

Protocols, Technologies, and Infrastructures for Secure Mobile Video Communications

Lead Guest Editor: Yuanlong Cao

Guest Editors: Yulei Wu, Jiyang Wu, and Kai Wang





**Protocols, Technologies, and Infrastructures
for Secure Mobile Video Communications**

Security and Communication Networks

**Protocols, Technologies, and
Infrastructures for Secure Mobile Video
Communications**

Lead Guest Editor: Yuanlong Cao

Guest Editors: Yulei Wu, Jiyan Wu, and Kai Wang






Copyright © 2022 Hindawi Limited. All rights reserved.

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

Chief Editor

Roberto Di Pietro, Saudi Arabia

Associate Editors

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

Academic Editors

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

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


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

Contents


Blockchain-Enabled Joint Resource Allocation for Virtualized Video Service Functions

Ping Yu , Hongwei Zhao , Kun Yang, Hanlin Chen, Xiaozhong Geng, Ming Hu, and Hui Yan 
Research Article (16 pages), Article ID 4349097, Volume 2022 (2022)

An Automatic Planning-Based Attack Path Discovery Approach from IT to OT Networks

Zibo Wang, Yaofang Zhang, Zhiyao Liu, Xiaojie Wei, Yilu Chen, and Bailing Wang 
Research Article (18 pages), Article ID 1444182, Volume 2021 (2021)



Detection and Location of Malicious Nodes Based on Homomorphic Fingerprinting in Wireless Sensor Networks

Zhiming Zhang , Yu Yang, Wei Yang, Fuying Wu, Ping Li, and Xiaoyong Xiong
Research Article (12 pages), Article ID 9082570, Volume 2021 (2021)




On Improving the Robustness of MEC with Big Data Analysis for Mobile Video Communication

Jianming Zhao , Peng Zeng , Yingjun Liu , and Tianyu Wang
Research Article (12 pages), Article ID 4539540, Volume 2021 (2021)

V-Lattice: A Lightweight Blockchain Architecture Based on DAG-Lattice Structure for Vehicular Ad Hoc Networks

Xiaodong Zhang , Ru Li , Wenhan Hou, and Hui Zhao
Research Article (17 pages), Article ID 9942632, Volume 2021 (2021)

Stochastic Adaptive Forwarding Strategy Based on Deep Reinforcement Learning for Secure Mobile Video Communications in NDN

Bowei Hao , Guoyong Wang , Mingchuan Zhang , Junlong Zhu , Ling Xing , and Qingtao Wu 
Research Article (13 pages), Article ID 6630717, Volume 2021 (2021)

Research Article

Blockchain-Enabled Joint Resource Allocation for Virtualized Video Service Functions

Ping Yu ^{1,2}, Hongwei Zhao ¹, Kun Yang,³ Hanlin Chen,² Xiaozhong Geng,² Ming Hu,² and Hui Yan ²

¹College of Computer Science and Technology, Jilin University, Changchun 130000, Jilin, China

²Changchun Institute of Technology, Changchun 130012, Jilin, China

³Information and Communication Engineering, University of Electronic Science and Technology, Chengdu 611731, Sichuan, China

Correspondence should be addressed to Hui Yan; yanhui7125@126.com

Received 7 May 2021; Revised 24 March 2022; Accepted 28 March 2022; Published 16 May 2022

Academic Editor: Yuanlong Cao

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

Power information is an important guarantee for energy security. As an important technical means of safety management and risk control, video monitoring is widely used in the power industry. Power video monitoring system uses efficient processing of multimodal video data and automatically identifies abnormal events and equipment status, replacing human monitoring with machine. Video monitoring data of power substations usually contain both visual information and auditory information, and the data types are diversified. The multimodal video data provides a rich underlying data source for the intelligent monitoring function, but it requires multiple service forms for efficient processing. Most intelligent edge monitoring equipment are only equipped with lightweight computing resources and limited battery supply, limited resources, and weak local processing data capabilities. Power video monitoring system has the characteristics of distribution, openness, interconnection, and intellectualization. Its intelligent edge video equipment is widely distributed, which also brings convenience and also brings security risks in terms of data security and reliability. For the outdoor multimodal power video monitoring system scenario, this paper adopts the edge-cloud distributed system architecture to solve the problem of resource shortage and adopts the first proposed service function virtualization (SFV) to solve the problem of multimodal video data processing. At the same time, the problem of security protection is solved by introducing blockchain to establish trust among intelligent video equipment and service providers. Under the security protection of virtualized service consortium blockchain (VSCB), virtualization technology is introduced into the service function chain (SFC) to realize SFV and solve the resource optimal allocation problem of multimodal video data processing. The work mainly involves the joint mapping of virtual resources, physical resources, and the joint optimization of computing and communication resources. Problems such as large state space and high dimensionality of action space have an impact on resource allocation. The stochastic optimization problem of resource allocation is established as a Markov decision process (MDP) model, and SFV technology is used to optimize cost and delay. The resource allocation optimization algorithm (RAOA-A3C) based on asynchronous advantage actor-critic algorithm (A3C) is proposed. Simulation experiments show that the RAOA-A3C proposed in this paper is more suitable for high-dynamic, multidimensional, and distributed power video monitoring system scenario and has achieved better optimization results in reducing time delay and deployment costs.

1. Introduction

Power is the energy basis for economic development, and power information is an effective guarantee for energy security. The power transformer is the most important core equipment in the operation of the power grid. If they fail, they

will have a significant impact. Therefore, real-time video monitoring and fault location of the transformer's operating states play a key role in ensuring the stable operation of the distribution network. For example, in order to ensure the long-term, high-efficiency, and safe operation of unattended or few-person-attended power equipment, the video

monitoring system have played an important role in the work of patrolling field equipment. As an important member of smart grid security protection, the power video monitoring system has the characteristics of distribution, openness, interconnection, and intelligence. First of all, most intelligent edge monitoring equipment equipped with limited battery supply and lightweight computing resources work without manual control. These equipment have obvious shortages of computing and storage resources and communication resources. Secondly, the interactive data of various types of intelligent edge monitoring equipment require multiple services to provide support. Thirdly, the widespread distribution of intelligent edge monitoring equipment makes security usually difficult to guarantee [1–4].

SFC is constituted by service requests for multimodal video data of intelligent edge monitoring equipment. SFV uses virtualization technology to realize the joint allocation of computing resources and communication resources.

The distributed power video monitoring system needs the support of edge computing technology. Edge computing is close to the edge of the network at the source of things or data and provides edge intelligent services nearby. It is an open platform that integrates core capabilities of network, computing, storage, and applications. It can meet the key demands of industry digitalization in agile connection, real-time business, data optimization, intelligence application, and so on [5–7]. Although the edge computing framework is suitable for application in the distributed power transformer video monitoring system scenario, it also faces challenges in terms of security.

The distributed power video monitoring system is a heterogeneous network, and its security needs the strong support of blockchain technology. Blockchain has the characteristics of smart contracts, distributed decision-making, collaborative autonomy, high security against tampering, openness, and transparency. Blockchain is similar to the power video monitoring system in terms of operation mode, topology, and especially security protection [8, 9]. Blockchain technology is an effective solution to establish trust among heterogeneous networks and realize reliable autonomous transaction management [10].

In view of the high dynamic and multidimensional characteristics of the power video monitoring system, Deep reinforcement learning (DRL) has gradually become a highly concerned optimization method [11–13]. DRL combines the perceptual ability of deep learning (DL) with the decision-making ability of reinforcement learning (RL). DRL is an artificial intelligence method that is closer to the way of human thinking and provides solutions for the perception and decision-making problems of complex systems [14]. DRL is suitable for solving complex optimization problems.

Edge computing systems can allocate resources to the edge of the network and provide low-delay network services for terminal equipment. However, there are still important issues such as resource management and safety protection in practical applications in the power video monitoring system scene [15].

It is very important to allocate resources reasonably and efficiently. Resources mainly include CPU resources, storage

resources, and communication resources. These resources are allocated by the controller to better solve problems such as cost and delay. In order to better improve the quality of service (QoS), edge computing and SFV are combined to decouple service functions from hardware equipment. SFV can realize flexible regulation and on-demand allocation of service resources [16, 17]. The use of mobile edge computing (MEC) technology can enable edge nodes to better achieve transaction autonomy [10]. Factors such as equipment heterogeneity, power supply status, and resource location of the power video monitoring system make the resource allocation more complicated. How to design the optimal resource allocation policy of SFC is a very challenging scientific issue [18]. DRL is widely regarded as an effective method to solve decision-making problems in complex environments [19, 20]. The SFC orchestration method based on DRL is used to solve the NP-hard problem of high-dimensional and intensive calculation [21–23]. DRL continuously interacts with the environment, automatically learns the optimal actions to be taken in different states, and optimizes resource allocation according to the optimal strategy.

The heterogeneity of edge nodes makes edge computing more complex and uncertain [24, 25]. The central control node may also suffer a single point of failure, which may cause data to leakage or malicious tampering and ultimately lead to task execution failure or economic loss [26]. Both the equipment themselves and the communications among equipment are facing threats of various security attacks. For example, the equipment may malfunction or be malicious so that the transmitted information may be leaked or tampered with. Therefore, it is very important to ensure data security. Blockchain is a kind of cryptology-supported, verifiable, and immutable ledger. Blockchain ensures interaction through transaction records and distributed consensus on the validity of transaction records. Blockchain with the characteristics of pan-central, distributed, and trustworthy provides new ideas for designing the framework and paradigm of cloud-edge computing [27].

In summary, the integration of SFV, blockchain, edge computing, and DRL technology to solve the resource allocation optimization problem of the power video monitoring system is very worthy of discussion.

2. Related Work

At present, there have been some papers that combine edge computing with blockchain. The introduction of blockchain can solve the security problem of the cloud-edge computing environment. Reference [28] proposes to use blockchain for decentralized task allocation and scheduling in MEC. The purpose is to eliminate the increase in the computational burden of the central server due to the attacker's distributed denial of service attack, so it affects the accuracy of data transmission. Reference [29] proposes an internet of vehicles (IoV) file-sharing scheme based on blockchain smart contracts and attribute encryption. Under the premise of ensuring the efficiency of filesharing, the file-sharing solution

adopts blockchain smart contract technology to avoid third-party participation and protect data security. Reference [30] proposes a blockchain-based energy transaction framework for energy transactions among electric vehicles and smart grids. The autonomous and controllable consensus mechanism puzzle generated by the edge server helps increase transaction speed. References [31, 32] propose that edge servers and terminal equipment participate in the blockchain, and the consortium blockchain is used to manage the virtual resources. Users registered in the consortium blockchain can define and deploy their own virtual systems and read and write blocks. The allocation mechanism of the virtual network function (VNF) improves the efficiency of resource allocation while ensuring security. Reference [33] uses a static VNF allocation policy to reduce the cost of operators while ensuring users' QoS. However, the network environment is dynamically changing, and it is more reasonable to consider long-term optimization. Reference [34] achieves the purpose of reducing end-to-end delay by reducing transmission delay and processing delay, but it does not pay attention to the utilization of physical network resources. Reference [35] realizes the reduction of service provider's capital expenditure and operating expenditure, but it sacrifices reliability and does not consider the end-to-end delay. Reference [36] proposes an algorithm based on deep Q-learning (DQL) to solve the decision-making problem of computing resource allocation at the edge of a multiuser shared network. Reference [37] applies the DRL algorithm to jointly optimize the computational efficiency of the MEC system and the transaction throughput of the blockchain system for the industrial internet system based on the blockchain.

Although the above papers have optimized the security and system performance of the cloud-edge computing environment to varying degrees, there are few related studies in the power video monitoring system scenario, and there are still some potential problems and challenges. First of all, although the introduction of blockchain technology can solve the security problem, the consensus process in the blockchain is inefficient, and there is a serious computational overhead in the system. Secondly, resource allocation still has the following problems. For example, most studies in many papers are based on the prerequisite of the known state of the environment and do not take into account the dynamic changes in the environment over time. Nor does it take into account the fact that the arrival of a large number of service requests will easily cause a backlog of service requests, which will affect the stability of the network. It also failed to take into account the user's QoS while optimizing the cost of resource allocation. Thirdly, there are also problems in solving optimization. The continuous increase in the number of agents will explode the dimensions of the state space, and it becomes infeasible to use the traditional tabular method to solve the problem. DRL can solve the problem of state space explosion caused by the increase in the number of nodes [38, 39]. DRL has been proven to effectively approximate the Q value of RL by using a deep neural network (DNN) [20]. The goal of this paper is to

achieve low-delay, low-cost resource optimization through the use of blockchain, DRL, and SFV technology in the cloud-edge computing environment.

