) in the blockchain network and performs distributed anomaly detection in the WSNs. By constructing a suitable block structure and consensus mechanism, the global model for detection can be iteratively updated to enhance the detection performance. Moreover, the blockchain guarantees the trust environment of the network, making the detection algorithm resistant to internal attacks. Finally, compared with similar schemes in terms of performance, cost, security, etc., the results prove that BCEAD performs better.

1.1. Contributions.

(1) We propose an ensemble anomaly detection network structure for WSN. The structure has three levels, and high-performance nodes in the sink layers bear most of the storage and computing overhead required by the solution. Multiple sink nodes will independently detect anomalies in the local network based on the global anomaly detection model and optimize the model by submitting some parameters. This network structure is suitable for resource-constrained and heterogeneous WSNs and can be equipped with multiple types of anomaly detection algorithms to ensure network security.

(2) We design an anomaly detection model stored and updated by the blockchain. The detection node filters a batch of isolated trees stored in the blockchain to form an isolated forest for detection. In addition, new isolated trees are continuously generated and published to the blockchain, so that the isolated forest model, which is utilized to detect, is constantly updated. In this scheme, the detection model keeps dynamically updated with the environment, which enhances the detection performance while maintaining security.

2. Related Works

2.1. Anomaly Detection Structure. Intrusion detection is an important part of network security protection, and it is a network security technology that actively protects its network and system from illegal attacks. An intrusion detection system [11] usually consists of data collectors, data analyzers, alarm modules, and other parts. Traditional intrusion detection systems begin to feel weak with the expansion of the network scale and the complexity of information. Therefore, distributed intrusion detection systems and collaborative intrusion detection systems have been proposed. They not only analyze system logs but also analyze network traffic and introduce a distributed data collection mechanism into the structure. Subsequently, the anomaly detection structure for network attacks has gradually changed from local and centralized to a distributed structure. Specifically, the new detection system also deploys multiple data analyzers. Data analyzers in each network domain can communicate with each other and share detection models and strategies.

Distributed or collaborative anomaly detection has better detection performance for large-scale heterogeneous networks. However, the distributed anomaly detection has to consider the issue of trust among nodes. For example, the system may crash if a detection node in the detection system falsely sends alarms or maliciously updates the global detection model. Therefore, it is vital to build a trusted environment that can prevent nodes from internal attacks. In addition, distributed detection faces privacy issues. During the detection process, a large amount of data is collected and uploaded by the agent, which exposes the system to hidden dangers and threats of data leakage.

2.2. Anomaly Detection Method. Anomaly detection is the identification of events or observations that do not match the expected pattern. In different scenarios, anomalies are also called outliers, noises, deviations, etc. Isolation forest is a type of typical anomaly detection algorithm [12, 13], which distinguishes abnormal data with “few” and “different” characteristics. Compared with other classic classification methods, isolated forests consume fewer computing resources and can still maintain good performance when processing large amounts of high-dimensional data.

The isolated forest algorithm continuously divides the data by isolated trees, calculates the isolation score according to the height of the data point in the tree, and judges the anomaly according to the average isolated score. Suppose T is a node of an isolated tree, then T may be a leaf node, or an intermediate node with a decision threshold β and two child nodes (T_l, T_r). Assuming a dataset $X = \{x_1, \ldots, x_n\}$, each data x is a d-dimensional vector. By continuously selecting attributes $q \in d$ randomly and the decision threshold $\beta \in q$ to continuously classify $X' \subseteq X$, an isolated tree can be established. The isolated forest classifies the data by constructing a large number of random isolated trees. In general, data points classified into leaf nodes earlier may be more suitable for the definition of anomalies. Therefore, the isolation forest can quantify the degree of the anomaly of the data by the average tree length path of the data points.

Specifically, given a sample set with φ instances, the average path length of each isolated tree is c,

$$c(\varphi) = \begin{cases} 2H(\varphi - 1) - \dfrac{2(\varphi - 1)}{n} & \varphi > 0 \\ 1 & \varphi = 2 \\ 0 & otherwise \end{cases}, \quad (1)$$

where $H(i)$ is the harmonic number, which can be estimated with $\ln(i) + 0.5772156649$ (Eulerian constant). $c(\varphi)$ is the average $h(x)$ for a given φ. The anomaly score s of instance x is defined as

$$s(x, \varphi) = 2^{-(E(h(x))/c(\varphi))}, \quad (2)$$

where $E(h(x))$ is the average value of $h(x)$ in an isolated tree, which is the average height of the tree. From the results, the score of a sample close to 1 is judged to be abnormal; the score close to 0 is judged to be safe; and if the scores of all samples are 0.5, it means that the sample set has no obvious abnormality. However, the isolation forest algorithm also has shortcomings. Because it belongs to unsupervised learning, the algorithm is more dependent on the quality of the training set. In practical applications, it is necessary to ensure the real-time performance of the training set and continuously train and change the detection model to adapt to environmental changes and ensure detection performance.

2.3. Anomaly Detection and Blockchain. Blockchain is a new decentralized infrastructure and distributed computing paradigm that has gradually emerged with the increasing popularity of digital cryptocurrencies such as Bitcoin. At present, it has been highly valued and widely concerned by government departments, financial institutions, technology companies, and capital markets [14]. Blockchain is often understood as a data structure [15]. The block stores the data composed of Merkle tree and forms a chain through hash pointers, thereby ensuring that the data are difficult to be changed. In addition, the blockchain uses pure mathematical methods to establish trust relationships among distributed nodes to form a decentralized trusted distributed system, which has the characteristics of decentralization, network robustness, security, and credibility. In addition, the blockchain uses pure mathematical methods to establish trust relationships among distributed nodes to form a decentralized trusted distributed system, which has the characteristics of decentralization, network robustness, security, and credibility.

Therefore, the blockchain enables mutual trust between different participants [16] in the Internet of Things environment, which greatly reduces the cost of reshaping or maintaining trust for each node. For the trust problem of distributed detection, blockchain is also a kind of solution worth trying. Some schemes, which combine intrusion detection with blockchain to ensure trusted data sharing in distributed intrusion detection, have been proposed [10, 17, 18]. They store different data, such as detection characteristics, detection alarms, and detection results in the blocks, and publish to the blockchain network to share. Subsequently, a type of anomaly detection framework driven by blockchain on edge intelligence appeared [19]. The framework stores the data features to be analyzed on the blockchain, then the cloud-based detection model reads the data features on the blockchain for anomaly detection and feeds back the detection results. In addition, the framework transfers the overhead pressure for detection to the distributed edge network, which is more suitable for the system structure of the IoT. Therefore, the advantages of low detection delay and global model update brought by distributed detection could be supported by the blockchain. Recently, a scheme for detecting abnormal behavior in social networks [20] has been proposed, which combines isolation forest and blockchain technology. The authors claim that the blockchain can protect the privacy leakage problem in the anomaly detection process. They execute the isolated forest algorithm to detect data anomalies by the smart contracts, then marked and stored the abnormal data on a separate blockchain.

All the above detection schemes utilize the advantages of blockchain to solve a certain degree of security or privacy issues, but each has certain limitations. Specifically, they all store detection-related data on the blockchain. Although each solution has made optimizations to reduce the storage overhead, for example, the blockchain only stores the characteristic value or the hash value of the detection data. However, as the detection cycle lengthens, the blockchain will still face a storage bottleneck, resulting in much loss of detection performance. Therefore, we propose the BCEAD, which is different from the traditional scheme, to solve the storage problem and enhance the detection performance by storing the detection model.

3. A Blockchain-Based Ensemble Anomaly Detection

As shown in Figure 1, there is the multilayer network structure in BCEAD, which consists of the sensing layer, the sink layer, and the blockchain layer. The roles and functions of each layer are explained as follows:

Sensing Layer. The sensing layer contains a large number of low-cost, low-energy, and low-performance sensor nodes. The sensor node collects physical information from the external environment in real-time and converges it to the sink node. The collected data will be processed and used to generate the corresponding feature matrix for subsequent anomaly detection.

Sink Layer. The sink layer converges the environmental information collected by the sensors and submits it to the base station. Compared with sensor nodes, sink nodes have better computing and storage capabilities. Therefore, the sink node is responsible for most of the

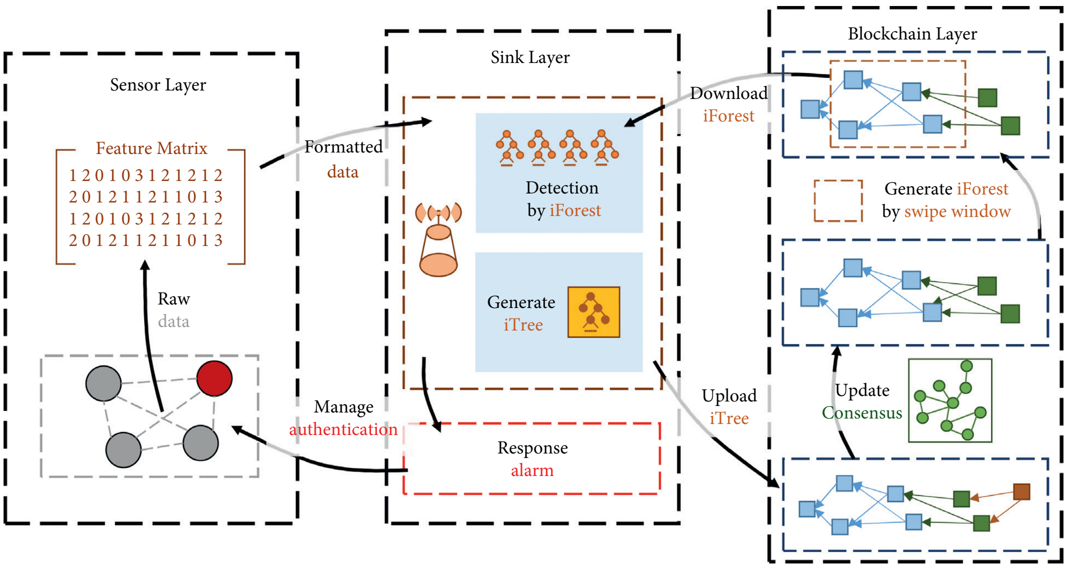


Figure 1: The framework of the BCEAD.

work of BCEAD. In detection, the sink nodes select a batch of the latest isolated trees from the blockchain by the sliding window algorithm. Then, an isolation forest is constructed to detect anomalies for the data submitted by the sensor layer. After detection, the sink node will contribute a new isolated tree to help optimize the global model. Finally, the sink node will respond according to the detection result.

Blockchain Layer. The blockchain layer is mainly responsible for maintaining a safe, usable, and constantly updated global detection model in a distributed network. The sink nodes keep verifying each released block according to the consensus mechanism to ensure that each update for blockchain is benign. The isolated forest, which is a global detection model, will be redundantly stored on the blockchain. Finally, the sliding windows mechanism will screen out suitable blocks to form the isolation forest model for anomaly detection.

3.1. Isolation Blocks

3.1. Isolation Blocks. In the detection, the sink node located in the sink layer utilizes the detection model stored in the blockchain layer to detect the data of the local sensor layer. The detection model (isolation forest) is composed of several isolated trees, so a single block in the blockchain is an isolated block containing an isolated tree. Then, the isolated blocks continue to increase, and the blockchain is periodically updated. Therefore, the detection model keeps iterative dynamically.

In BCEAD, the block format includes timeStamp, block ID, the hash value of current and previous blocks, node ID, and isolated tree,

$$Block = [timeStamp\|blockchainI\ D\|hashPre\|hashCur\|NID\|iTree]. \tag{3}$$

An isolated tree contains several nodes, each node contains the decision threshold β and the left node T_l and the right node T_r. The data evaluated by the tree will help the isolated tree to form its structure,

$$iTree = \left[iTree_{left}\|\beta\|iTree_{right}\right]. \tag{4}$$

As shown in Figure 2, the isolation tree *iTree* is the parameters submitted for each update of the global model. It is also the main content of the block released by the node after each round of consensus. This is the difference between BCEAD and previous solutions: BCEAD does not stores the detect-related data on the blockchain but instead stores the detection model. It avoids data privacy issues and reduces storage and communication overhead. However, the update of the global model can also be consensus because the global nodes can verify whether an isolation tree is appropriate.

3.2. Sliding Window

3.2. Sliding Window. The number of samples (isolated trees) used by the isolation forest will affect the detection performance of the model. The optimal value of the number of samples has been verified as $\varphi = 250$. BCEAD stores the detection model on the blockchain network, and the number of isolated blocks included in the blockchain keeps increasing. Therefore, the detection scheme has to filter the appropriate number of isolated blocks.