A power video monitoring system is a distributed heterogeneous cloud-edge network. The key to solving the problem is how to select server and physical links that meet service requirements from limited physical resources for allocation [40, 41]. The goal is to maximize resource utilization while ensuring network performance. This paper combines edge computing and SFV to build a cloud-edge computing basic model in order to achieve transaction autonomy at the edge and achieve better QoS. Power video monitoring system is a distributed heterogeneous network involving different public and private networks. The unreliability is obvious. This paper introduces blockchain technology to achieve reliable transaction autonomy. The resource optimization allocation problem is an intensive calculation problem. It is a high-dimensional NP-hard problem. This paper introduces DRL technology to solve the NP-hard problem. In summary, the resource allocation optimization problem is modeled as an MDP, and the resource allocation policy is optimized through SFV to maximize long-term utility performance. This paper proposed the RAOA-A3C algorithm based on A3C in order to obtain the optimal resource allocation policy and finally achieved the goal of improving safety protection and efficient resource management.

The main contributions of this paper are as follows:

- (1) The SFV concept was first proposed based on the characteristics of the power video monitoring system. Multimodal video data service requests constitute service function chains, which use SFV technology to optimize the allocation of computing resources and communication resources.
- (2) SFV, blockchain, edge computing, and DRL technology are used to solve the resource allocation optimization problem of the power video monitoring system. The optimization problem mainly involves the joint mapping of virtual resources and physical resources and the joint optimized allocation of computing resources and communication resources.
- (3) The system architecture is built. The proposed VSCB solves the problem of safety protection. The random optimization problem of resource allocation is modeled as an MDP model, and the RAOA-A3C algorithm is proposed. Simulation experiments show that the delay and cost of the RAOA-A3C algorithm are superior to other methods.

The structure of this paper is arranged as follows. Section 1 introduces the background. Section 2 introduces related work. Section 3 gives the system architecture and workflow. Section 4 proposes the system model. Section 5 proposes the optimization algorithm. Section 6 introduces performance evaluation and analysis. Section 7 summarizes the work.

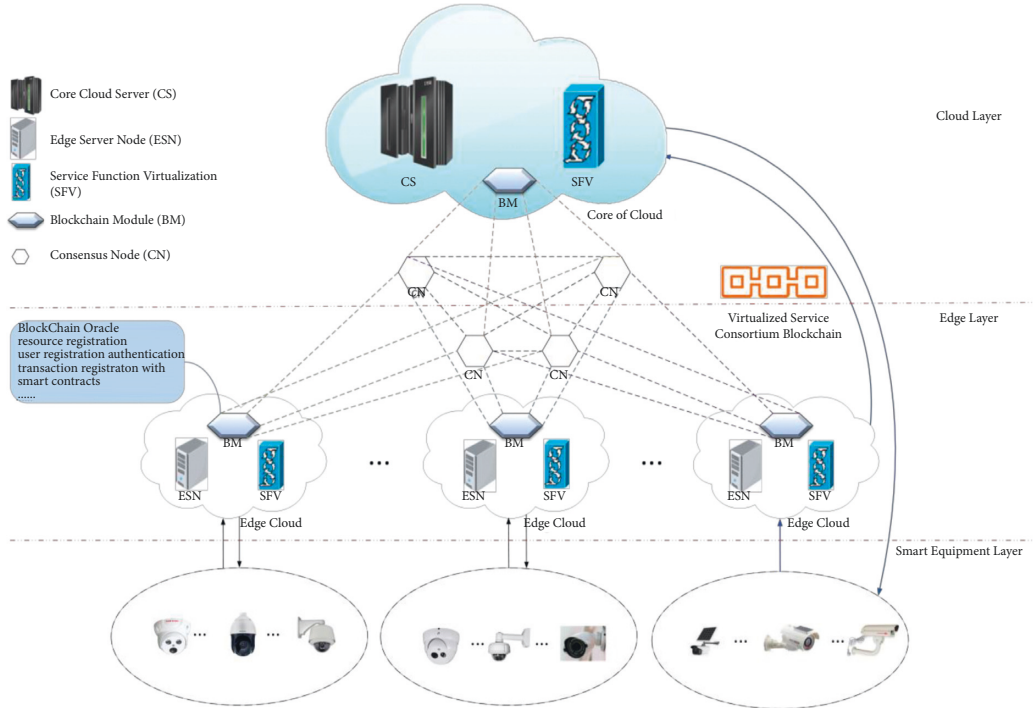


FIGURE 1: Blockchain-enabled system architecture.

3. Blockchain-Enabled System Architecture and Work Flow

3.1. *Blockchain-Enabled System Architecture.* The system architecture of the power video monitoring system is shown in Figure 1.

This paper combines the requirements of the power video monitoring system in resource management and safety protection to build a system architecture. This architecture mainly includes the following three layers:

- (1) **Intelligent Equipment Layer:** The intelligent equipment layer containing multiple types of equipment is at the bottom. Intelligent equipment mainly complete the work of data acquisition and data intelligent processing. Because the resources of the intelligent equipment layer are limited, the intelligent equipment layer that filters out data that have been processed locally send service requests to edge clouds or the core cloud layer.
- (2) **Edge Layer:** The edge layer is composed of heterogeneous edge clouds. The distribution and heterogeneity of the edge layer make the traditional edge layer unable to guarantee the reliability of the service. The edge layer applied with VSCB has the ability to ensure service consistency and provide reliable service management. Each edge cloud in the system model contains three components: (1) service node, (2) blockchain module, and (3) controller.
- (3) **Cloud Layer:** The core cloud layer and the edge layer reach consensus in the same blockchain. The power

video monitoring system belongs to a distributed heterogeneous cloud-side computing environment involving public and private networks. Its system architecture uses VSCB to build a trusted cloud-side computing environment. When the resources of the edge layer cannot meet the service quality and resource constraints of the terminal equipment, it can continue to send service request information to the core cloud in order to obtain relevant resources of the core cloud platform to complete the current service request. The cloud layer also mainly includes three components: (1) service node, (2) blockchain module, and (3) controller.

Next, the three main components included in both the cloud layer and the edge layer are introduced.

- (1) **Service Nodes:** Service nodes of the edge cloud and the core cloud are mainly composed of servers. Each server node is the actual host of the virtual service functions (VSF), which specifically provides various resource services.
- (2) **Blockchain Module:** The blockchain module is composed of high-performance equipment or other lightweight equipment. It is responsible for resource registration, user registration, authentication, smart contract, and transaction registration to ensure trusted and reliable resource allocation [10].
- (3) **Controller:** The controller mainly includes SFV. The essence of SFV is to turn dedicated hardware equipment into general software equipment to achieve the purpose of sharing hardware

infrastructures. The software equipment called VSFs realize functions such as the rapid establishment of the network among VSFs and the rapid allocation of resources. The quality of resource allocation by the controller affects the efficiency of service provision and physical resource usage [42]. The resource allocation optimization algorithm is deployed in the controller. The controller manages, allocates, and monitors the underlying resources. The controller obtains the system information reported from the bottom layer; analyzes the network topology, equipment operation energy consumption, resource utilization, and so on; and then performs tasks such as resource mapping, traffic scheduling, and policy management [21]. The controller helps improve the efficiency of resource management.

3.2. Consortium Blockchain. Blockchain is a kind of chained data structure that combines data blocks sequentially in a time sequence. It is a distributed ledger that cannot be tampered with and cannot be forged, and it is guaranteed by cryptography [43, 44]. The data of the blockchain is collectively maintained. Data operations are witnessed and stored by all nodes, so the data cannot be changed, and it is safe and reliable [45, 46]. Blockchain is a technology that realizes information security and information transparency based on a consensus mechanism. The consortium blockchain is a relatively new way of applying blockchain technology to businesses. It is suitable for providing services for joint collaboration among multiple enterprises, and it has the characteristics of partial decentralization. The consensus participants of the consortium blockchain are a group of preapproved nodes on the network, and the consortium blockchain can exercise a greater degree of control over the network.

Blockchain has been widely used in many fields. Resource allocation in the cloud-edge computing environment is one of the typical cases. The VSCB system proposed in this paper is based on a limited number of enterprises to form the consortium blockchain, and the number of nodes is also limited. Even if there is an expansion of nodes, it will not increase infinitely. The workflow trusted authentication mechanism of VSCB proposed in this paper is described as follows: For any node in the system, its operation is limited by the role control and permission control information on the consortium blockchain to limit its operation scope. The node can read the role control and permission control information to ensure that its work is legal. When the node completes the work and writes the flow information, the role control and permission control are authenticated on the entire consortium blockchain to ensure the normal operation of the entire workflow. At the same time, when the node wants to operate, it must reach a consensus on its authority on the consortium blockchain before writing its operation into the consortium blockchain. When the workflow

continues to flow to the next link, if there is a problem with the authority, then data writing and workflow flow cannot proceed normally.

This paper adopts the practical Byzantine fault tolerance (PBFT) algorithm. The advantages are: first, the system can be separated from the existence of encrypted tokens, the nodes of the algorithm consensus are composed of business participants or supervisors, and the security and stability are guaranteed by business stakeholders. Secondly, the time delay of consensus is short, which basically meets the requirements of commercial real-time processing. Thirdly, the consensus efficiency is high, which can meet the needs of the high-frequency trading volume. Moreover, because of the independence of the smart contract, its execution process and the generated transaction information will not be “maliciously polluted” by the outside world on the consortium blockchain, making the credibility of the transaction information far more than that of the public blockchain. Therefore, the consortium blockchain adopts a more competitive PBFT algorithm, which can improve the application level of the consortium blockchain at the enterprise level to a new level.

This paper uses the token-free optimized PBFT algorithm [10]. The master node is not determined by complex computing puzzles, the master node is determined by circular selection. Therefore, this optimization algorithm can better meet the needs of the power video monitoring system in terms of saving resources.

3.3. Work Flow. The workflow is roughly described as follows: In the cloud-edge computing environment, resources are registered as digital assets on the VSCB, and resource management is realized through the controller. The two main events of this system architecture are resource registration and resource allocation.

3.3.1. Resource Registration. The core cloud or edge cloud needs to register resource information on the blockchain module before providing services. They send information such as equipment identification and related attributes to the blockchain module. The blockchain module is maintained a list of information to form a resource pool and then uses this information to form a block. In this way, the core cloud or edge cloud can provide hosts for VSFs under the supervision of the VSCB.

3.3.2. Resource Allocation. When the smart equipment layer sends out a service request, the current request is first allocated to the adjacent edge cloud in the edge layer. The blockchain module in the edge cloud first verifies the user’s identity. After the identity of the user who sent the service request is authenticated, the request is passed to the controller to obtain the optimal resource allocation. If the resource constraints of the adjacent edge cloud and the QoS of the user cannot be guaranteed, the service request is sent to

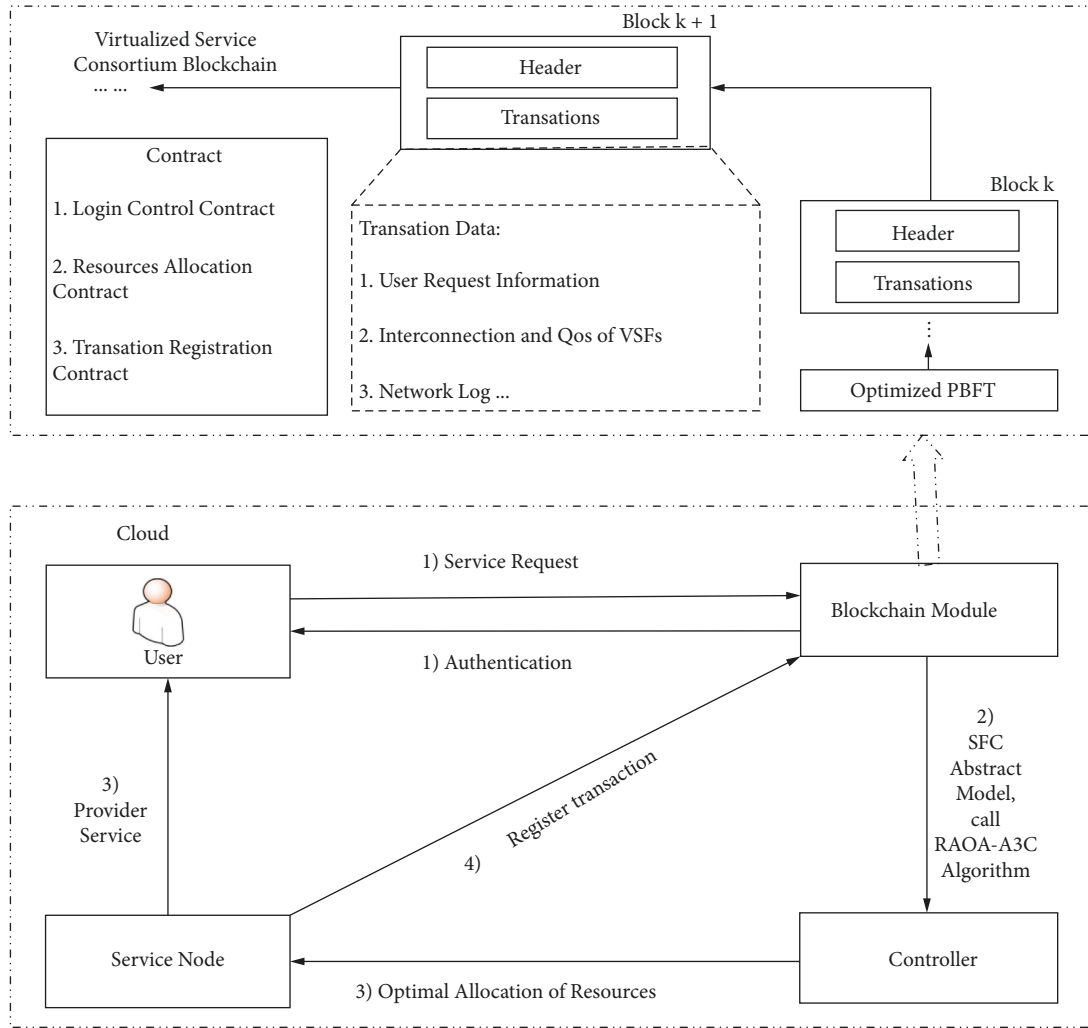


FIGURE 2: Resource allocation workflow.

the controller on the core cloud layer to obtain the optimal resource configuration.

Since the important component functions of the core cloud are the same as those of the edge cloud, their workflow is shown in Figure 2. The process includes the following four steps:

- (1) **User Registration and Authentication:** User information related to equipment identification, encryption data keys, and equipment attributes needs to be registered on the blockchain module. After a user sends a service request, the user's information will be authenticated.
- (2) **Resource Optimization Department:** The intelligent equipment sends a service request to the controller, and the service request invokes the RAOA-A3C algorithm in the controller to obtain the optimal resource allocation.

- (3) **Provide Services:** The controller controls the relevant service nodes to provide services to users according to the optimal resource allocation.
- (4) **Transaction Registration:** The registration transactions that include information such as interconnection, attributes, sequence of virtual files, user information, and QoS trigger the smart contract on the VSCB. The registration transaction executes the consistent process of the optimized PBFT algorithm. A new block is generated, the resource allocation transaction takes effect, and the trusted service is completed.

This paper uses the RAOA-A3C algorithm to achieve the parallel execution effect. Each controller as an agent extracts state information from the environment, and then the controller obtains the action probability by processing the state information, and then the controller calculates the

reward based on the agent's action. During the interaction between the controller and the local environment, the agent updates its local action probability according to the reward and regularly pushes its gradient to the global network.

4. System Model and Problem Formulation

The physical network of the optimized model in this paper mainly involves servers and physical links. They provide instantiated computing resources including CPU resources and storage resources and instantiated bandwidth resources for the VSFs that constitute each SFC. In this paper, CPU resources are used to represent computing resources. The physical network feeds back the current CPU resources and bandwidth resources to the RAOA-A3C in the controller. The algorithm makes decisions based on the current CPU resource status of the node, the current resource status of the link, and the current queue status in the SFC. Then the controller optimizes the resource allocation policy through the resource management entity [38]. This section introduces the network model, service request model, cost model, delay model, and optimization goals.

4.1. Network Model. The physical network is abstracted as an undirected graph $UG^P = (SN^P, Y^P)$, where SN^P represents a collection of nodes, the nodes are divided into two categories: (1) server nodes $n \in N^P$, which provide instantiated CPU resources for VSFs, and each server can instantiate multiple VSFs. And (2) switch nodes, which forward the traffic. Y^P represents a collection of physical links. C_n^P represents the CPU capacity of each server n . In order to ensure the resource utilization of the server and achieve the purpose of energy-saving, it is necessary to set a CPU resource threshold φ_n^P for the server. As long as the remaining CPU resources of the server in each time slot are less than φ_n^P , the server can be used. B_{mn}^P represents the bandwidth capacity of the physical link mn connecting adjacent servers n and m .

4.2. Service Request Model. SF represents the collection of SFCs. The i -th SFC can be formalized as an undirected graph $UG_i^N = (V_i^N, L_i^N)$, where V_i^N represents the collection of different types of VSFs on the i -th SFC and L_i^N represents the collection of virtual links on the i -th SFC. $V_{i,j}^N$ represents the j -th VSF on the i -th SFC; $C_{i,j}^N$ represents the CPU resource allocated by the server to the j -th VSF on the i -th SFC; and $B_{i,jk}^{mn}$ represents the virtual link bandwidth resource allocated by the physical link to the adjacent VSF jk on the i -th SFC. D_i represents the maximum delay limit of the i -th SFC. $\theta_{i,j}^n$ represents the mapping of VSF to the server, which is a Boolean variable. $\theta_{i,j}^n = 1$ represents the j -th VSF on the i -th SFC mapping the server n , and $\theta_{i,j}^n = 0$ represents no mapping relationship. $\eta_{i,jk}^{mn}$ represents the mapping of virtual links to physical links, which is also a Boolean variable. $\eta_{i,jk}^{mn} = 1$ represents the virtual link $l_{i,jk}^{jk}$ connecting the adjacent VSF jk on the i -th SFC that is mapped to the physical link y^{mn} connecting the adjacent server mn , and $\eta_{i,jk}^{mn} = 0$

represents no mapping relationship. This paper makes the following constraints.

In time slot t , each VSF $V_{i,j}^N$ can only select one server for mapping. That is,

$$\sum_{n \in N^P} \theta_{i,j}^n(t) = 1, \quad \forall i \in SF, \forall j \in V_i^N. \quad (1)$$

The binary variable that represents the mapping of VSF to the server is expressed as follows:

$$\theta_{i,j}^n(t) \in \{0, 1\}, \quad \forall i \in SF, \forall j \in V_i^N, \forall n \in N^P. \quad (2)$$

In time slot t , the amount of CPU resources allocated by the server should not exceed its CPU capacity C_n^P so that the system stability can be guaranteed. That is,

$$\sum_{i \in SF} \sum_{j \in V_{i,j}^N} \theta_{i,j}^n(t) C_{i,j}^n(t) \leq C_n^P, \quad \forall n \in N^P. \quad (3)$$

In time slot t , the remaining CPU capacity $R_n(t)$ of the server n can be expressed as the CPU capacity C_n^P minus the amount of CPU resources. That is,

$$R_n(t) = C_n^P - \sum_{i \in SF} \sum_{j \in V_{i,j}^N} \theta_{i,j}^n(t) C_{i,j}^n(t), \quad \forall n \in N^P. \quad (4)$$

And the constraint is as follows:

$$R_n(t) \leq \varphi_n^P, \quad \forall n \in N^P. \quad (5)$$

In time slot t , each virtual link $l_{i,jk}^{jk}$ connected to adjacent VSF jk can only select one physical link y^{mn} connected to adjacent server mn for mapping. That is,

$$\sum_{n,m \in N^P} \eta_{i,jk}^{mn}(t) = 1, \quad \forall i \in SF, \forall j, k \in V_i^N. \quad (6)$$

The binary variable that represents the mapping of virtual links to physical links is expressed as follows:

$$\eta_{i,jk}^{mn}(t) \in \{0, 1\}, \quad \forall i \in SF, \forall j, k \in V_i^N, \forall n, m \in N^P. \quad (7)$$

In time slot t , the amount of bandwidth resources allocated by physical link mn cannot exceed its bandwidth capacity B_{mn}^P . That is,

$$\sum_{i \in SF} \sum_{j,k \in V_{i,j}^N} \theta_{i,j}^n(t) \theta_{i,k}^m(t) \eta_{i,jk}^{mn}(t) B_{i,jk}^{mn}(t) \leq B_{mn}^P, \quad \forall n, m \in N^P. \quad (8)$$

In time slot t , the remaining bandwidth resource $R_{mn}(t)$ can be expressed by the bandwidth capacity B_{mn}^P minus the bandwidth resource. That is,

$$R_{mn}(t) = B_{mn}^P - \sum_{i \in SF} \sum_{j,k \in V_{i,j}^N} \theta_{i,j}^n(t) \theta_{i,k}^m(t) \eta_{i,jk}^{mn}(t) B_{i,jk}^{mn}(t), \quad \forall n, m \in N^P. \quad (9)$$

4.3. Cost Model. The allocation cost of resource allocation mainly includes the cost of occupying CPU resources $OC_{i,j}^n$ and the cost of occupying physical link bandwidth resources $OB_{i,jk}^{mn}$ [47]. $OC_{i,j}^n(t)$ is inversely proportional to the

remaining CPU resource $R_n(t)$ of server n in time slot t . That is,

$$OC_{i,j}^n(t) = \frac{\alpha}{R_n(t)}, \quad (10)$$

where α is a positive number.

$OB_{i,jk}^{nm}$ is inversely proportional to the remaining bandwidth resources $R_{nm}(t)$ of physical link nm , that is,

$$OB_{i,jk}^{nm}(t) = \frac{\beta}{R_{nm}(t)}, \quad (11)$$

where β is a positive number.

In summary, in time slot t , the resource allocation cost on the i -th SFC is

$$\begin{aligned} O_i(t) &= \sum_{j \in V_i^N} \sum_{n \in N^P} \theta_{i,j}^n(t) \frac{\alpha}{R_n(t)} \\ &+ \sum_{j,k \in V_i^N} \sum_{n,m \in N^P} \theta_{i,j}^n(t) \theta_{i,k}^m(t) n_{i,jk}^{mn}(t) \frac{\beta}{R_{nm}(t)}, \quad \forall i \in SF. \end{aligned} \quad (12)$$

4.4. Delay Model. The optimization model not only gives the attributes and order of VSFs but also provides QoS constraints. The delay of this model mainly considers the queuing delay, processing delay, and link transmission delay. We take the i -th SFC as an example; $Q_i(t)$ represents its queue length in time slot t ; $P_i(t)$ represents the size of the data packet, and it is assumed that the size of the data packet obeys the exponential distribution of parameter \bar{P} ; and $A_i(t)$ represents the data packet arrival process of the i -th SFC, and it is assumed that the arrival of data packets obeys the Poisson distribution with a parameter of λ_i [48]. The update process of the queue is expressed as follows:

$$Q_i(t+1) = \max[Q_i(t) - \ell_{i,1}(t) + A_i(t), 0], \quad (13)$$

where $\ell_{i,1}(t)$ represents the first VSF service rate of the i -th SFC and the service rate $\ell_{i,j}(t)$ of the j -th VSF on the i -th SFC is determined by the amount of CPU resources allocated to the j -th VSF by the server, that is, $\ell_{i,j}(t) = C_{i,j}^n \cdot \varepsilon$, where ε is the service rate coefficient, which represents the ratio between CPU resources and service rate [49]. The constraints of the delay model are as follows.