As shown in Figure 3, BCEAD sets up a sliding window algorithm to screen isolated trees for the detection model. The sliding window is essentially a fixed-size list, including a

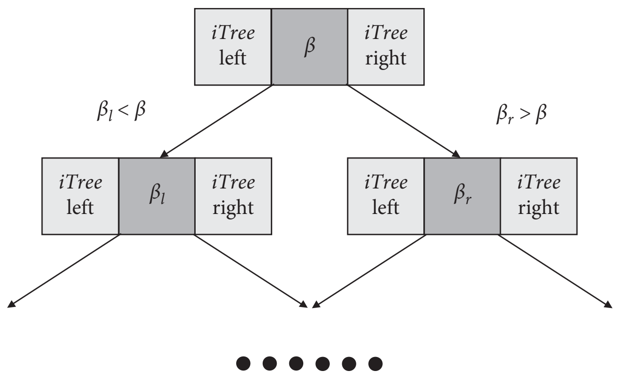


Figure 2: The structure of the isolated tree.

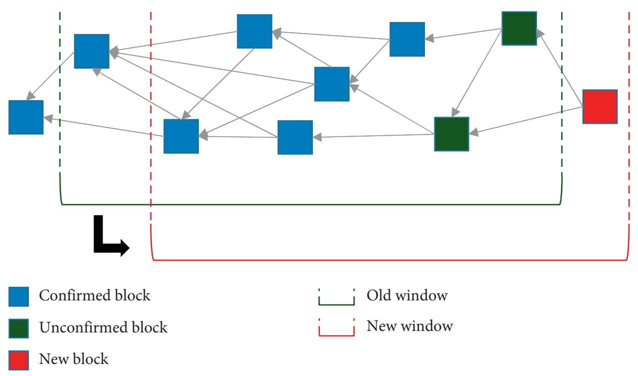


Figure 3: The sliding window for isolation forest.

series of block indexes. With the addition of new blocks, the total length of the blockchain will gradually increase, but the swiping window always traces back from the latest block and includes φ blocks. The update of the swiping window after each round of consensus in the blockchain network, each sink node will get a new slide window when synchronizing the latest blockchain.

Therefore, after detection, the sink node submits an isolated tree trained by the current data. The isolated tree will be packaged into blocks and released to the blockchain network. The isolated tree will be packaged into blocks and released to the blockchain network, and the model generated by each subsequent detection may utilize the isolated tree. This setting ensures the real-time update of the global detection model. Moreover, the training set used to generate the model is the current real-time detection data, which ensures the performance of the detection algorithm.

3.3. Block Forest. In the blockchain network, all nodes store a public ledger locally. During the continuous update of the ledger data, all nodes always communicate to reach consensus and ensure consistency. In BCEAD, the blockchain network uses IOTA (tangle) consensus. Compared with the traditional blockchain, the block in tangle is a single transaction. Each new transaction will verify and quote the previous two, so the network does not need to reach a consensus immediately [21]. Therefore, the network does not require miners, which avoids mining attacks such as in [22, 23]. Users can complete transaction verification by themselves, and the cost of the transaction is only the computational cost of verifying the other two transactions [24]. Therefore, the tangle blockchain can achieve mutual trust between nodes in distributed detection systems. It can both ensure security and considerable performance.

All sink nodes in BCEAD act as participants in the tangle, using isolated trees instead of blockchain transactions, and each block packs an isolated tree separately. In the block releasing, the new isolated tree will be packed into a specified structure block, and quote after the previous two verified blocks on the chain. The specific process is as follows:

(1) *Verify Block.* The sink node visits the tips list (maintains unconfirmed blocks in the network) and randomly selects blocks for verification. The verification will review the format of the block, the node ID of the publishing block, and its reference block. When a block is verified, the sink node puts it into the list, which contains references, and ends the verification work when the list has 2 elements.

(2) *Release Block.* The sink node refers to two blocks in the list, attaches the block ID, its ID, encrypts the block content with its private key, and calculates the hash value of the current block, then broadcasts the block to the blockchain network. Other nodes in the network will review the format and source of the new block, add it to the end of the tangle, and update tips. After several rounds of blockchain updates, subsequent references to a block will prove its credibility.

The application of a blockchain network for trusted communication among nodes can resist internal attacks because the blockchain network always verifies every message, even it comes from the trusted nodes. Therefore, attackers can only deplete the performance of the global detection model by publishing extreme or malicious isolated trees. Therefore, in the blockchain consensus, the verification includes querying the ID of the promulgator, and the purpose is to prohibit a node from frequently publishing the block. The experimental part in Section 4 proves that a single node publishing block has to set two rounds of consensus cycle cooling time, which can effectively prevent malicious nodes from destroying the model and ensure the benign performance of the detection model.

3.4. Algorithm Description. This paper proposes an isolation forest-based anomaly detection algorithm based on blockchain. The algorithm stores the isolation forest in the blockchain, and each detection will build a model based on the blockchain to detect the network data. The details are shown in Algorithm 1.

In this algorithm, first initialize the sink node SN (line 2). The sensor node runs $sni\,f\,ferPackets$ to capture the external environment information and obtain the unprocessed sensor information Raw_data (line 3). Through the feature extraction function $f\,eatureExtractor$, the feature matrix FM is obtained after processing the sensor data (line 4–6).

```
input: log files and data packets of the networks
output: response the detection and update the detection model
(1)  begin:
(2)  SN ← this;
(3)  Raw_data ← snifferPackets();
(4)  for all elements of Raw_data do
(5)      vector = featureExtractor(elements);
(6)      add(FM,bvector)
(7)  end
(8)  iForest = slideWindow(blockChain);
(9)  state = detection(FM, iForest);
(10)  if $state$ == False then
(11)      Response();
(12)  end
(13)  updateBlockchain();
(14)  end
```

Algorithm 1: B-iForest.

The sink node uses the algorithm $sli\ de\ Win\ do\ w$ to filter out suitable blocks from the blockChain to form an isolation forest *iForest* (line 8). Then, the sink node detects the feature matrix *FM* according the *iForest* and returns the detection result state (line 9). When the system detects an abnormality, that is, *state* = *False*, use $Res\,ponse$ to respond (line 10–11). Finally, the blockchain will be updated according to the update algorithm $up\ da\ teBlcokchain$, which is detailed in Algorithm 2 as follows:

In Algorithm 2, first initialize the number of samples of the detection algorithm, that is, the number of isolated trees φ (line 2). The algorithm *MCMC* filters out the latest and quotable block list $tips$ in the chain (line 4). According to the selection algorithm *Select*, the block to be quoted is selected from the list of references (line 6) and is verified (line 7), and the index of the verified block in the *List* (line 8) is recorded until the number of *List*'s elements reaches two (line 5). The isolated tree *iTree* to be submitted into blocks is packed and the two previous blocks that have been verified (line 11) is quoted. Finally, the packaged block to the entire network is broadcasted and the blockchain is updated (line 12).

4. Experiment

4.1. Data Processing. We implement related experiments of the proposed scheme through *Python*3.8. The isolation forest algorithm comes from the machine learning package Scikit-learn by *Python*. All experiments are executed on an x64 Windows10 personal computer using an Intel(R) Core (TM) i5-8500 CPU 3.00 GHz processor. We choose kddcup.data_10percent.gz in the popular KDD CUP′99 dataset in IDS research, that is, 10% of the dataset sampling. We select four typical attack samples with different attributes, $Buf\,fer_over\,flow$, $po\ d$, $guess_passws$, and $nmap$, remove the data labels, and mix them with normal samples to generate raw data for simulation. In the simulation process, samples are continuously sampled from the raw data to BCEAD for detection. The changes in the environment are simulated by controlling the abnormal proportions in the samples.

In this section, we conduct evaluation experiments through the following indicators:

Table 1 is the confusion matrix, where TP indicates that the real sample is positive and the prediction is also positive. FP indicates that the real sample is negative, but the prediction is positive. FN indicates that the real sample is positive, but the prediction is negative. TN means that the real sample is negative and the prediction is also negative. Some evaluation indicators, such as accuracy rate, precision rate, recall rate, and F1 value, can be obtained by the confusion matrix.

4.1.1. Accuracy. The ratio of the number of correctly classified samples to the total number of samples.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{5}$$

4.1.2. Precision. The probability that a sample predicted to be positive is indeed a positive sample.

$$Precision = \frac{TP}{TP + FP}. \tag{6}$$

4.1.3. Recall Rate. The probability of being correctly predicted as a positive sample among all positive samples is

$$Recall = \frac{TP}{TP + FN}. \tag{7}$$

4.1.4. F1 Value. An indicator of comprehensive precision rate and recall rate is

$$F1\_score = \frac{2 \times Recall \times Precision}{Recall + Precision}. \tag{8}$$

```
input: the iTree given by last detection
output: the newest detection model given by the blockchain
(1)  begin:
(2)  $\varphi \leftarrow$ this;
(3)  //estimatorSize, the size of
      slideWindow
(4)  tips ← MCMC();
(5)  while length(List)< 2 do
(6)     preBlock = Select(tips)
(7)     if $Verify(preBlock) == True$ then
(8)        add (List, preBlock)
(9)     end
(10) end
(11) curBlock = publish (List, iTree)
(12) blockchain = Brodcast (curBlock)
(13) end
```

Algorithm 2: Update Blockchain.

Table 1: Confusion matrix.

Label	Positive	Negative
Positive	True positive (TP)	False positive (FP)
Negative	False negative (FN)	True negative (TN)

4.2. Detection Performance. Jia et al. [25] proposed the connection between model generalization error and individual learners in ensemble learning. The upper limit of the generalization error is

$$PE \leq \frac{\overline{p}\left(1 - S^2\right)}{S^2}. \quad (9)$$

Here, $\overline{p}$ is the average value of all relevancies between every two classifiers, and S denotes the mean intensity of the individual classifier. Equation (9) shows that the generalization performance of the global model is better when the classification of individual classifiers is stronger, and the correlation between the classifiers is smaller. In EAD, isolated trees are generated independently by sink nodes during detection, so the low correlation between isolated trees ensures the detection performance of the global isolation forest.

As shown in Figure 4, some sampling points of the Nmap attack are drawn after being processed by PCA, and BCEAD can effectively distinguish between normal points and abnormal points. The ACU can reach 96.67%, and the F1 value is about 0.6 in the detection. It shows that BCEAD can effectively detect anomalies and distinguish between normal traffic and attack events in the network.

Figure 5 shows the accuracy of BCEAD's detection of various typical attacks. The detection performance of the detection model is maintained at a high level of 94% to 96%, and it has a good detection accuracy rate for all kinds of attacks. Figure 6 shows the false positive rate (FPR) of BCEAD detection of various typical attacks. The detection effect is worse for *Guess_passw d* and *Nmap* attacks, which is consistent with the response of the curve in Figure 5. The

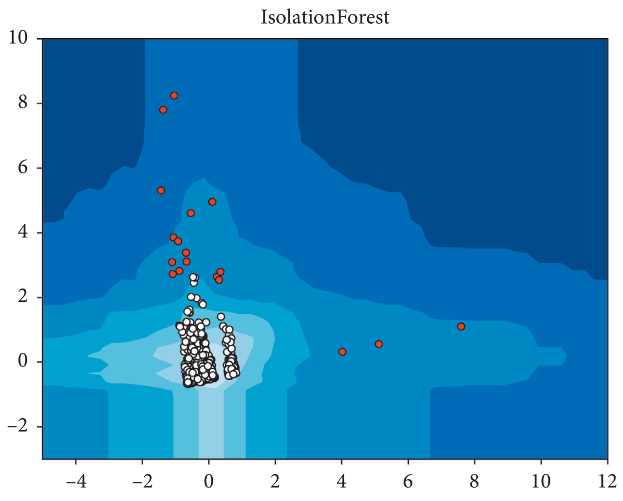


Figure 4: BCEAD detects two-dimensional data after PCA.

difference in the detection performance for different attacks is mainly due to the different launch frequency of various attacks. It confirms that the quality of the training set directly affects the performance of the detection model, and BCEAD, which updates the training data in real-time, can effectively guarantee its detection performance.

Figure 7 shows the $F1$ value changes of the scheme proposed by Liu et al. and BCEAD during a period. In the figure, the abscissa indicates the time by the ratio change of abnormal and normal data in the real-time detection. The blockchain-based anomaly detection proposed by Liu et al. does not have the update of the detection model, so the detection performance of the data is always constant in a

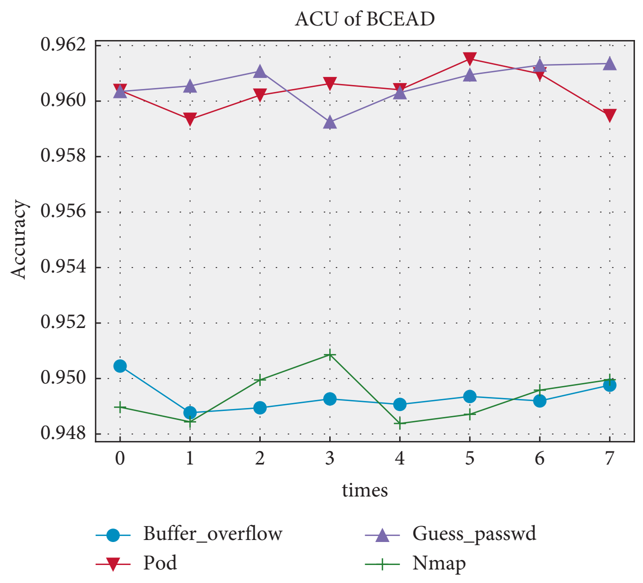


Figure 5: BCEAD's detection accuracy of the data after PCA.

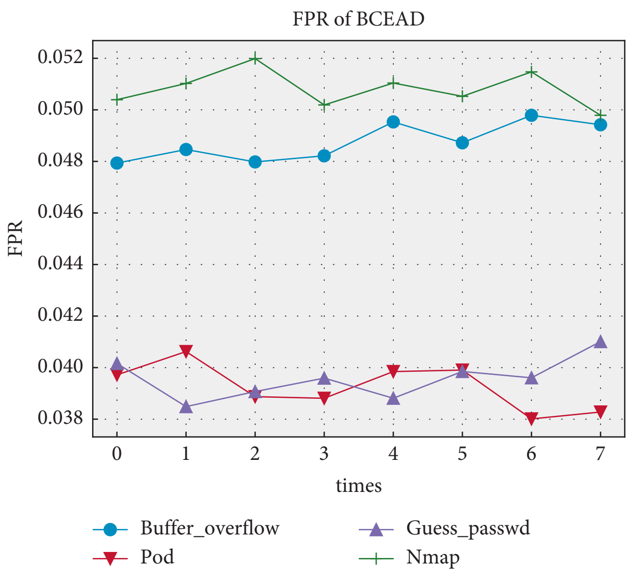


Figure 6: BCEAD's FPR of the data after PCA.

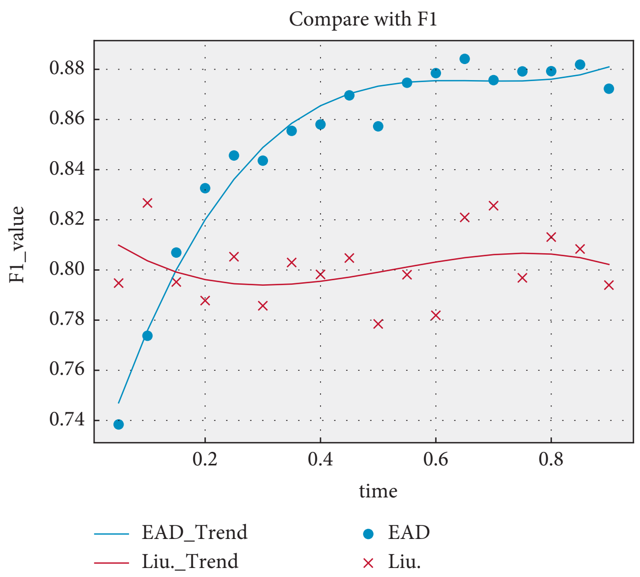


Figure 7: Performance comparison between BCEAD and similar scheme.

fixed range. However, BCEAD will dynamically select the training set to update the detection model in real-time. Therefore, the BECAD will optimize performance in long-term detection according to environmental changes.

Figure 8 shows the cost comparison between the scheme proposed by Liu et al. and BCEAD during the detection. The experiment specifically compared the communication overhead and storage overhead. As can be seen from the figure, because Liu et al.'s scheme stores detection data by the blockchain, the overhead will continue to grow during the long-term detection process. However, BCEAD uses the blockchain to store the detection model, so the cost in the early detection stage will be slightly higher, but the increase in cost is low. After a period of detection, the cost of BCEAD is significantly lower than similar blockchain-based detection schemes.

4.3. Security. Yang et al. proposed the security entropy to evaluate the benefits of system offense and defense [26], which can quantify the security of the system. We can suppose that the insecure factors caused by the attacker are $q_1, q_2, \ldots, q_n$, and the insecure entropy of the detection system at time t is $Q(t, q_1, q_2, \ldots, q_n)$, or simply $Q(t)$. If $Q(t)$ grows with time, that is, differential $dQ(t)/dt > 0$, then the system will become more and more unsafe. If $Q(t)$ decreases with time (differential $dQ(t)/dt < 0$), then the system will become more and more secure.

In BCEAD, the factors that affect the security of the system are generally classified into two: (1) the attacker publishes bad blocks, depletes the global model, and increases insecure entropy. (2) Ordinary nodes publish normal

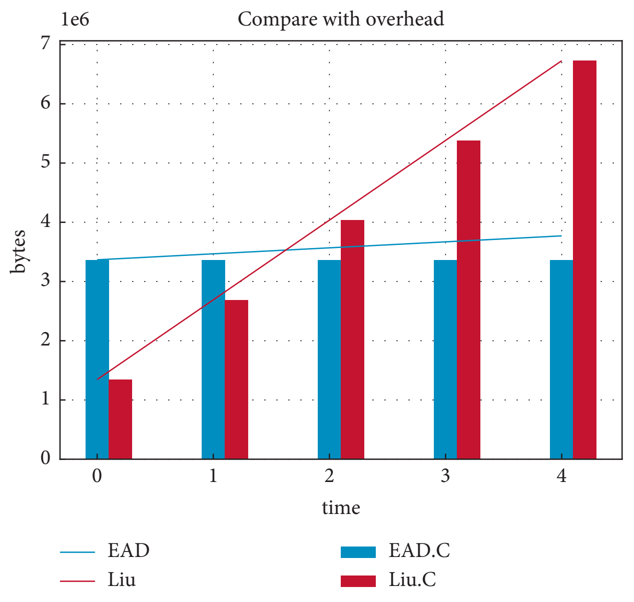


Figure 8: Overhead comparison between BCEAD and similar scheme.

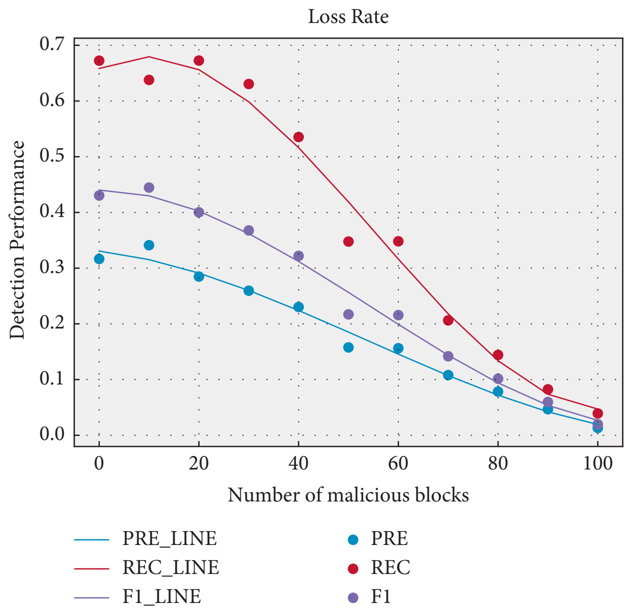


Figure 9: Changes in detection performance with the release of malicious blocks.

blocks, gaining the global model, and not increasing the insecure entropy. Then, the changes in the security entropy of the two factors are as follows:

$$\frac{dQ_1}{dt} = a_1 Q_1 \text{and} \frac{dQ_2}{dt} = a_2 Q_2. \tag{10}$$

Solving this system of equations,

$$Q_1 = c_1 e^{a_1 t} \text{and} Q_2 = c_2 e^{a_2 t}. \tag{11}$$

The time t is

$$t = \frac{\ln Q_1 - \ln c_1}{a_1} = \frac{\ln Q_2 - \ln c_2}{a_2}. \tag{12}$$

Set $a = a_1/a_2$, set $b = c_1/c_2^a$; then,

$$Q_1 = b(Q_2)^a. \tag{13}$$

In addition, it can be obtained that

$$\left\{\frac{dQ_1}{dt}\frac{1}{Q_1}\right\} : \left\{\frac{dQ_2}{dt}\frac{1}{Q_2}\right\} = a. \tag{14}$$

It can be seen from Equations (13) and (14) that the attacker's loss rate and the system's loss rate of the detection model is a power function with each other, and the ratio of the two to the model's loss change rate is a constant. In BCEAD, WSN's device authentication has prevented external attacks, and internal attackers can only undermine system security through the loss detection model. Because the blockchain guarantees the format of the network communication content, the attacker's loss model can only be done by publishing unhelpful isolated tree blocks. Therefore, restricting this single evil means can ensure the safety of the detection system.

The experimental results of malicious attacks on the loss of the system are shown in Figure 9. When malicious blocks continue to be released, the performance indicators of the detection system continue to decrease. It can be seen from the figure that after the release of 30 malicious blocks, the detection performance has dropped significantly. Since the BCEAD verification block is randomly selected, the probability of continuous bad blocks is low, and the submission of normal blocks has a gain effect on the detection model, so the solution itself has a certain attack resistance. From the above analysis, it can be seen that the ratio of the attacker's loss to the model and the program's gain to the model is constant, so this confrontation can find a balanced threshold. By disabling the node's continuous release of blocks, the security resistance of the detection system can be sufficient to resist malicious internal attacks.

5. Conclusion

This paper studies the security of wireless sensor networks, applies distributed anomaly detection to WSNs by the blockchain technology, and proposes the BCEAD scheme. The scheme divides the WSN into multiple layers. The sink layer performs anomaly detection for each network domain in the sensor layer. The detection model (isolation forest) is stored in the blockchain (tangle). Besides, this paper compares and analyzes the detection performance of BCEAD through experiments and proves its superiority. However, the actual deployment of the blockchain may affect the performance of the detection system, and the communication and storage overhead of the blockchain technology is also difficult to be balanced. Therefore, we will do some practice for the blockchain in the future. Nevertheless, the experiment proved that the scheme is compatible with some existing security mechanisms [27] for detection, which is enough to guarantee the application potential of the scheme.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Natural Science Foundation under Grant no. 61962009, Major Scientific and Technological Special Project of Guizhou Province under Grant no. 20183001, Science and Technology Support Plan of Guizhou Province ([2020]2Y011), and Foundation of Guangxi Key Laboratory of Cryptography and Information Security (GCIS202118).

References

[1] L. Qiang, H. Xiaohong, L. Supeng, L. Longjiang, and M. Yuming, "Deployment strategy of wireless sensor networks for internet of things," *China Commun*, vol. 8, pp. 111–120, 2011.

[2] N. Marriwala and P. Rathee, "An approach to increase the wireless sensor network lifetime, Proc. 2012 World Congr," *Inf. Commun. Technol. WICT*, vol. 2012, pp. 495–499, 2012.

[3] C. Mahmoud and S. Aouag, "Security for internet of things: a state of the art on existing protocols and open research issues," *ACM Int. Conf. Proceeding Ser.*vol. 17, pp. 1294–1312, 2019.

[4] Y. Chen, J. Sun, Y. Yang, T. Li, X. Niu, and H. Zhou, "PSSPR: a source location privacy protection scheme based on sector phantom routing in WSNs," 2021, https://arxiv.org/abs/2109.13774.

[5] A. M. FSabahi, "Intrusion detection: a survey," in *Proceedings of the 3rd Int. Conf. Syst. Networks Commun. ICSNC 2008 - Incl. I-CENTRIC 2008 Int. Conf. Adv. Human-Oriented Pers. Mech. Technol. Serv.*, pp. 23–26, Sliema, Malta, October 2008.

[6] J. J. Davis and A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: a review," *Computers & Security*, vol. 30, no. 6-7, pp. 353–375, 2011.

[7] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[8] F. Huang, G. Xie, and R. Xiao, "Research on ensemble learning," in *Proceedings of the 2009 Int. Conf. Artif. Intell. Comput. Intell. AICI*, pp. 249–252, Shanghai, China, November 2009.

[9] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[10] B. Hu, C. Zhou, Y.-C. Tian, Y. Qin, and X. Junping, "A collaborative intrusion detection approach using blockchain for multimicrogrid systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 8, pp. 1720–1730, 2019.

[11] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: a comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[12] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *Proceedings of the IEEE Int. Conf. Data Mining, ICDM.*, pp. 413–422, Pisa, December 2008.

[13] F. T. Liu and K. M. Ting, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, pp. 1–44, 2018.

[14] Y. Yuan and F. Y. Wang, "Blockchain: the state of the art and future trends," *Zidonghua Xuebao/Acta Autom. Sin.*vol. 42, pp. 481–494, 2016.

[15] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *SSRN Electronic Journal*, 2008.