$Q_i^{\max}(t)$ represents the maximum queue length of the i -th SFC. In order to ensure that the queue length does not overflow, $Q_i(t)$ satisfies

$$Q_i(t) < Q_i^{\max}(t), \quad \forall i \in SF. \quad (14)$$

T_i^{wait} represents the queuing delay of the i -th SFC. According to little theorem, the queuing delay T_i^{wait} is

$$T_i^{\text{wait}}(t) = \frac{Q_i(t)}{\lambda_i(t)}. \quad (15)$$

$T_{i,j}^{\text{proc}}$ represents the processing delay generated by each VSF, and $T_{i,j}^{\text{proc}}$ is

$$T_{i,j}^{\text{proc}}(t) = \frac{DR_{i,j}(t)}{\ell_{i,j}(t)}. \quad (16)$$

$DR_{i,j}(t)$ represents the amount of data packets arriving at VSF $V_{i,j}^N$ in time slot t , and the processing delay $T_i^{\text{proc}}(t)$ of the i -th SFC is

$$T_i^{\text{proc}}(t) = \sum_{j \in V_i^N} T_{i,j}^{\text{proc}}(t). \quad (17)$$

$T_{i,jk}^{\text{tran}}$ represents the transmission delay of the amount of data. $T_{i,jk}^{\text{tran}}$ is related to the amount of data transmitted and the bandwidth resources allocated by the physical link. $T_{i,jk}^{\text{tran}}$ is

$$T_{i,jk}^{\text{tran}}(t) = \frac{DR_{i,jk}(t)}{B_{i,jk}^{nm}(t)}. \quad (18)$$

$DR_{i,jk}(t)$ represents the amount of data from VSF $V_{i,j}^N$ to VSF $V_{i,k}^N$, that is, the amount of data packets arriving at VSF $V_{i,k}^N$ in time slot t . The transmission delay $T_i^{\text{tran}}(t)$ of the i -th SFC in time slot t is

$$T_i^{\text{tran}}(t) = \sum_{j,k \in V_i^N} T_{i,jk}^{\text{tran}}(t). \quad (19)$$

In summary, the total delay $T_i(t)$ of the i -th SFC is

$$T_i(t) = T_i^{\text{wait}}(t) + T_i^{\text{proc}}(t) + T_i^{\text{tran}}(t). \quad (20)$$

And the constraints are as follows:

$$T_i(t) < D_i, \quad \forall i \in SF. \quad (21)$$

4.5. Optimization Goals. The main optimization goal of this paper is to minimize the cost of resource allocation under the premise of ensuring security and meeting the requirements of delay. CPU resources and physical link bandwidth resources are reasonably allocated, which is conducive to the realization of low-delay and low-cost resource allocation. The utility function $U(t)$ is defined as follows:

$$U(t) = -e1 \frac{\sum_{i \in SF} O_i(t)}{O_{\max}} - e2 \frac{\sum_{i \in SF} T_i(t)}{\sum_{i \in SF} D_i(t)}, \quad (22)$$

where $e1$ and $e2$ are the weight values and $e1 + e2 = 1$. O_{\max} represents the maximum value of the allocation cost. After the algorithm normalizes the allocation cost, the optimization goal is expressed as follows:

$$\begin{aligned}
& \max_{\theta_{i,j}^n(t), \theta_{i,k}^m(t), \eta_{i,jk}^{mn}(t), C_{i,j}^n(t), B_{i,jk}^{mn}(t)} U(t), \\
& \text{s.t. C1: } \sum_{n \in N^P} \theta_{i,j}^n(t) = 1, \quad \forall i \in SF, \forall j \in V_i^P, \\
& \text{C2: } \sum_{m,n \in N^P} \eta_{i,jk}^{mn}(t) = 1, \quad \forall i \in SF, \forall j, k \in V_i^P, \\
& \text{C3: } \sum_{i \in SF} \sum_{j \in V_{i,j}^N} \theta_{i,j}^n(t) C_{i,j}^n(t) \leq C_n^P, \quad \forall n \in N^P, \\
& \text{C4: } \sum_{i \in SF} \sum_{j,k \in V_{i,j}^N} \theta_{i,j}^n(t) \theta_{i,k}^m(t) \eta_{i,jk}^{mn}(t) B_{i,jk}^{mn}(t) \leq B_{nm}^P, \quad \forall n, m \in N^P, \\
& \text{C5: } R_n(t) \leq \phi_n^P, \quad \forall n \in N^P, \\
& \text{C6: } Q_i(t) < Q_i^{\max}(t), \quad \forall i \in SF, \\
& \text{C7: } T_i(t) < D_i, \quad \forall i \in SF, \\
& \text{C8: } \theta_{i,j}^n(t) = \{0, 1\}, \quad \forall i \in SF, \forall j \in V_i^N, \forall n \in N^P, \\
& \text{C9: } \eta_{i,jk}^{mn}(t) = \{0, 1\}, \quad \forall i \in SF, \forall j, k \in V_i^N, \forall n, m \in N^P,
\end{aligned} \tag{23}$$

where C1 guarantees that each VSF in the virtual network can only select one server in the physical network for mapping. C2 guarantees that the virtual link of adjacent VSFs can only select the physical link of adjacent servers in the physical network for mapping. C3 guarantees that the sum of the CPU resources allocated by each server cannot exceed the CPU capacity of the server. C4 makes the sum of all communication resources mapped to a certain physical link not exceed the total bandwidth resources of the physical link. C5 makes the remaining CPU resources of each server lower than the threshold, guarantees the resource utilization of the server, and further achieves the effect of energy-saving. C6 guarantees that the queue length of each SFC does not overflow. C7 guarantees that each SFC must meet the delay requirement in any time slot. C8 and C9 are requirements for binary variables. In summary, the utility function is restricted by the C1–C7 constraints to ensure the effectiveness of the optimization objective.

5. Proposed Algorithm

In this section, the resource allocation optimization problem of the power video monitoring system is modeled as an MDP model, and then the RAOA-A3C algorithm is proposed in the cloud-edge computing environment to achieve the goals of security protection and efficient resource management.

5.1. Problem Transformation. The MDP model mainly includes state space, action space, transition probability, and reward function [38].

5.1.1. State Space. S represents the state space, which is mainly composed of the queue status of each SFC, the remaining CPU resources of the server, and the remaining resources of the physical link bandwidth. $s(t)$ represents the

state of the network in time slot t , which is expressed as follows:

$$s(t) = \{Q_i(t), R_n(t), R_{nm}(t)\} \quad \forall m, n \in N^P, \forall i \in SF. \tag{24}$$

5.1.2. Action Space. A represents the action space, which mainly includes allocating CPU resource $C_{i,j}^n(t)$, allocating bandwidth resource allocation $B_{i,jk}^{mn}(t)$, and deploying $\theta_{i,j}^n(t)$ and $\eta_{i,jk}^{mn}(t)$. $a(t)$ represents the action taken by the network in time slot t , which is expressed as follows:

$$a(t) = \{C_{i,j}^n(t), B_{i,jk}^{mn}(t), \theta_{i,j}^n(t), \eta_{i,jk}^{mn}(t)\} \quad \forall n, m \in N^P, \tag{25}$$

$$\forall i \in SF, \forall j, k \in V_i^N.$$

5.1.3. Transition Probability. In time slot t , there is a probability that the network state $s(t)$ takes action $a(t)$ and transitions to the network state $s(t+1)$. $\Pr(s(t), a(t), s(t+1))$ represents the transition probability, and $\Pr(s(t), a(t), s(t+1))$ is expressed as follows:

$$\begin{aligned}
& \Pr(Q_i(t), a(t), Q_i(t+1)) \\
\Pr(s(t), a(t), s(t+1)) = & \Pr(R_n(t), a(t), R_n(t+1)) \cdot \\
& \Pr(R_{nm}(t), a(t), R_{nm}(t+1))
\end{aligned} \tag{26}$$

5.1.4. Reward Function. This section uses the aforementioned utility function as a reward. $r(s(t), a(t))$ represents the reward function, which is the reward after the network state $s(t)$ takes an action $a(t)$. $r(s(t), a(t))$ can be expressed as follows:

$$r(s(t), a(t)) = U(t). \tag{27}$$

5.2. Algorithm Description. Based on formula (23) and the MDP model, the key problem to be solved in this paper is to determine the target server and resource allocation policy. The algorithm obtains the optimal value function of the state of each slot and then obtains the optimal action corresponding to the state, that is, the optimal action of each slot constitutes the optimal policy π^* . This paper introduces the DRL algorithm to solve the optimization problem of the MDP model. DRL uses DNN to effectively identify high-dimensional state spaces and uses RL algorithms to learn complex tasks in an end-to-end manner. DRL does not require complicated manual preprocessing of state features.

Most intelligent monitoring equipment have the capability of parallel computing. A3C is an asynchronous actor-critic parallel learning algorithm based on the advantage function. It is a lightweight DRL framework. The framework uses an asynchronous gradient descent method to optimize the parameters of the controller, which is suitable for solving the problems of too large state space and the high dimension of action space in the optimal allocation of resources. This paper proposes the RAOA-A3C algorithm to solve the MDP model.

As shown in Figure 2, after the user sends a service request and performs security authentication, the controller in the server node collects the environment status and takes actions to react to the status. The general workflow of the algorithm is shown in Figure 3. The environment state is provided to the actor network and the critic network, and the policy and the value function are obtained, respectively. The actor executes the action, and then the critic evaluates whether the action is good or bad. The policy π is a function of state s , which returns the probability distribution of all actions, and sums up to 1. That is, $\pi(a|s)$ represents the probability of choosing action $a(t)$ in state $s(t)$. In the actual execution process, the actor selects actions based on the distribution of policy $\pi(a(t)|s(t); \theta)$ or directly selects the action with the highest probability. Accordingly, the critic evaluates the current policy based on the TD error between the value function $V(s(t); \theta_v)$ and the current reward, where θ is the actor network parameter, θ_v is the critic network parameter, and TD-error is used to update θ to correct the action probability; θ_v can improve the accuracy of the value evaluation.

The algorithm uses the following iterative definition as the value function $V(s)$ of the expected discount return:

$$V(s) = E_{\pi(s)}(r + \gamma V(s')). \quad (28)$$

The return obtained in the current state is the sum of the return obtained in the next state and reward r obtained during the state transition, where γ represents the discount factor in RAOA-A3C.

There is also an action value function $Q(s, a)$ closely related to the value function, which is defined as follows:

$$Q(s, a) = r + \gamma V(s'). \quad (29)$$

The advantage function $A(s, a)$ is defined as follows:

$$A(s, a) = Q(s, a) - V(s) = r + \gamma V(s') - V(s), \quad (30)$$

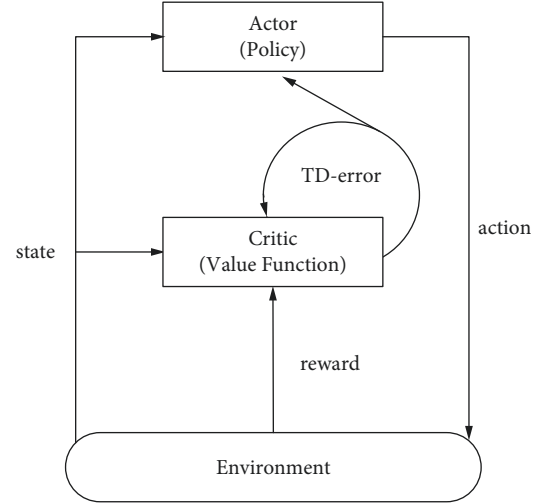


FIGURE 3: Algorithm basic workflow.

where $A(s, a)$ represents that action a is good or bad in state s . If action a is better than average, then $A(s, a)$ is positive; otherwise, it is negative.

The algorithm defines the objective function $J(\pi)$ used to measure the quality of the policy as follows:

$$J(\pi) = E_{\rho^{\pi_0}}[V(s_0)], \quad (31)$$

where $J(\pi)$ represents all the average discount rewards obtained by a policy starting from the initial state s_0 .

According to the policy gradient theorem, the algorithm can obtain the definition of the gradient of the objective function:

$$\nabla_{\theta} J(\pi) = E_{s \sim \rho^{\pi}, a \sim \pi(s)}[A(s, a) \cdot \nabla_{\theta} \log \pi(a|s)]. \quad (32)$$

The function obtains the reward obtained from sample (s_0, a_0, r_0, s_1) , and then the function predicts the value in the next step and provides an estimated approximation. However, the function uses more steps to provide n -step return.

$$V(s_0) \longrightarrow r_0 + \gamma r_1 + \dots + \gamma^n V(s_n). \quad (33)$$

The advantage of the n -step return is that the change in the approximate function propagates is faster.