[16] T. Li, Y. Chen, Y. Wang et al., "Rational protocols and attacks in blockchain system," *Security and Communication Networks*, vol. 2020, pp. 1–11, 2020.

[17] W. Li, S. Tug, W. Meng, and Y. Wang, "Designing collaborative blockchained signature-based intrusion detection in IoT environments," *Future Generation Computer Systems*, vol. 96, pp. 481–489, 2019.

[18] W. Li, Y. Wang, J. Li, and M. H. Au, "Toward a blockchain-based framework for challenge-based collaborative intrusion detection," *International Journal of Information Security*, vol. 20, no. 2, pp. 127–139, 2021.

[19] X. Xie, Y. Fang, Z. Jian, Y. Lu, T. Li, and G. Wang, "Blockchain-driven anomaly detection framework on edge intelligence," *CCF Transactions on Networking*, vol. 3, no. 3-4, pp. 171–192, 2020.

[20] X. Liu, F. Jiang, and R. Zhang, "A new social user anomaly behavior detection system based on blockchain and smart contract," *IEEE Int. Conf. Networking, Sens. Control. ICNSC*, vol. 2020, 2020.

[21] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in IoT: the challenges, and a way forward," *Journal of Network and Computer Applications*, vol. 125, pp. 251–279, 2019.

[22] T. Li, Z. Wang, G. Yang, Y. Cui, Y. Chen, and X. Yu, "Semi-selfish mining based on hidden Markov decision process," *International Journal of Intelligent Systems*, vol. 36, no. 7, pp. 3596–3612, 2021.

[23] T. Li, Z. Wang, Y. Chen, C. Li, Y. Jia, and Y. Yang, "Is semi-selfish mining available without being detected?" *International Journal of Intelligent Systems*, 2021.

[24] W. F. Silvano, R. Marcelino, and I. Tangle, "Iota Tangle: a cryptocurrency to communicate Internet-of-Things data," *Future Generation Computer Systems*, vol. 112, pp. 307–319, 2020.

[25] B. Jia and Y. Liang, "Anti-D chain: a lightweight DDoS attack detection scheme based on heterogeneous ensemble learning in blockchain," *China Communications*, vol. 17, no. 9, pp. 11–24, 2020.

[26] Y. X. Yang and X. X. Niu, *The General Theory of Information Security*, Publishing House of Electronics Industry, Beijing, China, 2018.

[27] W. Meng, W. Li, and L.-F. Kwok, "EFM: enhancing the performance of signature-based network intrusion detection systems using enhanced filter mechanism," *Computers & Security*, vol. 43, pp. 189–204, 2014.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 3116593, 12 pages
https://doi.org/10.1155/2021/3116593

Review Article

Attacks and Solutions for a Two-Factor Authentication Protocol for Wireless Body Area Networks

Chien-Ming Chen,[1] **Zhen Li**,[1] **Shehzad Ashraf Chaudhry**,[2] **and Long Li**[3]

[1]*College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China*
[2]*Department of Computer Engineering, Istanbul Gelisim University, Istanbul, Turkey*
[3]*Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Gullin, China*

Correspondence should be addressed to Long Li; lilong@guet.edu.cn

Received 1 September 2021; Accepted 4 October 2021; Published 21 October 2021

Academic Editor: Azees M

As an extension of the 4G system, 5G is a new generation of broadband mobile communication with high speed, low latency, and large connection characteristics. It solves the problem of human-to-thing and thing-to-thing communication to meet the needs of intelligent medical devices, automotive networking, smart homes, industrial control, environmental monitoring, and other IoT application needs. This has resulted in new research topics related to wireless body area networks. However, such networks are still subject to significant security and privacy threats. Recently, Fotouhi et al. proposed a lightweight and secure two-factor authentication protocol for wireless body area networks in medical IoT. However, in this study, we demonstrate that their proposed protocol is still vulnerable to sensor-capture attacks and the lack of authentication between users and mobile devices. In addition, we propose a new protocol to overcome the limitations mentioned above. A detailed comparison shows that our proposed protocol is better than the previous protocols in terms of security and performance.

1. Introduction

Since the beginning of human civilization, the efficient and fast transmission of information has always been an unswerving pursuit for mankind. From writing to printing, from cell towers to radio, from telephones to mobile Internet, the speed of modern technology development has always depended on the speed of information dissemination, and new ways of information dissemination often bring about radical changes in society. 5G (fifth-generation mobile communication technology) is the current stage of progress in the latest wave of mobile communication [1]. 5G is a new generation of broadband mobile communication with high speed, low latency, and large connection characteristics. It is a network infrastructure that enables the interconnection of people, machines, and things. 5G has three major application scenarios: enhanced mobile broadband, ultra-high reliability and low-latency communications, and massive machine-like communications. Enhanced mobile broadband mainly responds to the explosive growth of Internet traffic, and it results in improved user experience for mobile Internet users. Low-latency communication is mainly for applications with high requirements for latency and reliability, such as telemedicine, autonomous driving, and virtual reality. Massive machine-like communication is mainly for applications that involve the sensing and collection of data, such as Internet of Things (IoT) [2–4], smart cities [5–7], smart homes, and environmental monitoring [8–10].

In the long run, consumer demand for health will continue to rise, and the development potential of the medical and health fields is huge. Currently, 5G is particularly useful for the healthcare sector, especially for the Internet of Things in the medical field [11–13]. 5G will empower the existing smart healthcare service system, and it will improve the service capability and management efficiency of wireless body area networks, telemedicine, and emergency rescue. It will also give rise to the development and prosperity of smart healthcare.

Owing to rapid advancements in life informatization, people's requirements for medical monitoring are constantly improving. There is also a high demand for more convenient and effective telemedicine and health-sign monitoring. A wireless body area network (WBAN) [14, 15] is a network composed of different intelligent components, such as sensors, nodes, and actuators. The network is designed for collecting and monitoring data from the human body and its surrounding environment. Its typical architecture is shown in Figure 1. For the elderly, sensors/wearable devices on the elderly send the information collected to a gateway node. For the patient, the sensor acquires the patient's body monitoring data, connects it to a bedside monitor or other receiver, and transmits it wirelessly to a doctor for monitoring or diagnosis. The gateway acts as a local server which analyzes, stores, and manages the data sent by the sensor or monitor. Users, who can be doctors, nurses, or other medical professionals, can communicate with the gateway and access the data they want to know via mobile devices or computer-based devices on a LAN with the gateway. For example, a nurse can specifically track and check a patient's body data, so that if an abnormality is detected, the patient's condition can be checked and dealt with in a timely manner.

Because data transmission over a WBAN takes place over a public channel, attackers can access highly sensitive health information of patients. To ensure the security of a WBAN, a secure authentication and key agreement (AKA) protocol should be implemented before communication. Numerous AKA protocols have been proposed [16–21]. However, many of these AKA protocols have proven to be insecure against many types of attacks. Recently, Fotouhi et al. [22] proposed a lightweight and secure two-factor AKA protocol for WBANs in the healthcare-based IoT. They claimed that their proposed protocol is secure against many attacks, such as key disclosure simulation attacks, special session temporary information attacks, and offline password guess attacks.

In this study, we first demonstrate that Fotouhi et al.'s proposed protocol [22] is still vulnerable to sensor-capture attacks. Additionally, their proposed protocol fails to provide authentication between users and mobile devices. To overcome these security pitfalls, we propose a secure and efficient AKA protocol for WBANs. The security analysis shows that our proposed protocol is secure. We also provide a detailed comparison to demonstrate that our proposed protocol achieves improved efficiency and security.

The remainder of this paper is organized as follows. In Section 2, we briefly review the authentication protocol proposed by Fotouhi et al. In Section 3, we provide a reasonable cryptanalysis of Fotouhi et al.'s proposed protocol. In Section 4, we propose a new protocol for improving the flaws in the old protocol. In Section 5, we perform a security analysis, which includes both formal and informal analyses, to demonstrate the security and stability of our proposed protocol. In Section 6, we analyze the security and performance of our proposed protocol in terms of security, performance, and communication cost. Finally, we provide the conclusions to this study.

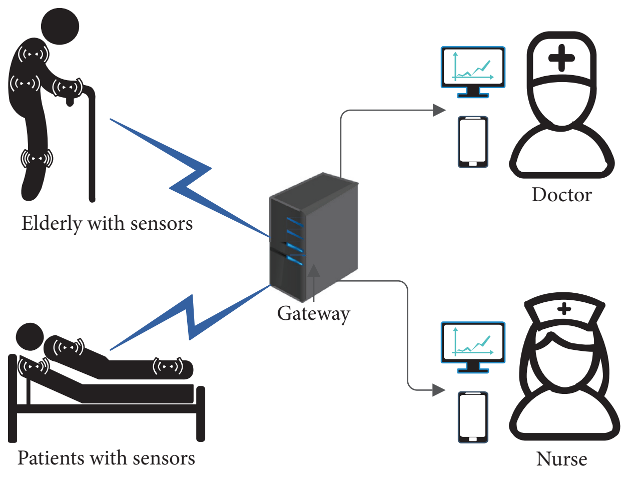


Figure 1: The typical architecture of a WBAN.

2. Review of Fotouhi et al.'s Protocol

In this section, we briefly review Fotouhi et al.'s authentication protocol. Their proposed protocol includes four phases: initialization, registration, authentication, and password modification. Here we describe only the first two phases. The detailed steps of their proposed protocol can be found in [22]. The notations used in this study are listed in Table 1.

2.1. Sensor Node Registration

2.1. Sensor Node Registration. In this phase, the corresponding gateway injects the necessary information into each sensor node. We assume that a gateway GW_j is the corresponding gateway of SN_k. GW_j generates two random numbers, R_y and R_z, after which it injects $\{SID_k, SG_k, QID_k, GID_j, R_y, R_z\}$ into the memory of SN_k, where $SG_k = h(SID_k \| G_j \| N_l)$. GW_j also stores $\{SID_k, N_l, QID_k, R_y, h(R_z)\}$ in its database.

2.2. User Registration. Assuming that a user, U_i, desires to register to GW_j, the following steps are performed:

Step 1: U_i sends ID_i and HPW_i to GW_j through a secure channel, where $HPW_i = h(PW_i \| R_0)$.

Step 2: if U_i is an unregistered user, GW_j generates a pseudoidentity CID_i and a random number R_x, and it stores $\{ID_i, HPW_i, CID_i, R_x\}$ in GW_j's database. GW_j then calculates $A_1 = h(CID_i \| R_x \| GID_j \| GID_j) \oplus HPW_i$ and $A_2 = h(ID_i \| G_j) \oplus h(ID_i \| HPW_i)$, after which it sends $\{CID_j, GID_j, A_1, A_2\}$ to U_i through a secure channel.

Step 3: U_i calculates $A_3 = h(ID_i \| PW_i) \oplus R_0$, after which it stores $\{CID_i, GID_j, A_1, A_2, A_3\}$ in the mobile device.

2.3. Authentication Phase. Assuming that U_i desires to communicate with SN_k, the following steps are performed:

Table 1: Notations table.

Symbol	Description
$U_i, \mathrm{ID}_i, \mathrm{PW}_i$	i-th user, his/her identity, his/her password
$\mathrm{GW}_j, \mathrm{GID}_j, G_j$	j-th gateway, its identity, its secret key
$\mathrm{SN}_k, \mathrm{SID}_k$	k-th sensor, its identity
N_l	Network identifier of the sensor set
SG_k	Shared key between sensor and gateway
SK_u	Session key generated by user
SK_g	Session key generated by gateway
SK_s	Session key generated by user
M_i	i-th message
$\mathrm{CID}_i, \mathrm{QID}_k$	Temporary pseudoidentity of U_i and SN_k
$R_s, R_0, R_u, R_g, R_x, R_y, R_z$	Temporary random number
$\mathrm{Gen}(\cdot)$, $\mathrm{Rep}(\cdot)$	Biometric extraction function, decryption function
BIO_i	Biometric information of the i-th user
$h(\cdot)$	Hash function
$\oplus$	Bitwise XOR operation
$\Vert$	Concatenate operation

Step 1: U_i generates a random number, R_u, after which it calculates $R_0 = A_3 \oplus h(\mathrm{ID}_i \Vert \mathrm{PW})$, $\mathrm{HPW}_i = h(\mathrm{PW}_i \Vert R_0)$, $B_1 = A_1 \oplus \mathrm{HPW}_i$, $B_2 = B_1 \oplus \mathrm{HPW}_i \oplus R_u$, $B_3 = \mathrm{SID}_k \oplus H(\mathrm{ID}_i \Vert R_u)$, and $B_4 = h(\mathrm{CID}_i \oplus \mathrm{GID}_j \oplus \mathrm{SID}_k \oplus B_1 \oplus \mathrm{ID}_i \oplus R_u)$. Afterwards, U_i transmits M_1 to GW_j, where $M_1 = \{\mathrm{CID}_i, \mathrm{GID}_j, B_2, B_3, B_4\}$.