By extending the advantage function $A(s, a)$, the gradient $d\theta$ of the update policy π in the actor network can be obtained as follows:

$$d\theta \longleftarrow d\theta + \nabla_{\theta'} \log \pi(a_t|s_t; \theta') \times \left[\sum_{i=0}^{k-1} \gamma^i r(t+i) + \gamma^k V(s_{t+k}; \theta'_v) - V(s_t; \theta'_v) \right] + \delta \nabla_{\theta'} H(\pi(s_t; \theta')), \quad (34)$$

where H represents the entropy to avoid premature convergence to the suboptimal deterministic policy. δ is the entropy hyperparameter, which is used to control the strength of the entropy regularization term. θ'_v is a parameter of the state value function in the critic network, descending

Input:

- (1) Initialize the actor network with parameter θ
- (2) Initialize the critic network with parameter θ_v
- (3) Initialize the actor network parameters θ'_i for each thread
- (4) Initialize the critic network parameters θ'_v for each thread
- (5) for each SFC $f \in SF$ do
- (6) for each VSF resource allocation scheme x in $X_{n,v}$ in edge cloud do
- (7) if x is not registered in the VSCB then
- (8) remove x from $X_{n,v}$
- (9) end for
- (10) if $X_{n,v}$ is empty then
- (11) Calculate $X_{n,v}$ in core cloud
- (12) Select the VSF allocation method $\theta'_{i,j}$ from $X_{n,v}$
- (13) Select the VSF allocation method $\eta'_{i,j,k}$ from $X_{n,v}$
- (14) Calculate the total cost
- (15) end for
- (16) while the SF set is not empty do
- (17) Set the gradient of the two networks $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$
- (18) Synchronize the parameters of the thread $\theta' \leftarrow \theta$ and $\theta'_v \leftarrow \theta_v$, $t_{\text{start}} = t$
- (19) Obtain state s_t
- (20) Repeat:
- (21) The state features are extracted from the network based on a multiple threshold mechanism, and action a_t is executed according to policy $\pi(a_t|s_t; \theta')$
- (22) Get the reward r_t of environmental feedback and the next state s_{t+1}
- (23) $t \leftarrow t + 1$
- (24) Until termination status S_t , or $t - t_{\text{start}} = t_{\text{max}}$
- (25) Obtain $R = \begin{cases} 0 & \text{for } \text{termin al } s_t \\ V(s_t, \theta'_v) & \text{for non-termin al } s_t \end{cases}$ from the critic network
//Boot strap from last state
- (26) for each step $t \leftarrow t - 1, \dots, t_{\text{start}}$ do
- (27) $R = r_t + \gamma R$
- (28) Calculate the gradient of the actor network about θ' :
$$d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_t|s_t; \theta') \times [\sum_{i=0}^{k-1} \gamma^i r(t+i) + \gamma^k V(s_{t+k}; \theta'_v) - V(s_t; \theta'_v)] + \delta \nabla_{\theta'} H(\pi(s_t; \theta'))$$
- (29) Calculate the gradient of the critic network about θ'_v :
$$d\theta_v \leftarrow d\theta_v + \partial (\sum_{i=0}^{k-1} \gamma^i r(t+i) + \gamma^k V(s_{t+k}; \theta'_v) - V(s_t; \theta'_v))^2 / \partial \theta'_v$$
- (30) $t \leftarrow t - 1$
- (31) end for
- (32) Use $d\theta$ and $d\theta_v$ to asynchronously update the global shared parameters θ and θ_v . Send the $d\theta$ and $d\theta_v$ to SFC agent
- (33) end while

Output: $\pi^*(s)$

ALGORITHM 1: RAOA-A3C algorithm.

by a gradient in TD mode. The updated gradient $d\theta_v$ in the critic network is as follows:

$$d\theta_v \leftarrow d\theta_v + \partial \frac{(\sum_{i=0}^{k-1} \gamma^i r(t+i) + \gamma^k V(s_{t+k}; \theta'_v) - V(s_t; \theta'_v))^2}{\partial \theta'_v} \quad (35)$$

The network structure of the RAOA-A3C algorithm mainly uses convolutional neural networks and fully connected neural networks, and the output of the fully connected layer is used as the input of the actor network and the critic network. The actor network outputs the corresponding

action value to select actions; the critic network outputs a state value to calculate the advantage.

We use Algorithm 1 to solve equation (23). The pseudocode of the RAOA-A3C Algorithm 1 is described as follows:

The RAOA-A3C algorithm consists of two parts. It mainly includes network initialization and resource allocation optimization. $X_{n,v}$ represents a feasible resource allocation scheme. The mapping service node x of the VSFs in the edge cloud needs to be authenticated through the blockchain module. If it is not registered on the VSCB, x will be deleted from the configuration scheme set. Then, when the edge cloud resource configuration scheme is empty,

select the core cloud and authenticate again. VSF resource allocation methods are randomly selected from the available configuration scheme $X_{n,v}$. The resources can be optimally allocated.

6. Performance Evaluation and Analysis

6.1. Simulation Settings. In the experiment, Docker 18.06, Python 3.0, TensorFlow, and OpenAI Gym were installed in Ubuntu 16.04 to configure the environment, and MATLAB was used for simulation experiments. Other related settings are as follows: the virtual machines are interconnected via a 1Mbit/s virtual LAN card, and each blockchain node has a 2.0 GHz 8-VCPUs attribute. There are 30 nodes (10 server nodes and 20 switches) and 50 links. The discount factor γ is 0.9, and the entropy hyperparameter δ is 0.1 [10]. The maximum delay limit of SFC D_i is 30 ms; the data packet arrival process follows the independent identical distributed Poisson process; and the parameter value is $\lambda_i = 2$. The packet size is 500 kb/packet; the physical link bandwidth resource is 640 MB; the CPU resource capacity of server n is 8 cores; the service rate of a single CPU is $\ell = 25MB/s$; the positive number is $\alpha = 30$; and the positive number $\beta = 20$ [38].

6.2. Performance Evaluation. Firstly, eight and ten consortium peers were deployed on the core cloud and the edge cloud, respectively; for comparison, it is verified that the RAOA-A3C algorithm has the performance of consistent delay.

As shown in Figure 4, the consensus efficiency of the core cloud is higher than the edge cloud. The delay increases significantly as the number of SFCs increases. The reason is that users who send service requests need to be authenticated; SFC transactions also need to be registered on the VSCB; and when the number of consortium partners increases, the consensus delay also increases with the increase in the number of SFCs.

The weighted values $e1$ and $e2$ are iterated by using max equality constraints, optimality constraints, and max inequality constraints. Assuming that 200 SFCs arrive in the virtual network, and after 10,000 iterations, the allocation cost and average delay are shown in Figures 5 and 6.

As shown in Figures 5 and 6, after 10,000 iterations, the allocation cost and the average delay under different constraints are obtained. When the number of iterations reaches about 6,000, the convergence is obvious. The algorithm is effective.

In the following, the RAOA-A3C algorithm is compared with DQN [21] and A3C [10] algorithms in the three aspects of total allocation cost, average delay, and utility function.

As shown in Figures 7 and 8, as the number of SFCs increases, the average delay and total allocation cost of the three algorithms are increasing. The RAOA-A3C algorithm has a lower allocation cost than DQN and A3C algorithms. DQN algorithm is mainly suitable for solving the discrete action space, but the action space in this paper is a continuous value, which causes the DQN algorithm to be

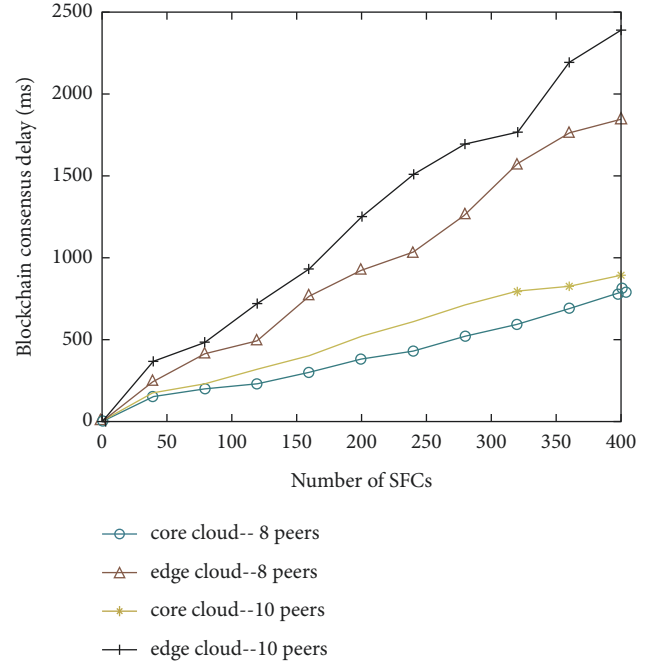


FIGURE 4: Blockchain conformance delay.

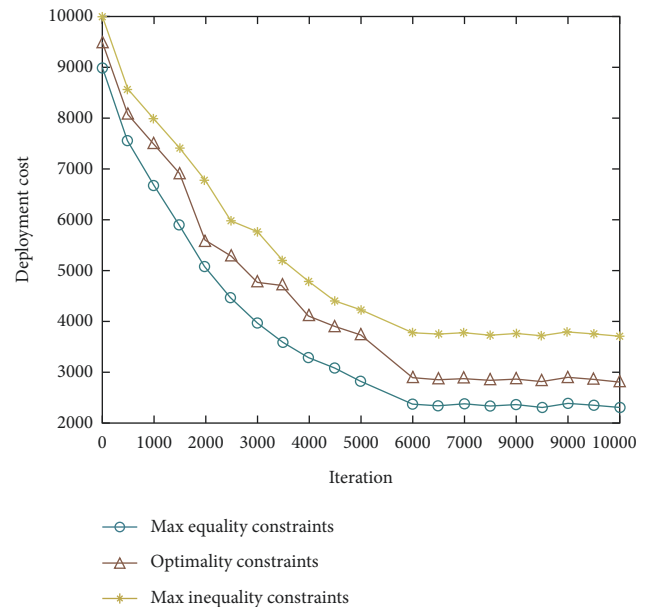


FIGURE 5: Comparison of allocation cost under different constraints.

significantly weaker than the A3C algorithm and the RAOA-A3C algorithm in optimizing the delay and allocation cost.

As shown in Figure 8, when the number of SFCs is less than 100, the RAOA-A3C algorithm is higher than the A3C algorithm in the system average delay. The reason is that there is a time delay in the resource authentication stage. However, as the number of SFCs increases, the proportion of time delay in the blockchain becomes smaller, and the influence becomes weaker, and advantages of the RAOA-A3C algorithm are revealed.

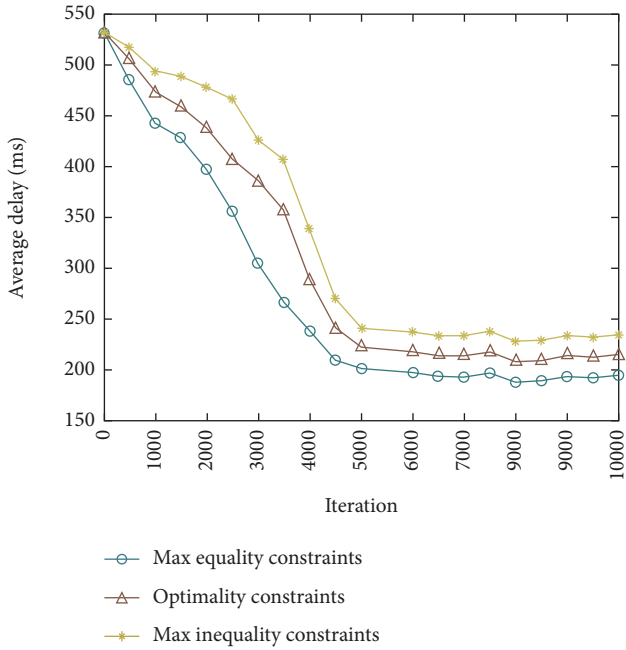


FIGURE 6: Comparison of average time delays under different constraints.

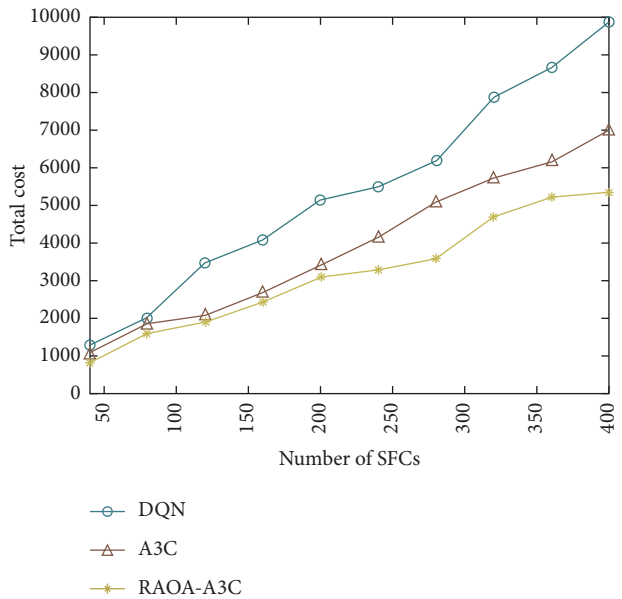


FIGURE 7: Total cost of allocation comparison.

As shown in Figure 9, the total system delay and total allocation cost increase; as the number of SFCs increases, the utility will also decrease as the number of SFCs increases, but the utility of the RAOA-A3C algorithm proposed in this paper decreases the slowest.

As shown in Figure 10, when the number of SFCs is less than 200, the RAOA-A3C algorithm fluctuates due to the influence of the consortium blockchain. When the number of SFCs is greater than 200, the advantages of RAOA-A3C

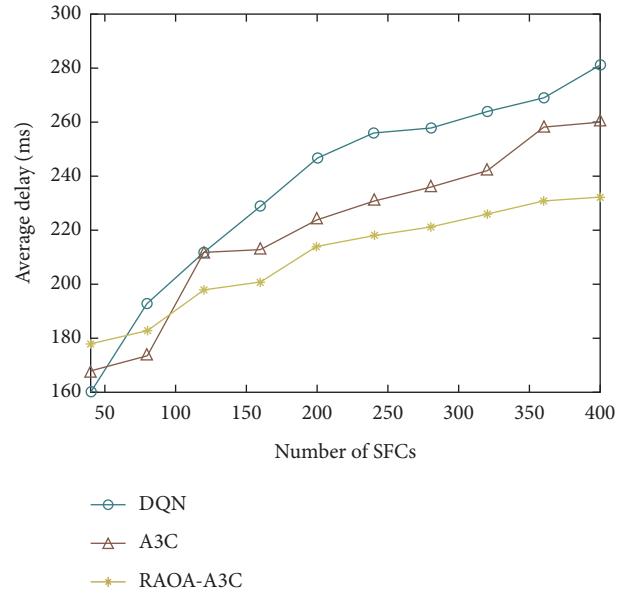


FIGURE 8: Average delay comparison.

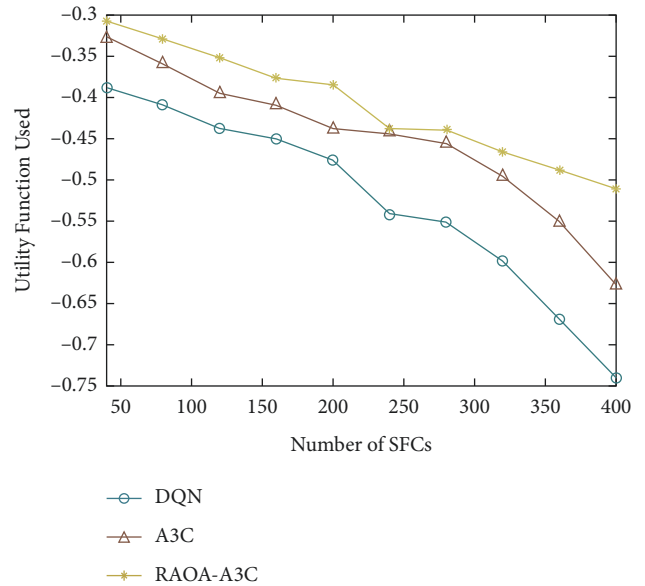


FIGURE 9: Utility function comparison.

are better reflected, and the variance of server usage of RAOA-A3C is lower than that of DQN and A3C.

As shown in Figure 11, the variance of the utilization rate of the link of RAOA-A3C is lower than that of DQN and A3C on the whole.

As shown in Figures 10 and 11, the experiment compares the variance of the link usage rate and the variance of the server usage rate of the DQN, A3C, and RAOA-A3C algorithms. It can be seen that a smaller difference indicates that the service is more evenly distributed on the server and the link. The RAOA-A3C algorithm congestion control results are better.

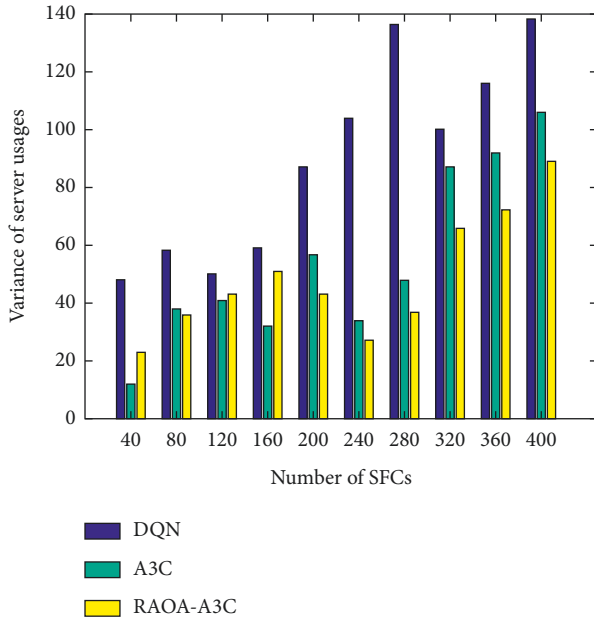


FIGURE 10: Comparison of server usage under different numbers of SFCs.

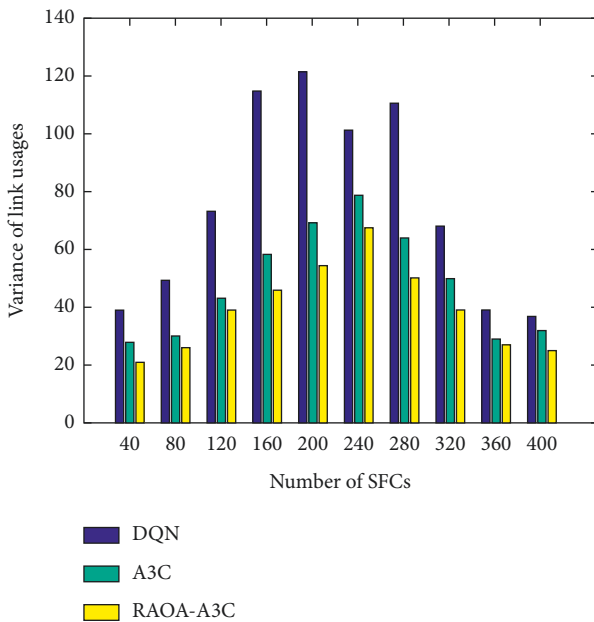


FIGURE 11: Comparison of link usage under different numbers of SFCs.

7. Conclusion

Although the power video monitoring system based on the cloud-edge computing architecture brings many benefits. The problems of distribution and heterogeneity cannot guarantee the reliability and durability of the service. In order to establish trust between service providers and users, VSCB is integrated into the management of the power video monitoring system. In addition, the SFV technology was first proposed to realize the optimal allocation of resources; the

MDP model was constructed; and then the RAOA-A3C algorithm was proposed. Simulation experimental results show the advantages of the resource allocation optimization model in cost-saving, time-saving, and security.

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 funded by the Provincial Science and Technology Innovation Special Fund Project of Jilin Province, grant no. 20190302026GX; Natural Science Foundation of Jilin Province, grant no. 20200201037JC; the Higher Education Research Project of Jilin Association for Higher Education, grant no. JGJX2018D10; the Fundamental Research Funds for the Central Universities for JLU, Platform of Jilin Province Science and Technology Department, grant no. 20190902011TC; 2021 Digital transformation and innovation platform construction project of Jilin Provincial Development and Reform Commission, grant nos. 2021C049; 2020 Industrial Technology Research and Development Project of Jilin Development and Reform Commission, grant no. 2020C020-1; 2019 Jilin Province S&T Development Plan Technology Research Project Research Project, grant no. 20190302115GX; Changchun Philosophy and Social Science Planning Project, grant no. CSKT2021ZX-054; and Changchun Institute of Technology Science and Technology Fund Project, grant no. 320200010.

References

- [1] Y. Y. Sun, T. Jia Zhen, and H. Y. Zhu, "A review of multimodal deep learning," *Computer Engineering and Applications*, vol. 56, no. 21, pp. 1-10, 2020.
- [2] B. Wang, B. H. Wang, R. Zhang, and F. J. Li, "Research on the security architecture of power wireless communication system based on TD-LTE," *Microcomputer Applications*, vol. 36, no. 7, pp. 61-63, 2020.
- [3] S. Zhang, J. Zhang, Y. Wang, J. L. Liu, B. Yan, and Z. W. Shang, "Research on pedestrian Re-identification network architecture technology based on multi-view and cross-modality in power field," *Sichuan Power Technology*, vol. 43, no. 6, pp. 6-10+15, 2020.
- [4] L. Yang, M. Li, X. Y. Ye, E. C. Sun, and Y. H. Zhang, "Research on optimal allocation of industrial Internet resources by integrating edge computing and blockchain," *High Technology Communications*, vol. 30, no. 12, pp. 1253-1263, 2020.
- [5] P. Yu, H. Yan, H. L. Chen, and D. Long, "Joint optimization of collaborative edge caching and resource allocation for mobile terminal energy saving based on edge computing," *Test Engineering and Management*, vol. 83, no. 5-6, Article ID 18994, 2020.
- [6] Aliyun Computing Co Ltd, "China Institute of standardization of electronic technology," *White Paper on Edge Cloud Computing Technology and Standardization*, 2018.

- [7] M. S. Munir, S. F. Abedin, N. H. Tran, and C. S. Hong, "When edge computing meets microgrid: a deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7360–7374, 2019.
- [8] M. Crosby and P. N. Pattanayak, "Blockchain technology: beyond bitcoin," *Applied Innovation Review*, vol. 2, pp. 6–19, 2016.
- [9] Y. Sun, S. Yang, G. J. Gong, J. X. Yang, and B. Zhou, "Research on endogenous security of power distribution Internet of things based on trusted computing and blockchain," *Huadian Technology*, vol. 42, no. 8, pp. 61–67, 2020.
- [10] S. Guo, Y. Dai, S. Xu, and F. Qi, "Trusted cloud-edge network resource management: DRL-driven service function chain orchestration for IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6010–6022, 2020.
- [11] X. Y. Zhang, C. S. Li, H. C. Shi, L. Peng, and D. Jing, "AdapNet: adaptability decomposing encoder-decoder network for weakly supervised action recognition and localization," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2020.
- [12] X. Y. Zhang, S. P. Wang, and X. C. Yun, "Bidirectional active learning: a two-way exploration into unlabeled and labeled dataset," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3034–3044, 2015.
- [13] X. Y. Zhang, H. C. Shi, C. S. Li, and L. Peng, "Multi-instance multi-label action recognition and localization based on spatio-temporal pre-trimming for untrimmed videos," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 12886–12893, 2020.
- [14] D. B. Zhao, K. Shao, Y. H. Zhu et al., "A review of deep reinforcement learning: concurrently discuss the development of computer go," *Control Theory & Applications*, vol. 33, no. 6, pp. 701–717, 2016.
- [15] J. G. Wu, T. L. Liu, J. Y. Li, and J. Y. Huang, "Research progress of blockchain technology in mobile edge computing," *Computer Engineering*, vol. 46, no. 8, pp. 1–13, 2020.
- [16] C. Li, X. D. Duan, and W. Chen, "Thoughts and practices about SDN and NFV," *Telecommunications Science*, vol. 30, no. 8, pp. 23–27, 2014.
- [17] M. Jin, L. L. Li, W. J. Zhang, and W. Liu, "Service function chain mapping algorithm based on deep reinforcement learning," *Application Research of Computers*, vol. 37, no. 11, pp. 3456–3460+3466, 2020.
- [18] W. H. Zhan, J. Wang, Q. X. Zhu, H. C. Duan, and Y. L. Ye, "Computational offloading scheduling method based on deep reinforcement learning in mobile edge computing," *Application Research of Computers*, vol. 38, no. 1, pp. 241–245+263, 2021.
- [19] D. Silver, T. Hubert, J. Schrittwieser et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.
- [20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Anderj, J. Veness, and A. Graves, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [21] D. P. Wu, Q. R. Liu, H. G. Wang, W. Dalei, and W. Ruyan, "Socially aware energy-efficient mobile edge collaboration for video distribution," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2197–2209, 2017.
- [22] S. Z. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.
- [23] L. X. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.
- [24] Y. J. Zhu, "A survey on mobile edge platform with blockchain," in *Proceedings of the 3rd Information Technology, Networking, Electronic and Automation Control Conference*, pp. 879–883, IEEE Press, Washington D.C., USA, March 2019.
- [25] H. Jiang, F. Shen, S. Chen, and K.-C. Li, "A secure and scalable storage system for aggregate data in IoT[J]," *Future Generation Computer Systems*, vol. 49, pp. 133–141, 2015.
- [26] G. Oliva, S. Cioaba, and C. N. Hadjicostis, "Distributed calculation of edge-disjoint spanning trees for robustifying distributed algorithms against man-in-the-middle attacks," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 1646–1656, 2018.
- [27] J. J. Fang and K. Lei, "Summary of blockchain technology for edge artificial intelligence computing," *Journal of Applied Sciences*, vol. 38, no. 1, pp. 1–21, 2020.
- [28] W. Tang, X. Zhao, W. Rafique, and W. Dou, "A blockchain-based offloading approach in fog computing environment," in *Proceedings of the IEEE Intl Conf on Ubiquitous Computing & Communications; IEEE Intl Conf on Parallel & Distributed Processing with Applications; IEEE Intl Conf on Social Computing & Networking; IEEE Intl Conf on Big Data & Cloud Computing; IEEE Intl Conf on Sustainable Computing & Communications*, Melbourne, VIC, Australia, December 2018.
- [29] Z. F. Ma, X. C. Wang, D. K. Jain, H. Khan, G. Hongmin, and W. Zhen, "A blockchain-based trusted data management scheme in edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2013–2021, 2020.
- [30] A. Jindal, G. S. Aujla, and N. Kumar, "SURVIVOR: a blockchain based edge-as-a-service framework for secure energy trading in SDN-enabled vehicle-to-grid environment," *Computer Networks*, vol. 153, pp. 36–48, 2019.
- [31] M. Samaniego and R. Deters, "Hosting virtual IoT resources on edge-hosts with blockchain," in *Proceedings of the IEEE International Conference on Computer and Information Technology (CIT)*, pp. 116–119, Nadi, Fiji, December 2016.
- [32] M. Samaniego and R. Deters, "Virtual resources & blockchain for configuration management in IoT," *Journal of Ubiquitous Systems and Pervasive Networks*, vol. 9, no. 2, pp. 1–13, 2017.
- [33] P. Vizarrreta, M. Condoluci, and C. M. Machuca, "QoS-driven function placement reducing expenditures in NFV deployments," in *Proceedings of the IEEE International Conference on Communications*, pp. 1–7, IEEE Press, Paris, France, May 2017.
- [34] G. Xiong, Y. X. Hu, L. Tian, and J.-L. Lan, "A virtual service placement approach based on improved quantum genetic algorithm," *Frontiers of Information Technology & Electronic Engineering*, vol. 17, no. 7, pp. 661–671, 2016.
- [35] J. Sun, G. Y. Zhu, G. Sun, and D. Liao, "A reliability-aware approach for resource efficient virtual network function deployment," *IEEE Access*, vol. 12, no. 6, Article ID 18238, 2018.
- [36] C. Qiu, F. R. Yu, H. P. Yao, and C. Zhano, "Blockchain based software-defined industrial Internet of Things: a dueling deep Q-learning approach," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4627–4639, 2019.
- [37] W. H. Zhan, C. B. Luo, J. Wang, G. Min, and H. Duan, "Deep reinforcement learning-based computation offloading in vehicular edge computing," in *Proceedings of the IEEE Global Communications Conference*, pp. 1–6, IEEE Press, Piscataway, NJ, USA, December 2019.