Step 2: GW_j obtains the corresponding ID_i, R_x, and HPW_i from its database. GW_j then calculates $B_1 = h(\mathrm{CID}_i \Vert R_x \Vert \mathrm{GID}_j \Vert G_j)$ and $R_u = B_2 \oplus B_1 \oplus \mathrm{HPW}_i$, after which it verifies the correctness of B_4. GW_j then generates two random numbers, R_g and R_z', obtains SID_k with B_3, obtains R_y from its database, and generates a new pseudonym QID_k'. GW_j then calculates $\mathrm{SG}_k = h(\mathrm{SID}_k \Vert G_j \Vert N_l)$, $S = h(\mathrm{SG}_k \Vert \mathrm{GID}_j)$, $B_5 = (R_u \oplus \mathrm{HPW}_i) \oplus S \oplus R_y$, $B_6 = R_g \oplus S \oplus \mathrm{SID}_i \oplus R_y$, $B_7 = \mathrm{QID}_k' \oplus R_g \oplus R_y$, $B_8 = h(R_g \Vert R_y \Vert S) \oplus R_z'$, and $B_9 = h(\mathrm{QID}_k \Vert B_7 \Vert B_8 \Vert \mathrm{SG}_k \Vert R_u \oplus \mathrm{HPW}_i \Vert R_g)$. Afterwards, GW_j transmits $\{\mathrm{QID}_k, B_5, B_6, B_7, B_8, B_9\}$ to SN_k.

Step 3: SN_k verifies the correctness of QID_k. If it is correct, SN_k calculates $S = h(\mathrm{SG}_k \Vert \mathrm{GID}_j)$, $(R_u \oplus \mathrm{HPW}_i) B_5 \oplus S \oplus R_y$, and $R_g = B_6 \oplus S \oplus \mathrm{SID}_k \oplus R_y$. If B_9 is correct, SN_k generates a random number, R_s, and it calculates $R_z' = h(R_g \Vert R_y \Vert S) \oplus B_8$, $\mathrm{QID}_k' = B_7 \oplus R_g \oplus R_y$, and $B_{10} = R_g \oplus S \oplus R_z$. SN_k then stores QID_k', R_z', and $R_y' = h(R_y)$, and it calculates $\mathrm{SK}_s = h(R_u \oplus \mathrm{HPW}_i \Vert R_g \Vert R_s)$. It then calculates $B_{11} = h(\mathrm{SG}_k \Vert R_g) \oplus h(R_y) \oplus R_s$ and $B_{12} = h(B_{10} \Vert B_{11} \Vert \mathrm{SK}_s \Vert \mathrm{SID}_k \Vert \mathrm{GID}_j \Vert R_s)$, after which it transmits $\{B_{10}, B_{11}, B_{12}\}$ to GW_j.

Step 4: GW_j calculates $R_y' = h(R_y)$ and $R_z' = R_g \oplus S \oplus B_{10}$. It then verifies whether $h(R_z)$ is equal to $h(R_z')$. If the verification is passed, it calculates $R_s = B_{11} \oplus h(\mathrm{SG}_k \Vert R_g) \oplus R_y'$ and obtains the session key $\mathrm{SK}_g = h(R_u \oplus \mathrm{HPW}_i \Vert R_g \Vert R_s)$. It further verifies the correctness of B_{12}, generates a new CID_i' for U_i, stores QID_k' and R_z', and replaces R_y' and $h(R_x)$ with R_y and R_x, respectively. It then calculates $B_{13} = h(\mathrm{CID}_i' \Vert h(R_x) \Vert \mathrm{GID}_j \Vert G_j) \oplus h(R_u \Vert \mathrm{HPW}_i)$, $B_{14} = h(R_u \Vert \mathrm{ID}_i) \oplus R_g$, $B_{15} = h(R_u \Vert R_g \Vert \mathrm{HPW}_i) \oplus R_s$, $B_{16} = h(h(\mathrm{ID}_i \Vert G_j) \Vert R_s) \oplus \mathrm{CID}_i'$, and $B_{17} = h(\mathrm{SK}_g \Vert \mathrm{ID}_i \Vert B_{13} \Vert \mathrm{CID}_i')$. GW_j then generates $\{B_{13}, B_{14}, B_{15}, B_{16}, B_{17}\}$ and transmits it to U_i.

Step 5: U_i calculates $R_g = B_{14} \oplus h(R_u \Vert \mathrm{ID}_i)$, $R_s = B_{15} \oplus h(R_u \Vert R_g \Vert \mathrm{HPW}_i)$, and $\mathrm{CID}_i' = B_{16} \oplus h((A_2 \oplus h(ID_i \Vert HPW_i)) \Vert R_s)$. U_i then calculates the session key $\mathrm{SK}_u = h(R_u \oplus \mathrm{HPW}_i \Vert R_g \Vert R_s)$ and verifies B_{17}. When the verification is passed, U_i calculates $A_1' = B_{13} \oplus h(R_u \Vert \mathrm{HPW}_i)$ and stores CID_i' and A_1'.

3. Cryptanalysis of Fotouhi et al.'s Protocol

This section shows that Fotouhi et al.'s protocol [22] is vulnerable to sensor-capture attacks and a lack of authentication between users and mobile devices.

3.1. Threat Model.

3.1. Threat Model. The attacker model briefly describes the capabilities of an attacker. In this study, we use the $D-Y$ model [23–25] and assume that the attacker is A. The detailed capabilities are as follows:

(1) A can eavesdrop and intercept information transmitted by public channels and can forge, delete, replay, and tamper with such information

(2) A can extract the information from the captured sensor nodes

(3) A can access the information stored in the gateway

3.2. Sensor-Capture Attack. Assuming that A captures SN_k and obtains $\{\mathrm{SID}_k, \mathrm{SG}_k, \mathrm{GID}_j, R_y, R_z, \mathrm{QID}_k\}$ in the memory of sensor SN_k, A can calculate the session key SK through the following steps:

Step 1: calculate $S = h(\mathrm{SG}_k \| \mathrm{GID}_j)$, and then obtain $(R_u \oplus \mathrm{HPW}_i)$ by calculating $B_5 \oplus S \oplus R_y$

Step 2: obtain R_g by calculating $B_6 \oplus S \oplus \mathrm{SID}_k \oplus R_y$

Step 3: obtain R_s by calculating $h(\mathrm{SG}_k \| R_g) \oplus h(R_y) \oplus B_{11}$

Therefore, A can calculate the correct session key $\mathrm{SK} = h(R_u \oplus \mathrm{HPW}_i \| R_g \| R_s)$ shared among U_i, GW_j, and SN_k.

3.3. Lack of Authentication between Users and Mobile Devices. Assuming that an attacker A captures U_i's mobile device, A performs the following steps:

Step 1: because A does not know PW_i, A randomly generates PW_i' and then inputs ID_i and PW_i' to the captured mobile device. The mobile device calculates and transmits M_1 with the fake password PW_i' to GW_j.

Step 2: GW_j verifies GID_j and CID_i, after which it calculates B_1 and R_u. Afterwards, GW_j attempts to verify the correctness of B_4, and GW_j realizes that M_1 sent from U_i is not legal.

Essentially, A does not need to capture a mobile device because the attacker can eavesdrop the M_1 between any user and GW_j and then send M_1 to GW_j.

The scenario mentioned above illustrates two weaknesses in Fotouhi et al.'s proposed protocol. First, the mobile device does not verify the password that a user inputs. Regardless of whether the password or account number entered by U_i is correct, the mobile device sends all the necessary messages to GW_j. Second, GW_j calculates B_1 and R_u before verifying B_4. Owing to the limited computing power of a gateway, if an attacker has been sending a large number of error messages to a gateway through multiple mobile devices, the gateway may be paralyzed and unable to respond to the requests of other users, which will result in immeasurable losses in medical Internet environments.

4. The Improved Protocol

In this section, we present an enhanced lightweight and secure two-factor authentication protocol (AELSA) for medical IoT and WBANs to address and enhance the outstanding vulnerabilities and fragile shortcomings of Fotouhi et al.'s protocol. AELSA also applies to the WBAN architecture and includes three main participants: (*a*) the physician or nurse as the user, (*b*) the gateway node as the server, and (*c*) as the sensor. The sensors can include the dynamic collection of patient data for real-time data. On the other hand, the gateway represents a server, which acts as an authentication and data-delivery center for ensuring mutual authentication between the physician and the sensor. The physician or nurse, as the user, can access the information from the sensor, which is delivered using the gateway through a device, such as a mobile device or a computer that can log into the system. AELSA comprises four main phases: (*a*) initialization, (*b*) registration, (*c*) login, and (*d*) mutual authentication and key exchange phases. The registration phase includes the user registration and sensor registration phases. The symbols used are also listed in Table 1.

4.1. Initialization Phase. We assume that all the gateways are considered trusted parts, the gateways are identified through GID_j when transmitting messages, and the gateways generate G_j as their private key during initialization. In this phase, important parameters and functions of the system are generated and published, such as initializing the stored information within the gateway.

4.2. Registration Phase. This phase comprises a sensor node enrollment phase and a user enrollment phase with the following steps.

4.2.1. Sensor Node Enrollment. In the sensor registration phase of AELSA, if a new sensor SN_k wants to join the WBAN, it must interact with the data and submit registration information to the gateway GW_j. First, SN_k sends its SID_k and N_l to GW_j over a secure channel. After GW_j receives the message, it determines whether SID_k is a new identity and generates a new pseudoidentity QID_k for SN_k if it is a new identity. Next, it computes SG_k as a shared key for SN_k and GW_j, where $\mathrm{SG}_k = h(\mathrm{SID}_k \| G_j \oplus N_l)$, and it stores $\{\mathrm{QID}_k, N_l\}$ into the memory. Afterwards, GW_j securely sends $\{\mathrm{SG}_k, \mathrm{QID}_k\}$ to SN_k. Once SN_k receives the message, it encrypts SG_k using its SID_k, $\mathrm{RSG}_k = \mathrm{SG}_k \oplus \mathrm{SID}_k$, and it stores $\{\mathrm{RSG}_k, \mathrm{QID}_k\}$.

4.2.2. User Enrollment. In this stage, the user completes the registration in GW_j based on the generation function of the bioinformation embedded in the mobile device as well as other information. The user enters their identity ID_i, password PW_i, and bioinformation BIO_i on the mobile device. The mobile device then generates σ_i and τ_i using the generation function Gen. It uses σ_i to mask and protect PW_i, calculates $\mathrm{HPW}_i = h(\mathrm{PW}_i \| \sigma_i)$, and sends $\{\mathrm{ID}_i, \mathrm{HPW}_i\}$ to GW_j on the anti-interference channel. Upon receiving $\{\mathrm{ID}_i, \mathrm{GW}_j\}$ determines whether the identity is new. A new identity represents an unregistered identity. If it is new, it then calculates $\mathrm{CID}_i = h(\mathrm{ID}_i)$ and stores $\mathrm{CID}_i, \mathrm{HPW}_i$. It then selects a secret random number R_0 and computes $A_1 = h(\mathrm{CID}_i \| \mathrm{GID}_j \| R_0 \oplus G_j) \oplus \mathrm{Hpw}_i$ and $A_2 = h(\mathrm{GID}_j \| \mathrm{HPW}_i) \oplus (R_0 \oplus G_j)$, which, in turn, store A_1 into memory. It then transmits the secure message $\{A_2, \mathrm{GID}_j\}$ to U_i over the private channel. After U_i receives the secure message, it computes $A_3 = h(ID_i \| \mathrm{HPW}_i)$ and stores $\{A_2, A_3, \mathrm{GID}_j, \mathrm{Gen}(.), \mathrm{Rep}(.), an\ d\ \tau_i\}$, where Rep can decrypt σ_i using the biological information BIO_i and τ_i.

4.3. Login Phase. Compared to the protocol proposed by Fotouhi et al., AELSA adds a login phase in which the mobile device verifies the legitimacy of U_i's identity and effectively prevents the consumption of redundant functions resulting from the nonuse of authentication. It is assumed that when U_i logs into the mobile device, U_i enters ID_i^* and PW_i^* and enters biological information BIO_i^*, such as the fingerprint and iris. The mobile device calculates $\mathrm{Rep}(\mathrm{BIO}_i^*, \tau_i)\sigma_i^*$, $\mathrm{HPW}_i^* = h(\mathrm{PW}_i^* \| \sigma_i^*)$, and $A_3^* = h(\mathrm{ID}_i^* \| \mathrm{HPW}_i)$. It then verifies A_3 by comparison. If $A_3 = A_3^*$, the mobile device allows U_i to log in. Otherwise, it denies U_i to log into the system and sends an alert. Figure 2 shows the detailed process of the user login phase.