- [38] L. Q. He, *Research on Virtual Network Function Deployment and Migration Optimization Algorithm Based on Deep Reinforcement Learning*, Chongqing University of Posts and Telecommunications, Chongqing, China, 2020.
- [39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: an Introduction*, pp. 51–85, The MIT Press, Massachusetts, MA, USA, 1998.
- [40] T. W. Kuo, B. H. Liou, and C. J. Lin, “Deploying chains of virtual network functions: On the relation between link and server usage,” *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1562–1576, 2018.
- [41] Z. L. Ye, X. J. Cao, J. P. Wang, H. Yu, and C. Qiao, “Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization,” *IEEE Network*, vol. 30, no. 3, pp. 81–87, 2016.
- [42] M. Otokura, K. Leibnitz, Y. Koizumi, D. Kominami, T. Shimokawa, and M. Murata, “Evolvable virtual network function placement method: Mechanism and performance evaluation,” *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 27–40, 2019.
- [43] X. Han, Y. Yuan, and F. Y. Wang, “Security problems on blockchain: the state of the art and future trends,” *Acta Automatica Sinica*, vol. 45, no. 1, pp. 206–225, 2019.
- [44] A. M. Antonopoulos, *Mastering Blockchain Programming the Open Blockchain*, O’Reilly Media, Massachusetts, MA, USA, 2017.
- [45] J. Gao, X. G. Yan, B. Liang, and H. Guo, “Research on IIoT architecture of edge computing based on blockchain,” *Application Research of Computers*, vol. 37, no. 07, pp. 2160–2166, 2020.
- [46] Ministry of Industry and Information Technology, *China Blockchain Technology and Application Development White Paper*, 2016.
- [47] L. Tang, H. Yang, G. F. Zhao, Y. W. Wang, and Q. B. Chen, “5G network slicing node and link mapping algorithm based on delay perception,” *Journal of Beijing University of Posts and Telecommunications*, vol. 41, no. 6, pp. 71–77, 2018.
- [48] A. L. Cheng, J. Li, Y. L. Yu, and H. Jin, “Delay-sensitive user scheduling and power control in heterogeneous networks,” *IET Networks*, vol. 4, no. 3, pp. 175–184, 2015.
- [49] J. Yang, S. B. Zhang, X. M. Wu, and Y. Ran, “Online learning-based server provisioning for electricity cost reduction in data center,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 1044–1051, 2017.

Research Article

An Automatic Planning-Based Attack Path Discovery Approach from IT to OT Networks

Zibo Wang,^{1,2} Yaofang Zhang,^{1,2} Zhiyao Liu,³ Xiaojie Wei,^{1,2} Yilu Chen,^{1,2}
and Bailing Wang ^{1,2}

¹School of Computer Science and Technology, Harbin Institute of Technology, Weihai 264209, China

²School of Cyber Science and Technology, Harbin Institute of Technology, Harbin 150001, China

³China Industrial Control Systems Cyber Emergency Response Team, Beijing 100040, China

Correspondence should be addressed to Bailing Wang; wbl@hit.edu.cn

Received 30 July 2021; Revised 17 September 2021; Accepted 16 October 2021; Published 31 October 2021

Academic Editor: Yulei Wu

Copyright © 2021 Zibo Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the convergence of IT and OT networks, more opportunities can be found to destroy physical processes by cyberattacks. Discovering attack paths plays a vital role in describing possible sequences of exploitation. Automated planning that is an important branch of artificial intelligence (AI) is introduced into the attack graph modeling. However, while adopting the modeling method for large-scale IT and OT networks, it is difficult to meet urgent demands, such as scattered data management, scalability, and automation. To that end, an automatic planning-based attack path discovery approach is proposed in this paper. At first, information of the attacking knowledge and network topology is formally represented in a standardized planning domain definition language (PDDL), integrated into a graph data model. Subsequently, device reachability graph partitioning algorithm is introduced to obtain subgraphs that are small enough and of limited size, which facilitates the discovery of attack paths through the AI planner as soon as possible. In order to further cope with scalability problems, a multithreading manner is used to execute the attack path enumeration for each subgraph. Finally, an automatic workflow with the assistance of a graph database is provided for constructing the PDDL problem file for each subgraph and traversal query in an interactive way. A case study is presented to demonstrate effectiveness of attack path discovery and efficiency with the increase in number of devices.

1. Introduction

Since information technology (IT) was introduced into all walks of life, the threat from hackers and virus attacks have never been got rid of. However, it does not prevent industrial enterprises from adopting the commercial-off-the-shelf software and hardware and the general network connectivity into operational technology (OT) networks, such as industrial control networks [1]. The IT/OT convergence provides attackers more opportunities to launch targeted attacks whose consequences can be disastrous against the real physical world. The industrial control security incidents in the past decade are the best proof that cyberattacks are gradually infiltrating from the IT networks to the OT networks [2].

Apart from the cyberattacks migrated from IT networks, some inherent issues exist in the OT networks, such as design

defects in industrial control network protocols [3] and vulnerabilities of proprietary devices [4]. On account of frequent interactions between IT devices and OT components, there are no clear boundaries between IT and OT partitions in the current industrial environment. In other words, any compromise that occurred on the devices or networks in either IT or OT side has an undesirable impact on the overall safety and security. Therefore, both IT and OT aspects should be taken into consideration simultaneously for cybersecurity analysis in a comprehensive assessment [5].

In general, the security assessment mostly relies on a standalone vulnerability scanning for services or devices in the IT and OT networks. Although more specific vulnerability information can be obtained by scanning, they are difficult to be used to understand means and intents of

sophisticated attacks, let alone to suit for the comprehensive assessment. To that end, the concept of the attack path is presented to describe an alternative sequence of exploitation steps that can lead to a successful attack [6]. Direct, indirect, and subliminal attack paths are concerned by security practitioners and researchers. The process of generating those paths can be treated as that of simulating specific attack behaviors [7].

Focusing on finding all possible attack paths, dependencies of topologies, vulnerabilities, exploitations, and targets are integrated into a graph, called attack graph [8]. Since the attack graph model was first constructed in 1997, quantities of generation methods for it have been widely used in a variety of scenarios to discovery attack paths [6]. Among them, automated planning, a branch of the artificial intelligence (AI), is adopted in the attack graph generation, which transforms finding valid paths into solving problems of a given attack scenario by a planner [9]. It has two obvious characteristics compared with other attack graph generation methods. For one thing, various mature domain-independent planners on the graph plan algorithm can be utilized to accomplish path discovery tasks. For another, a standardized planning domain definition language (PDDL) has favorable data representation with rich-level semantics and the numerical logic deduction to support the planner, which is suitable for modeling changeable attack scenarios by encoding domain knowledge and variable conditions [10, 11].

Nevertheless, when implementing into the IT and OT networks, the planning-based method also suffers from several problems mainly in three aspects: (1) Despite the advantage of the PDDL descriptions in the modeling, it loses some heterogeneous and scattered information with the abstraction of the attacking knowledge and the network topology, which is uncondusive to make sense of attack paths. (2) Most attack graph generation methods are limited by the scalability with the increasing scale of the network topology. Unfortunately, there is no exception for the planning-based method whose bottleneck restrictions exist in parsing the complex PDDL problem file of encoding a complete and larger attack scenario and attack path enumeration to call the planner. (3) Enhancing the automation in each stage is a long-standing topic for the attack graph modeling. Faced with frequent changes in the attack scenarios, a challenge comes from how to reconstruct corresponding PDDL descriptions.

To cope with the problems as mentioned above, we proposed an automatic planning-based attack path discovery approach on the basis of the literature [9]. We aim to extend planning-based method for large-scale attack graph generation. The main contributions of this work are summarized as follows:

- (1) We present a formal data description method for attacking knowledge and network topology. Modeling by PDDL still possesses the advantage in descriptions. The combination with a graph data model does favor to globally understand attack paths with the integration of scattered information in the form of entities and relations.

- (2) We improve the conventional planning-based attack graph generation method for a large-scale network. A device reachability graph partitioning algorithm is introduced to obtain subgraphs that are small enough and of limited size, facilitating the discovery of solution by planner, and reducing the burden of parsing the PDDL files.
- (3) We provide an automatic workflow with the assistance of a graph database. In response to the attack scenarios changing, an automatic construction for the PDDL files is implemented for each subgraph. The graph database with the model makes it possible for subsequent visualization and assessment in an interactive way.

The rest of the paper is organized as follows: A related work is summed up in Section 2. An overview of our proposed approach is given in Section 3. Section 4 provides a formal representation method including the PDDL and a graph data model. In Section 5, 4 algorithms are elaborated to complete a series of tasks on the attack path discovery. A case study demonstrated the function and performance of our proposed approach in Section 6. Finally, we conclude the research in Section 7.

2. Related Work

In this section, we simply review methodologies and techniques to generate various types of attack graphs, such as model checking, deductive reasoning, automated planning, parallel computing, and graph data modeling. We mainly focus on the following two perspectives, namely, the scalability and the formal data representation.

In general, model checking is a technique for determining whether a formal model satisfies a given property or not. By finding counterexamples on attack sequences, paths that breach security attributes are represented in a state attack graph [12]; however, it also confronts exponential state-space problems even used in middle-scale networks. Different from the state attack graph, logical attack graphs are built by deductive reasoning to demonstrate attack steps and their prerequisites for each action [13–16]. A well-known and open-source reasoning framework, called MulVAL (Multihost, Multistage Vulnerability Analysis), is utilized to infer attack paths among attack goals and configuration information [15, 16], which is fit for risk assessment in the large-scale enterprise environment.

Those two kinds of attack graphs belong to the complete graph, probably involving attack paths that cannot reach the attack goal, whereas a minimal attack graph is defined as all attack paths terminated to the specific goal [9]. Planning-based methodologies can be adopted to generate the minimal attack graph [17–20]. A set of domain-independent planners can be used to build attack graphs, such as GraphPlan, FF, Metric-FF, LPG-td, and SGPlan [11]. Nevertheless, it is universal acknowledge that the planning-based methodologies has limitations in scalability. Consequently, a graph reduction method is introduced to simplify the task of searching without sacrificing the quality of attack

paths in [17]. Concentrating on penetration information of a testing goal, a compact planning graph algorithm is proposed in [20] to prune redundant attack path branches for the improvement of the scalability.

Compared to the serial approaches mentioned above, another attack graph generation has a dependence on parallel computing [21–23]. Its core idea is using a partition algorithm to split a large-scale complex network topology and processing each subgraph by multiple agents at the same time, which emphasizes the load balancing to enhance the efficiency of the attack graph generation.

Besides the methodologies and techniques, an appropriate formal data representation of the expert knowledge and the network configuration is important for the attack graph generation as well. It is not only meaningful to describe multisource information in each corresponding modeling approach, but also helpful to promote readability for understanding attack paths. There is a research direction for modifying a generated attack graph. For instance, a process-mining algorithm is employed to extract the chronological order and logical relationship among cyber-attack behaviors, generating an attack graph on massive security alerts [24]. Then, the complex graph is split based on some branches without changing its original structure, which makes it easier to understand. In [25], an ontology is constructed for the MP (multiple prerequisite) graphs that scales nearly linearly as the network size growing. It enables network administrators to make sense of the semantics of attack paths by knowledge inferences. In order to build a simplified attack graph, the concept of abstracted visualizations is implemented by aggregating part of the original attack steps according to an asset [26].