4.4. Mutual Authentication and Key Exchange Phase. In the key exchange phase, the user, gateway, and sensor negotiate to create a three-way trusted key for ensuring the correctness and security of future messages. This phase comprises five steps, as described below. Among other things, Figure 3 shows the stages of mutual authentication and key exchange.

Step 1: user U_i selects the SID_k of the sensor to be accessed, generates a random number R_u, and creates a timestamp T_1. U_i computes $(R_0 \oplus G_j) = A_2 \oplus h(\mathrm{GID}_j \| \mathrm{HPW}_i)$, $B_1 = \mathrm{SID}_k \oplus h(\mathrm{GID}_j \| \mathrm{HPW}_i)$, $B_2 = R_u \oplus h(\mathrm{GID}_j \| \mathrm{HPW}_i \oplus SID_k)$, and $B_3 = (R_0 \oplus G_j) h(\mathrm{GID}_j \| R_u)$, after which U_i transmits the message M_1 $\{\mathrm{CID}_i, \mathrm{GID}_j, B_1, B_2, B_3, T_1\}$ to the gateway GW_j.

Step 2: after receiving the message M_1, GW_j verifies the legitimacy of T_1 by determining whether it matches $|T_1 - T_C|\Delta T$. GW_j searches and obtains the corresponding HPW_i and QID_k in the memory based on CID_i in M_1. Afterwards, GW_j computes $\mathrm{SID}_k = B_1 \oplus h(\mathrm{GID}_j \| \mathrm{HPW}_i)$, $R_u = B_2 \oplus h(\mathrm{GID}_j \| \mathrm{HPW}_i \oplus \mathrm{SID}_k)$, $(R_0 \oplus G_j) = B_3 \oplus h(\mathrm{GID}_j \| R_u)$, and $A_1^* = h(\mathrm{CID}_i \| \mathrm{GID}_j \| R_0 \oplus G_j) \oplus \mathrm{HPW}_i$, and it verifies $A_1 \overset{?}{=} A_1^*$. If the verification fails, GW_j aborts the conversation. Otherwise, GW_j confirms the legitimacy of the identity of U_i, after which it generates a random number R_g and a new timestamp T_2, and it computes $\mathrm{SG}_k = h(\mathrm{SID}_k \| G_j \oplus N_l)$, $B_4 = R_u \oplus \mathrm{HPW}_i \oplus \mathrm{SG}_k$, $B_5 = R_g \oplus h(\mathrm{SG}_k \| \mathrm{SID}_k)$, and $B_6 = h(\mathrm{QID}_k \| B_4 \| B_5 \| SG_k \| R_u \oplus \mathrm{HPW}_i \| R_g)$. Finally, GW_j sends $M_2\{\mathrm{QID}_k, B_4, B_5, B_6, T_2\}$ to the sensor node SN_k.

Step 3: once M_2 is received, SN_k verifies that $|T_2 - T_C| \leqq \Delta T$, and if this is true, then the message M_2 is fresh. Afterwards, SN_k obtains the corresponding RSG_k in storage based on QID_k. It computes $\mathrm{SG}_k = \mathrm{RSG}_k \oplus \mathrm{SID}_k$, $(R_u \oplus \mathrm{HPW}_i) = B_4 \oplus \mathrm{SG}_k$, $R_g = B_5 \oplus h(\mathrm{SG}_k \| SID_k)$, and $B_6^* = h(\mathrm{QID}_k \| B_4 \| B_5 \| \mathrm{SG}_k \| R_u \oplus \mathrm{HPW}_i \| R_g)$, and it verifies whether $B_6^* \overset{?}{=} B_6$. If the verification is successful, SN_k creates a random number R_s and a timestamp T_3, after which it computes the keys $\mathrm{SK}_s = h(R_u \oplus \mathrm{HPW}_i \| R_g \| R_s)$, $B_7 = h(\mathrm{SG}_k \| R_g) \oplus R_s$, and $B_8 = h(R_g \| R_s \| \mathrm{SG}_k \| T_3)$. SN_k then sends $M_3\{B_7, B_8, T_3\}$ to GW_j over the public channel.

Step 4: after receiving message M_3, GW_j verifies the freshness of timestamp T_3 using $|T_3 - T_C| \leqq \Delta T$. After verifying that it passes, GW_j generates timestamp T_4 and computes $R_s = h(\mathrm{SG}_k \| R_g) \oplus B_7$ and $B_8^* = h(R_g \| R_s \| \mathrm{SG}_k \| T_3)$, after which it verifies the legitimacy of B_8. If B_8 qualifies, the key $\mathrm{SK}_g = h(R_u \oplus \mathrm{HPW}_i \| R_g \| R_s)$, $B_9 = h(R_u \oplus \mathrm{GI}D_j \| \mathrm{HPW}_i) \oplus (R_g \| R_s)$, and $B_{10} = h(R_0 \oplus G_j \| \mathrm{SK}_g \| R_u)$. Finally, GW_j generates M_4 $\{B_9, B_{10}, T_4\}$ and passes M_4 back to U_i.

Step 5: in the final step, after receiving the message M_4, U_i verifies whether $|T_4 - T_C| \leqq \Delta T$, and if this is correct, it computes $(R_g \| R_s) = B_9 \oplus h(R_u \oplus \mathrm{GID}_j \| \mathrm{HPW}_i)$, $\mathrm{SK}_u = h(R_u \oplus \mathrm{HPW}_i \| R_g \| R_s)$, and $B_{10}^* = h(R_0 \oplus G_j \| \mathrm{SK}_u \| R_u)$. Finally, U_i verifies whether $B_{10}^* \overset{?}{=} B_{10}$, and if this is true, the verification and key exchange phase is complete.

5. Security Analysis

In this section, we use the random oracle model (ROR) to conduct a rigorous formal security analysis of the improved protocol. In addition, an informal security analysis is carried out to logically analyze the protocol. Through the following security analysis, it is easy to prove the security and robustness of the improved protocol.

5.1. Formal Security Analysis. In this section, the ROR model is mainly used to prove the security and feasibility of our proposed protocol, and we successfully demonstrated that users and sensor nodes can securely establish session keys through the gateway. In the proof process, U represents a user, G represents a gateway, and S represents a sensor node. The detailed proof of the procedure is presented as follows.

5.1.1. ROR Model. In this section, we will use the ROR model to prove the security and reliability of our proposed new scheme, where $\mathscr{A}$ represents the attacker. There are three participants which are user U, gateway G, and sensor S. Suppose Π_U^x represents the x-th communication of the user, Π_{U*}^i represents the i-th instance of the user, Π_G^j represents the j-th instance of the gateway, and Π_S^k represents the k-th instance of the sensor. The attacker has special capabilities and can initiate the following queries:

Execute $(\Pi_{U*}^x, \Pi_G^j, \Pi_S^k)$: by executing this query, $\mathscr{A}$ can intercept and obtain the messages transmitted between the various participant instances on the public channel. Passive attacks can be executed by this query

Send (Π_U^x, M): in this query, $\mathscr{A}$ can get the corresponding response by sending message M to Π_U^x. $\mathscr{A}$ can perform man-in-the-middle attacks and impersonation attacks.

U_i		*Mobile device*
U_i enters ID_i^* and PW_i^* imprints BIO_i'	$\xrightarrow{Input\ to\ mobile\ device}$	
		$Rep(BIO_i^*, \tau_i) = \sigma_i^*$ $HPW_i^* = h(PW_i^* \parallel \sigma_i^*)$ $A_3^* = h(ID_i^* \parallel HPW_i)$ Verify $A_3 \overset{?}{=} A_3^*$ If true, user authentication passed

Figure 2: Login phase.

U_i	GW_j	SN_k
Selects SID_k, R_u, T_1 Computes $(R_0 \oplus G_j) = A_2 \oplus h(GID_j \parallel HPW_i)$ $B_1 = SID_k \oplus h(GID_j \parallel HPW_i)$ $B_2 = R_u \oplus h(GID_j \parallel HPW_i \oplus SID_k)$ $B_3 = (R_0 \oplus G_j) \oplus h(GID_j \parallel R_u)$ $\xrightarrow{M_1=\{CID_i,GID_j,B_1,B_2,B_3,T_1\}}$		
	Verify $\lvert T_1 - T_C \rvert \leqq \Delta T$ Gets HPW_i, QID_k Computes $SID_k = B_1 \oplus h(GID_j \parallel HPW_i)$ $R_u = B_2 \oplus h(GID_j \parallel HPW_i \oplus SID_k)$ $(R_0 \oplus G_j) = B_3 \oplus h(GID_j \parallel R_u)$ $A_1^* = h(CID_i \parallel GID_j \parallel R_0 \oplus G_j) \oplus HPW_i$ Check $A_1 \overset{?}{=} A_1^*$ Selects R_g, T_2 $SG_k = h(SID_k \parallel G_j \oplus N_l)$ $B_4 = R_u \oplus HPW_i \oplus SG_k$ $B_5 = R_g \oplus h(SG_k \parallel SID_k)$ $B_6 = h(QID_k \parallel B_4 \parallel B_5 \parallel SG_k \parallel R_u \oplus HPW_i \parallel R_g)$ $\xrightarrow{M_2=\{QID_k,B_4,B_5,B_6,T_2\}}$	
		Verify $\lvert T_2 - T_C \rvert \leqq \Delta T$ Gets RSG_k based on QID_k $SG_k = RSG_k \oplus SID_k$ $(R_u \oplus HPW_i) = B_4 \oplus SG_k$ $R_g = B_5 \oplus h(SG_k \parallel SID_k)$ $B_6^* = h(QID_k \parallel B_4 \parallel B_5 \parallel SG_k \parallel R_u \oplus HPW_i \parallel R_g)$ Verify $B_6^* \overset{?}{=} B_6$ Selects R_s,T_3 Computes $SK_s = h(R_u \oplus HPW_i \parallel R_g \parallel R_s)$ $B_7 = h(SG_k \parallel R_g) \oplus R_s$ $B_8 = h(R_g \parallel R_s \parallel SG_k \parallel T_3)$ $\xleftarrow{M_3=\{B_7,B_8,T_3\}}$
	Verify $\lvert T_3 - T_C \rvert \leqq \Delta T$ Computes $R_s = h(SG_k \parallel R_g) \oplus B_7$ $B_8^* = h(R_g \parallel R_s \parallel SG_k \parallel T_3)$ Check $B_8^* \overset{?}{=} B_8$ Selects T_4 $SK_g = h(R_u \oplus HPW_i \parallel R_g \parallel R_s)$ $B_9 = h(R_u \oplus GID_j \parallel HPW_i) \oplus (R_g \parallel R_s)$ $B_{10} = h(R_0 \oplus G_j \parallel SK_g \parallel R_u)$ $\xleftarrow{M_4=\{B_9,B_{10},T_4\}}$	
$\lvert T_4 - T_C \rvert \leqq DeltaT$ Compute $(R_g \parallel R_s) = B_9 \oplus h(R_u \oplus GID_j \parallel HPW_i)$ $SK_u = h(R_u \oplus HPW_i \parallel R_g \parallel R_s)$ $B_{10}^* = h(R_0 \oplus G_j \parallel SK_u \parallel R_u)$ Checks $B_{10}^* \overset{?}{=} B_{10}$ If ture, communication is possible		

Figure 3: Mutual authentication and key agreement phase.

Hash $(\Pi_U^x$, string): in this query, the hash value of the input string can be obtained by $\mathscr{A}$.

Corrupt (Π_U^x): through this query, $\mathscr{A}$ can send this query to the instance Π_U^x and Π_U^x returns the secret value of U: long-term private key, password, and secret parameters stored in the smart card (based on the smart card). $\mathscr{A}$ can simulate the execution of forward secrecy, privilege insider (internal) attacks, and stolen smart card attacks.

Reveal (Π_U^x): $\mathscr{A}$ can send this query to the instance Π_U^x and Π_U^x returns the current session key SK generated by its partner to $\mathscr{A}$. $\mathscr{A}$ can simulate the execution of known session key attacks.

Test (Π_U^x): $\mathscr{A}$ can perform this query by flipping a coin C. If C results in 1, the attacker will get the correct session key; otherwise, the attacker will receive a random string.

Theorem 1. *In the above* ROR *model, we redefine the $\mathscr{A}$'s capabilities and allow the attacker to execute the above query, so the probability P of our proposed new protocol being broken is expressed as* $\mathrm{Adv}_{\mathscr{A}}^{\mathrm{v}}(\xi) \leq q_{\mathrm{send}}/2^{l-2} + 3q_{\mathrm{hash}}^2/2^{l-1} + 2\max\{C', q_{\mathrm{send}}^{s'}, q_{\mathrm{send}}/2^l\}$, *where* q_{hash} *represents the number of* hash *queries performed and* q_{send} *represents the* *number of queries performed. The number of bits of biological information is expressed by l, C′ and s′ are* Zipf′s law *[26].*

Proof. We define GM_0 to GM_5 to mimic and verify the behavior that may be performed by $\mathscr{A}$. $\mathrm{Succ}_{\mathscr{A}}^{GM_i}(\xi)$ is used to denote the probability of success of $\mathscr{A}$'s attack on the protocol in GM_i. The specific process is as follows:

GM_0: in GM_0, $\mathscr{A}$ does not initiate any queries. Therefore, in GM_0, the probability P that the protocol is broken in this query round is

$$\mathrm{Adv}_{\mathscr{A}}^{\mathrm{v}}(\xi) = \left|2\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_0}(\xi)\right] - 1\right|. \quad (1)$$

GM_1: GM_1 adds Execute query, and the others have no difference with GM_0. We can obtain

$$\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_1}(\xi)\right] = \Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_0}(\xi)\right]. \quad (2)$$

GM_2: GM_2 adds Send query, and there is no difference with GM_1. Therefore, we can get

$$\left|\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_2}(\xi)\right] - \Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_1}(\xi)\right]\right| \le \frac{q_{\mathrm{send}}}{2^l}. \quad (3)$$

GM_3: GM_3 and GM_2 are indistinguishable except that it adds the Hash query and deletes the Send query. We can obtain

$$\left|\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_3}(\xi)\right] - \Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_2}(\xi)\right]\right| \le \frac{q_{\mathrm{hash}}^2}{2^{l+1}}. \quad (4)$$

GM_4: in GM_4, whether a session key is secure or not can be seen in the following two cases. The first case is whether the protocol can ensure perfect forward secrecy security when $\mathscr{A}$ obtains the long-term private key. The second is whether the protocol can resist the temporary information leakage attack when the temporary information is compromised.

(1) Perfect forward secrecy: using Π_G^j, $\mathscr{A}$ tries to obtain the long-term key SG_k between the gateway and the sensor, or $\mathscr{A}$ uses Π_{U*}^x or Π_S^k to try to get a certain secret value in the registration phase
(2) Known session-specific temporary information attacks: $\mathscr{A}$ uses one of Π_G^j or Π_{U*}^i or Π_S^k to try to obtain temporary information from one entity

In both cases, $\mathscr{A}$ only needs to use Send and Hash queries to compute $SK_u = h(R_u \oplus HPW_i \| R_g \| R_s)$. For the first case, assuming that $\mathscr{A}$ obtains the long-term key SG_k, although $R_u \oplus HPW_i$ can be computed by intercepting B_4, $\mathscr{A}$ has no access to SID_k and thus cannot compute R_g and R_s and thus even less likely to compute SK. For the second case, assuming that $\mathscr{A}$ obtains the temporary information R_u, $\mathscr{A}$ has no access to the other random numbers R_g and R_s and thus cannot crack this protocol. Therefore, we get

$$\left|\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_4}(\xi)\right] - \Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_3}(\xi)\right]\right| \le \frac{q_{\mathrm{send}}}{2^l} + \frac{q_{\mathrm{hash}}^2}{2^{l+1}}. \quad (5)$$

GM_5: in GM_5, $\mathscr{A}$ can execute smart card stolen attacks. $\mathscr{A}$ uses $\mathrm{Corrupt}(\Pi_U^x)$ to get the information stored in $SC\{A_2, A_3, GID_j, \mathrm{Gen}(.), \mathrm{Rep}(.), \tau_i\}$. The mobile user uses password PW_i and biological information BIO_i to register. If $\mathscr{A}$ tries to guess $A_3^* = h(ID_i^* \| HPW_i)$, since HPWi is encrypted with biological information, the probability of $\mathscr{A}$ guessing the biometric σ_i is $1/2^l$ [27]. $\mathscr{A}$ can also guess low-entropy passwords; using Zipf′*s* law [26], we can get

$$\left|\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_5}(\xi)\right] - \Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_4}(\xi)\right]\right| \le \max\left\{C', q_{\mathrm{send}}^{s'}, \frac{q_{\mathrm{send}}}{2^l}\right\}. \quad (6)$$

GM_6: GM_6 is used to verify whether the proposed protocol is resistant to impersonation attacks. In GM_6, if $\mathscr{A}$ issues a $h(R_u \oplus HPW_i \| R_g \| R_s)$ query, the game is terminated. So we can obtain

$$\left|\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_6}(\xi)\right] - \Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_5}(\xi)\right]\right| \le \frac{q_{\mathrm{hash}}^2}{2^{l+1}}. \quad (7)$$

Since GM_6 has half the probability of success and failure,

$$\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_6}(\xi)\right] = \frac{1}{2}. \quad (8)$$

To sum up, we can obtain the following conclusions:

$$\begin{aligned}
\frac{1}{2}\mathrm{Adv}_{\mathscr{A}}^{\mathrm{V}}(\xi) &= \left|\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_0}(\xi)\right] - \frac{1}{2}\right| \\
&= \left|\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_0}(\xi)\right] - \Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_6}(\xi)\right]\right| \\
&= \left|\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_1}(\xi)\right] - \Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_6}(\xi)\right]\right| \\
&\le \sum_{i=0}^{5}\left|\Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_{i+1}}(\xi)\right] - \Pr\left[\mathrm{Succ}_{\mathscr{A}}^{GM_i}(\xi)\right]\right| \\
&= \frac{q_{\mathrm{send}}}{2^{l-1}} + \frac{3q_{\mathrm{hash}}^2}{2^{l-1}} + \max\left\{C', q_{\mathrm{send}}^{s'}, \frac{q_{\mathrm{send}}}{2^l}\right\}.
\end{aligned} \quad (9)$$

Finally, we can get

$$\mathrm{Adv}_{\mathscr{A}}^{\mathrm{v}}(\xi) \le = \frac{q_{\mathrm{send}}}{2^{l-1}} + \frac{3q_{\mathrm{hash}}^2}{2^{l-1}} + 2\max\left\{C', q_{\mathrm{send}}^{s'}, \frac{q_{\mathrm{send}}}{2^l}\right\}. \quad (10)$$

Therefore, we can use the ROR model to demonstrate that our proposed new protocol can provide perfect forward security against common attacks such as smart card theft attacks, man-in-the-middle attacks, and other more common attacks. □

5.2. Informal Security Analysis. In this section, we prove that our proposed protocol is secure against common attacks. The security of our proposed protocol and the reasons it can withstand attacks are analyzed.

5.2.1. Resisting Sensor Node Capture Attacks. If an attacker captures a sensor node and obtains its memory information, although the attacker already knows the parameters RSG_k and QID_k, to obtain SK, the attacker must also know SID_k and the long-term key SG_k between the gateway and the sensor node, which is obtained from RSG_k and SID_k through heterodyning. However, SID_k is not stored in the memory of the sensor node. Therefore, our proposed protocol is improved to effectively prevent sensor node capture attacks.

5.2.2. Ensuring Authentication between Users and Mobile Devices. An attacker can replay eavesdropped messages and obtain valuable information through replay and feedback. For example, an attacker can replay message M_1 by imitating the user. However, our improved protocol does not provide this opportunity to the attacker. This is because we add a timestamp T to verify the freshness of the message, and we set a reasonable timestamp threshold. Moreover, we add biometric authentication to ensure accurate authentication between users and mobile devices, thereby preventing attackers from attacking the gateway using large amounts of useless information resulting from the lack of authentication between users and devices.

5.2.3. Perfect Forward Secrecy. If an attacker cannot obtain the previous session key when the private long-term key is destroyed, the authentication protocol has perfect forward confidentiality [28, 29]. Assuming that an attacker has obtained the long-term key SG_k between the gateway and the sensor, although it can be obtained through the message B_4 of the common channel ($R_u \oplus \mathrm{HPW}_i$), R_g and R_s are protected by the long-term key SG_k in addition to SID_k. Therefore, an attacker cannot obtain SID_k while obtaining the long-term key. As such, it can be inferred that the attacker cannot crack the long-term key in the case of obtaining the past session key. Thus, our proposed protocol demonstrates perfect forward security.

5.2.4. Resisting Session-Specific Temporary Information Attacks. If short-term secret information, such as random numbers, is cracked and obtained by an attacker, the attacker cannot calculate the key SK. Because the improved protocol uses a three-way random number and the encrypted value of the user's password information composition, an attacker cannot obtain the user's password information through the knowledge of the random number. Therefore, our proposed protocol can resist temporary information leakage attacks.

5.2.5. Resisting Offline Password-Guessing Attacks. In the authentication stage, we use the pseudo-password HPW_i as a substitute for the user password to ensure the security and privacy of the password. Because the user password is obtained through the user's biological information and password encryption, assuming that the attacker obtains HPW_i, the user password cannot be calculated. In the login phase, assuming that the attacker obtains A_3 and ID_i, the attacker cannot calculate PW_i from these data. Therefore, our proposed protocol can resist offline password-guessing attacks.

5.2.6. Resisting Privileged Insider Attacks. Assuming that an attacker is an insider of the gateway and has access to the gateway's memory information [30], the attacker can obtain CID_i, HPW_i, and QID_i. After obtaining this internal information, the attacker cannot compute any valuable information, and thus, the exact protocol is completely resistant to privileged insider attacks.

5.2.7. Resisting Relay Attacks. In the general three-party authentication protocol, the general steps involve authenticating communications between the user and the server. The server then communicates with the sensor or other devices for authentication, after which the sensor and other devices pass the information to the user through the server, and the information finally reaches the user, server, sensors, and other devices involved in the three-party authentication process. However, the transmission process is prone to relay attacks [30, 31], where information can easily be intercepted by the attacker using disguised devices to obtain the correct information sent by the official server or the user, so that they can disguise themselves as legitimate servers and send instructions to the user or disguise themselves as legitimate users to obtain valuable information. However, in our proposed protocol, the server GW_j properly verifies the legitimacy of user U_i and sensor SN_k by comparing A_1 and B_8. Additionally, the sensors and users verify the legitimacy of the server, and they employ a timestamp to verify the freshness of the message. Thus, our proposed protocol is resistant to relay attacks.

5.2.8. Resisting Stolen-Verifier Attacks. In a stolen authentication attack, we assume that the user authentication value stored on the server side is stolen by an attacker, and the attacker can directly use the authentication value to disguise themselves as a user and log into the system. Further, we assume that the secret information stored on the server side is also stolen, and the attacker can use this information to obtain the public key. Assuming that an attacker obtains the stored information inside the gateway GW_j, which is $\{\mathrm{CID}_i, \mathrm{HPW}_i, A_1, \mathrm{QID}_k, N_l\}$, the key to determining SK involves obtaining SG_k and obtaining Ru using SG_k. However, SG_k cannot be obtained using the information in the memory of GW_j. Therefore, our proposed protocol can resist stolen authentication attacks.

Table 2: Comparisons of security.

Security properties	Fotouhi et al. [22]	Kumari et al. [32]	Srinivas et al. [33]	Gope and Hwang [34]	Ours
Perfect forward secrecy	✓	✓	✓	✓	✓
Resists impersonation attacks	✓	✓	✓	✓	✓
Resists offline password-guessing attacks	✓	×	×	×	✓
User anonymity security	✓	✓	✓	✓	✓
Mutual authentication	✓	✓	✓	✓	✓
Resists replay attacks	×	✓	✓	✓	✓
Resists sensor-capture attacks	×	×	✓	✓	✓
Resists known session temporary information attacks	✓	×	✓	✓	✓
Resists relay attacks	✓	✓	✓	✓	✓
Resists man-in-the-middle attacks	✓	×	✓	✓	✓
Provable security	×	×	×	×	✓

Table 3: The computational cost of complex operations.

Operations	Host node(s)
Hash function	0.00032
Fuzzy function	0.0171
Chaotic map function	0.0171
Encryption and decryption	0.0056

Table 4: Calculation cost comparison.

Protocol	User	Gateway	Sensor	Total (ms)
Fotouhi et al.'s [22]	$10T_h$	$17T_h$	$7T_h$	$37T_h = 10.88$
Kumari et al.'s [32]	$8T_h + 2T_s$	$4T_h + T_s$	$4T_h + 2T_s$	$16T_h + 5T_s = 33.12$
Srinivas et al.'s [33]	$4T_h + 2T_c + 2T_s$	$6T_h + 2T_s$	$3T_h + 2T_c$	$4T_c + 4T_s + 13T_h = 94.96$
Gope et al.'s [34]	$7T_h$	$9T_h$	$3T_h$	$19T_h = 6.08$
Ours	$9T_h + 1T_{fe}$	$10T_h$	$4T_h$	$23T_h + 1T_{fe} = 24.46$

6. Security and Performance Comparisons

In this section, we discuss the typical costs of the authentication protocols from three aspects: protocol security, computing cost, and storage consumption [22, 32–34].

6.1. Security Comparisons. As shown in Table 2, we compared the security analysis of the mentioned protocols and used ✓ and × to signify whether the protocol meets the security requirements involved. The security of the protocol proposed by Kumari et al. [32] was disproved by Li et al. [35] in that it cannot resist sensor node capture attacks, session-specific temporary information attacks, sensor node impersonation attacks, and man-in-the-middle attacks. Therefore, Li et al. designed a mutual authentication and key agreement protocol for wireless sensor networks. However, it was later proved to be unsafe. The protocol proposed by Srinivas et al. [33] cannot resist offline password-guessing attacks. The security of the protocol proposed by Gope and Hwang [34] was disproved by Adavoudi-Jolfaei et al. [36] in that the adversary can obtain the session key between the user and the sensor using the dy model. Compared to the protocols mentioned above, our proposed protocol can resist such attacks and meet the security requirements.

6.2. Performance Comparisons. We performed a performance comparison between the new authentication protocol and the other four authentication protocols listed in Table 4. Additionally, we made the following calculations in terms of the time consumption of cryptographic operations, as shown in Table 3, including hash functions, symmetric key encryption/decryption, chaotic mapping functions, and fuzzy extraction functions, as the most important operations [22]. The meanings of symbols in Table 4 are as follows: T_h denotes the time of the regular hash operation, T_{fe} denotes the operation time of the fuzzy function, T_s denotes the operation time of symmetric encryption and decryption, and T_c denotes the operation time of the chaotic map function.

In the login and mutual authentication phase, we compared the computation times of the user, gateway, and sensor node sides along with other protocols to design our proposed protocol. As shown in Table 4, the newly designed protocols guarantee security and time appropriateness. Although our new protocol takes slightly more time than the protocols proposed in Fotouhi et al.'s [22] and Gope and Hwang's [34], it ensures improved security. This is because the extra time spent is mainly in the user login phase, where the user biometric information needs to be compared, a very important and indispensable step that amounts to a partial performance sacrifice to improve the security of the

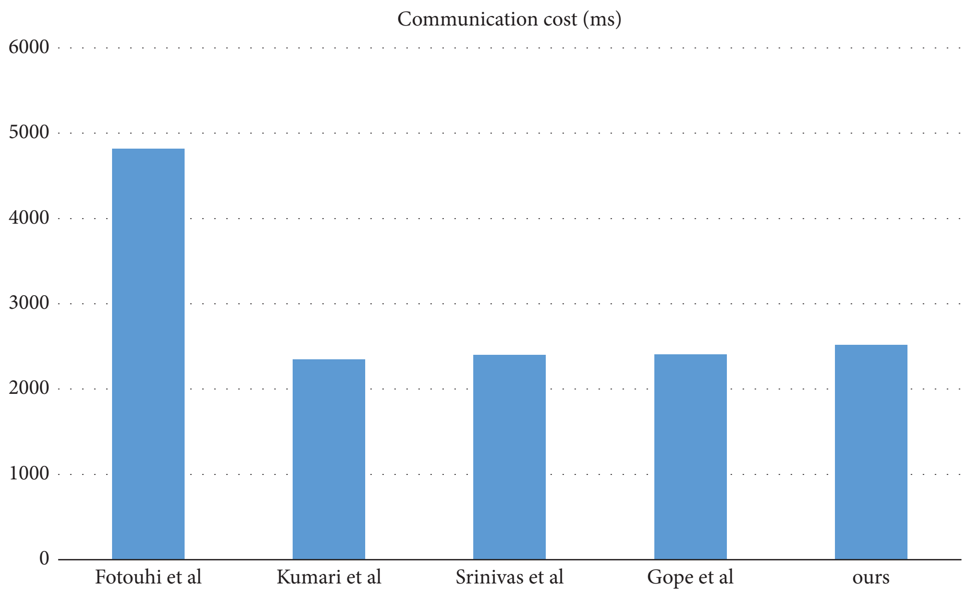


FIGURE 4: Communication cost.

protocol. As a result, the new protocol is more secure than the two protocols and ensures that the user's legitimacy is verified. Compared to Kumari et al.'s [32] and Srinivas et al.'s [33] proposed protocols, it is evident that our proposed protocol significantly reduces the computational cost. In addition, we compared the communication costs, as shown in Figure 4. Considering the computational cost and communication in terms of cost and security for the new protocol, it is evident that our proposed protocol can be better adapted to the wireless human medical environment regional network, thereby providing improved service experience for hospital staff and individual patients.

7. Conclusion

In this study, we improve on the WBAN-based authentication protocol proposed by Fotouhi et al. in medical IoT. The improved protocol compensates for the defects in the original protocol, and it can resist attacks that cannot be resisted by the original protocol. It also improves the authentication speed of the protocol, thereby reducing computational expenditure. Moreover, it is advantageous in that it is lightweight compared to the original protocol. The improved protocol adds biometric authentication and login authentication to significantly increase the security of the user login process, and it also makes extensive use of single hash, heterogeneous, and joint operations to reduce computational cost. Our proposed protocol is highly secure against a range of attacks, such as sensor node capture attacks, replay attacks, and internal privilege attacks. It demonstrates excellent performance in terms of security and efficiency. Therefore, it can be considered more suitable for the WBAN-based medical IoT. For every new technology development there are bound to be technical implementation and realization challenges, and the Internet of Healthcare is facing some problems in terms of adoption for the time being. Most of the problems exist because there is no all-in-one healthcare IoT solution; all solutions are tailored to specific challenges and therefore can be too expensive for any organization. The second is the lack of a set of standards for the healthcare industry to protect extremely sensitive healthcare data from security risks and threats. It is hoped that this paper will provide a reference for addressing the security aspects of healthcare 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

The work of Long Li was supported by Guangxi Key Laboratory of Trusted Software (no. KX202033).

References

[1] M. Shafi, A. F. Molisch, P. J. Smith et al., "5G: a tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201–1221, 2017.

[2] H. Xiong, X. Huang, M. Yang, L. Wang, and S. Yu, "Unbounded and efficient revocable attribute-based encryption with adaptive security for cloud-assisted internet of things," *IEEE Internet of Things Journal*, vol. 2021, Article ID 3094323, 2021.

[3] H. Xiong, Y. Wu, C. Jin, and S. Kumari, "Efficient and privacy-preserving authentication protocol for heterogeneous systems in IIoT," *IEEE Internet of Things Journal*, vol. 7, no. 12, pp. 11713–11724, 2020.

[4] X. Chen, M. Li, H. Zhong, Y. Ma, and C. H. Hsu, "DNNOff: offloading DNN-based intelligent IoT applications in mobile

edge computing," *IEEE Transactions on Industrial Informatics*, vol. 2021, Article ID 3075464, 2021.

[5] J. W. Jiao Wang, J.-S. P. Jiao Wang, S.-C. C. Jeng-Shyang Pan, Z.-Y. M. Shu-Chuan Chu, and H. L. Zhen-Yu Meng, "Improved black hole algorithm for intelligent traffic navigation," *Journal of Internet Technology*, vol. 22, no. 4, pp. 725–734, 2021.

[6] X. Xue, X. Wu, C. Jiang, G. Mao, and H. Zhu, "Integrating sensor ontologies with global and local alignment extractions," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6625184, 2021.

[7] S. Lv and Y. Liu, "PLVA: privacy-preserving and lightweight V2I authentication protocol," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2021, Article ID 3059638, 2021.

[8] P. Wang, C. M. Chen, S. Kumari, M. Shojafar, R. Tafazolli, and Y. N. Liu, "HDMA: hybrid D2D message authentication scheme for 5G-enabled VANETs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2020, Article ID 3013928, 2020.

[9] J. Song, Q. Zhong, W. Wang, C. Su, Z. Tan, and Y. Liu, "FPDP: flexible privacy-preserving data publishing scheme for smart agriculture," *IEEE Sensors Journal*, vol. 2020, Article ID 3017695, 2020.

[10] E. K. Wang, X. Liu, C. M. Chen, S. Kumari, M. Shojafar, and M. S. Hossain, "Voice-transfer attacking on industrial voice control systems in 5G-aided IIoT domain," *IEEE Transactions on Industrial Informatics*, vol. 2020, Article ID 3023677, 2020.

[11] W. Zhang, Y. Wu, H. Xiong, and Z. Qin, "Accountable attribute-based encryption with public auditing and user revocation in the personal health record system," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 15, no. 1, pp. 302–322, 2021.

[12] J. Sun, H. Xiong, X. Liu, Y. Zhang, X. Nie, and R. H. Deng, "Lightweight and privacy-aware fine-grained access control for IoT-oriented smart health," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6566–6575, 2020.

[13] 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.

[14] E. Jovanov, A. Milenkovic, C. Otto et al., "A WBAN system for ambulatory monitoring of physical activity and health status: applications and challenges," *IEEE*, vol. 2005, Article ID 1615290, 3813 pages, 2005.

[15] M. R. Yuce, "Implementation of wireless body area networks for healthcare systems," *Sensors and Actuators A: Physical*, vol. 162, no. 1, pp. 116–129, 2010.

[16] J. Liu, Z. Zhang, X. Chen, and K. S. Kwak, "Certificateless remote anonymous authentication schemes for wirelessbody area networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 332–342, 2013.

[17] Z. Zhao, "An efficient anonymous authentication scheme for wireless body area networks using elliptic curve cryptosystem," *Journal of Medical Systems*, vol. 38, no. 2, pp. 13–17, 2014.

[18] S. Chatterjee, A. K. Das, and J. K. Sing, "A novel and efficient user access control scheme for wireless body area sensor networks," *Journal of King Saud University - Computer and Information Sciences*, vol. 26, no. 2, pp. 181–201, 2014.

[19] C. Wang and Y. Zhang, "New authentication scheme for wireless body area networks using the bilinear pairing," *Journal of Medical Systems*, vol. 39, no. 11, pp. 1–8, 2015.

[20] P. K. Dhillon and S. Kalra, "Multi-factor user authentication scheme for IoT-based healthcare services," *Journal of Reliable Intelligent Environments*, vol. 4, no. 3, pp. 141–160, 2018.

[21] F. Wu, X. Li, A. K. Sangaiah et al., "A lightweight and robust two-factor authentication scheme for personalized healthcare systems using wireless medical sensor networks," *Future Generation Computer Systems*, vol. 82, pp. 727–737, 2018.

[22] M. Fotouhi, M. Bayat, A. K. Das, F. Han, S. M. Pournaghi, and M. A. Doostari, "A lightweight and secure two-factor authentication scheme for wireless body area networks in health-care IoT," *Computer Networks*, vol. 177, Article ID 107333, 2020.

[23] 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.

[24] D. Wang, D. He, P. Wang, and C. H. Chu, "Anonymous two-factor authentication in distributed systems: certain goals are beyond attainment," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 428–442, 2014.

[25] D. Wang and P. Wang, "Two birds with one stone: two-factor authentication with security beyond conventional bound," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 708–722, 2016.

[26] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[27] V. Odelu, A. K. Das, and A. Goswami, "A secure biometrics-based multi-server authentication protocol using smart cards," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1953–1966, 2015.

[28] P. Li, J. Su, and X. Wang, "iTLS: lightweight transport-layer security protocol for iot with minimal latency and perfect forward secrecy," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6828–6841, 2020.

[29] L. Xiong, D. Peng, T. Peng, H. Liang, and Z. Liu, "A lightweight anonymous authentication protocol with perfect forward secrecy for wireless sensor networks," *Sensors*, vol. 17, no. 11, p. 2681, 2017.

[30] 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, 2021.

[31] M. Safkhani, C. Camara, P. Peris-Lopez, and N. Bagheri, "RSEAP2: an enhanced version of RSEAP, an RFID based authentication protocol for vehicular cloud computing," *Vehicular Communications*, vol. 28, Article ID 100311, 2021.

[32] S. Kumari, X. Li, F. Wu, A. K. Das, H. Arshad, and M. K. Khan, "A user friendly mutual authentication and key agreement scheme for wireless sensor networks using chaotic maps," *Future Generation Computer Systems*, vol. 63, pp. 56–75, 2016.

[33] J. Srinivas, D. Mishra, and S. Mukhopadhyay, "A mutual authentication framework for wireless medical sensor networks," *Journal of Medical Systems*, vol. 41, no. 5, p. 80, 2017.

[34] P. Gope and T. Hwang, "A realistic lightweight anonymous authentication protocol for securing real-time application data access in wireless sensor networks," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp. 7124–7132, 2016.

[35] J. Li, W. Zhang, S. Kumari, K. K. R. Choo, and D. Hogrefe, "Security analysis and improvement of a mutual authentication and key agreement solution for wireless sensor networks using chaotic maps," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 6, Article ID e3295, 2018.

[36] 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.