Since the input and output information of attack graphs are manifested as connected data, it can be naturally stored as graph data in the form of nodes and edges. With the recent popularity of the graph database, the graph data model is paid more attention by researchers to represent data, such as the topology and vulnerabilities [27–30]. In addition to in consideration of the data storage form, gaining the valid information among the large-scale and isolated data is still an impending demand for building attack graphs. Graph traversal queries can provide an efficiency and scalable solution for the difficulty. In [28], a cyberattack-oriented graph data model is given around attack paths to capture relationships among entities in the security domain allowing for the division of graph models into interconnected layers, which is convenient for supporting other collaborative assessment tasks.

Inspired by the researches listed in this section, we attempt to solve the problems in three aspects described in the introduction part. Our proposed approach innovatively combines two methodologies that are graph partitioning and graph data modeling to mainly promote automation and scalability for the IT and OT network environment.

3. Proposed Method

Our method aims to find attack paths from IT to OT networks in a rapid and automatic way. We applied the PDDL

for formal representation to model the network topology and attacking knowledge as inputs of an independent AI path planner. To further improve the scalability of attack path enumeration using the planner, a device reachability graph partitioning is introduced before the attack path planning phase. Subsequently, calling the enumeration algorithm in a multithreading manner for each subgraph, all attack paths can be discovered. As a core component of our method, a graph database has advantages in the aspects of the storage and traversal query. It manages entities and maintains relationships in the phases of information gathering, key element extraction, and attack path planning. Moreover, automatically constructing PDDL files is realized by obtaining property fields lying in the entities. Particularly for the large-scale networks, the interactive query plays a vital role in multisource information on attack paths, and the graph database makes it possible for the attack path visualization or assessment in a limited time.

As illustrated in Figure 1, the proposed framework for attack path discovery consists of information gathering, key element extraction, topology analysis, graph data construction, formal representation, attack path planning, and its application. We accomplish attack path discovery tasks of different phases with the generation and exchange of data workflow. The descriptions of the major function modules are provided as follows:

- (i) Topology analysis is as follows: it analyzes basic data from information gathering to parse network reachability among devices, calculating the overall network complexity. According to a preset value of subgraph scale, the network topology in a large scale is split into multiple parts for subsequent attack path planning in a multithreading way.
- (ii) Graph data construction is as follows: the data from information gathering, key element extraction, and topology analysis is stored in the form of graph data. By means of fast traversal query APIs in the graph database, it is possible to obtain essential information for both formal representations using the PDDL and attack path application. Besides, the dependency between vulnerabilities, stemming from attack paths, can be added into graph data relations.
- (iii) Formal representation is as follows: using the results of the graph database queries, two PDDL files on domain and problem can be generated automatically for the AI planner. The domain is a set of actions which focuses on the state transition of attacking, and the problem contains initial states of devices, vulnerabilities, topology, as well as the goal states.
- (iv) Attack path planning: the AI planner generates a shortest attack path based on the specific PDDL files. In order to find all attack paths, an enumeration algorithm is developed for each subgraph, which needs to modify the problem file for replanning paths. When a set of attack paths is

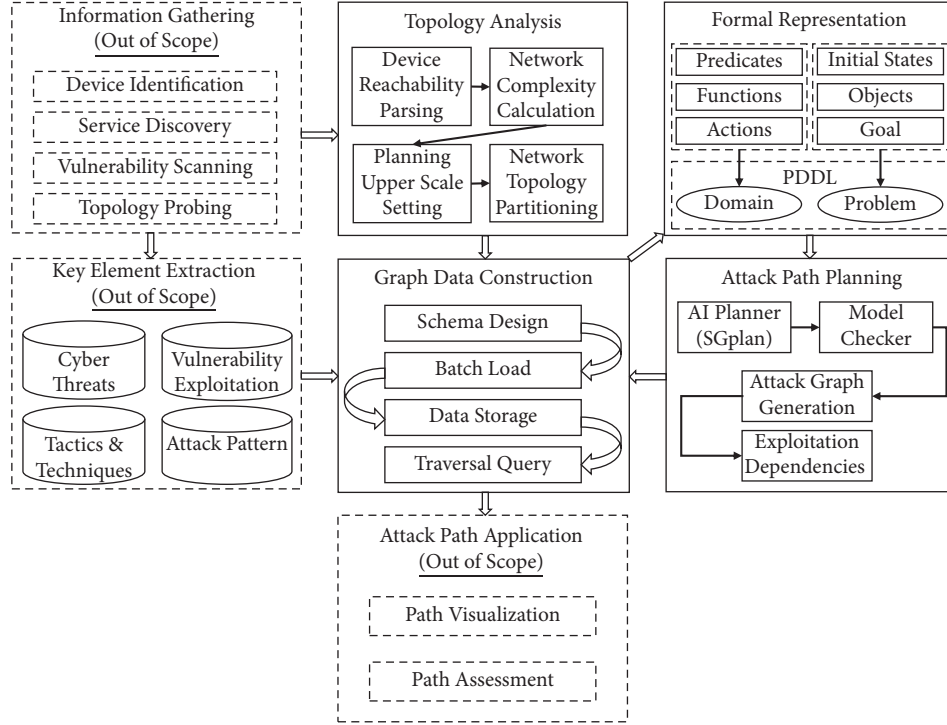


FIGURE 1: Overview of our automatic planning-based attack path discovery method.

available, a global attack graph is generated for further analyzing exploitation dependencies.

In the following sections, we discuss the main parts of the proposed method in detail.

4. Formal Data Representation

In this section, we formally define a network model, a device reachability, an attack path, and an attack graph. After that, the definitions of domain and problem in PDDL, as well as AI planner, are given. Considering the demands of the attack path generation as aforementioned, a graph data model is constructed to represent entities and relationships in a network.

4.1. Preliminaries. Given a formal network model, the first and core step in building an attack graph is determining its reachability. An attack graph is an abstraction of representing all paths that an attacker is able to exploit interdependencies among existing vulnerabilities:

- (i) *Definition 1.* A network model is defined as $N = \langle D, R, V \rangle$, where D is a set of devices D_i ($i = 1, 2, 3 \dots$), $R \subseteq D \times D$ is a set of reachable conditions R_{ij} ($i = 1, 2, 3 \dots, j = 1, 2, 3 \dots$), and V is a set of vulnerabilities V_i ($i = 1, 2, 3 \dots$).
- (ii) *Definition 2.* A device reachability refers to whether some ports on the device can be accessed via TCP or UDP connections from other devices in the network, which can be analyzed from firewall rules and the network topology. Boolean $R_{ij} = 1$ denotes that D_i reaches D_j via an open port.

- (iii) *Definition 3.* An attack path (AP) is a finite acyclic path of a sequence on devices D_i and vulnerabilities V_i , where $\mathbf{AP} = \{(x_1, x_2, \dots, x_n) | x_n \in (D \times V) \cup (V \times D)\}$, $n = 1, 2, 3 \dots$

- (iv) *Definition 4.* An attack graph (AG) is a data structure that represents the intersection of all attack paths, where $\mathbf{AG} = \{AP_i | AP_i \text{ is an attack path in } N\}$, $i = 1, 2, 3 \dots$

4.2. Domain and Problem in PDDL. The PDDL and its variants are often used for encoding domain knowledge. Given that the information has been represented into PDDL files of domain action models and problem statements, there are a variety of classical and heuristic planners accessible to generate attack plans:

- (i) *Definition 5.* A PDDL domain is an abstract description on a series of problems, including requirements, functions, predicates, and available actions with the preconditions and postconditions. It models a variety of attacks which corresponds to vulnerabilities, tactics, and techniques.
- (ii) *Definition 6.* A PDDL problem is a concrete instance of a certain PDDL domain, including objects, initial states with numerical predicates, and a goal state. It models the device reachability, service running on the devices, and attacker abilities, such as privileges, credentials, and even an entry point of a compromised network.
- (iii) *Definition 7.* An AI planner is a search algorithm designed for a specific purpose, finding out a plan

that satisfies a PDDL problem. In this paper, we choose a domain-independent planner, called SGPlan (<https://wah.cse.cuhk.edu.hk/wah/programs/SGPlan/sgplan5.html>), to solve the planning problem in PDDL, which generates a shortest attack path at a time.

Figure 2 depicts our running example that a simple network scenario is modeled by domain and problem using the PDDL and the planner finds out a valid attack path. The simple network scenario contains two devices where they both run services with vulnerabilities. One vulnerability exists in the TIA portal used on engineering workstations (EWS) owing to the improper input validation. The other vulnerability affects the CPU of Siemens S7-300 PLCs where it can be arbitrarily switched to a defect mode. The planning attack path is $\langle \text{Attacker} \rightarrow \text{Improper Input Validation} \rightarrow \text{EWS} \rightarrow \text{Improper Control} \rightarrow \text{PLC} \rangle$.

Note that we expect to express that an attacker may send specially crafted packets to gain privileges on the EWS and then exploit the vulnerability of the PLC CPU to take an improper control of it. However, the output of the planner tends to miss some key information, which is used to describe an attack path. For that reason, we next devise entities and relationships in graph data to make up for the missing semantics.

4.3. Graph Data Model. In our approach, we construct a graph data model to represent around devices, networks, and vulnerabilities. It is made up of a set of nodes and edges, where nodes represent entities and edges represent relationships between entities. There are five entities in this model including the Device, the Vulnerability, the Component, the Domain, and the Tactic & Technique. The properties of each entity are shown in Table 1. Facts corresponding to the entities are taken from the phases of the information gathering and the key element extraction. Particularly, the Component is a general term, referring to services, operating systems, or even the hardware, using type field in the property to distinguish. The term ‘‘Tactic & Technique’’ is derived from ATT & CK for ICS (https://collaborate.mitre.org/attacks/index.php/Main_Page) to describe individual techniques under the tactics on some specific vulnerabilities.

In addition, seven relationships are summarized in Table 2. In the initial stage of the construction, the top five relationships described in the table can be provided among the node of facts. However, the relations between the attack and the exploit are added into the graph data model when attack paths are generated by the proposed enumeration method.

As illustrated in Figure 3, we further expand the semantics of the attack path mentioned in the previous section. Take the PLC in the simple network scenario as an example. In the process control domain, the CPU of the PLC may be affected by a vulnerability so that the PLC switches from run mode to defect mode. To that end, attacker gains privilege on the EWS which connects to the PLC and exploits the vulnerability of an improper control, launching the attack to make the PLC denial of

service (DoS). As a result, the PLC may inhibit response functions (IRFs) to message feedback by sensors or actuators in the field. To some extent, the combination of the graph data model with the original discovery method is a desirable way to enhance the readability of attack paths.

5. Attack Path Discovery Approach

In this paper, an automatic, multithreading, AI planning-based enumeration approach is proposed as the core attack path discovery algorithm. Based on the device reachability defined in Section 4.1, a graph partitioning method is introduced to generate subgraphs within the preset numbers of nodes. Each subgraph is assigned to a thread and then the planner is called separately to complete the attack path enumeration. Using the graph data model mentioned in Section 4.3, files of the domain and problem in PDDL can be combined automatically with the help of traversal query for properties, which follows the principles of PDDL syntax.

Given a pair of the domain and problem for a specific situation, a fixed shortest attack path can be solved. By modifying the problem in PDDL, all attack paths can be generated. For that end, an attack path enumeration needs to be developed. We improved the enumeration algorithm mentioned in [9], where it can adapt to a larger network scale for the attack path discovery. Once all attack paths are obtained, an attack graph can be built for analyzing the relations of the attack and the exploitation, which need to be added for the device nodes and vulnerability nodes in the aforementioned graph data model. In the following sections, we will give implementations for the graph partitioning (Algorithm 1), the PDDL files combination (Algorithm 2), the attack path enumeration (Figure 4), and the multithreading execution (Algorithm 3).

5.1. Device Reachability Graph Partitioning. Reachability determines the accessibility conditions among the services running on the target devices. With an increase of numbers of devices in a large-scale IT and OT networks, finding attack paths is a huge burden in a limited time. As we know, the original planning and enumerating attack paths are also not suitable for the large-scale networks, due to the time-consuming process of parsing large PDDL files and the repeated path traversal. Naturally, it is an urgent demand for the attack path discovery to divide the device reachability graph into small-scale subgraphs. Subsequently, we use a multithreaded method to find the attack path for each subgraph.

The complex graph segmentation algorithm is presented in [24], which is originally used to enhance the readability of an attack graph. The idea of the algorithm is that it searches for the branch nodes to split the whole graph and complete subgraphs based on their structure. However, we simplify the above algorithm to partition a device reachability graph in this paper. The input contains a device reachability graph and the subgraph size. The output is all subgraphs. More details are shown in the pseudocode of Algorithm 1.

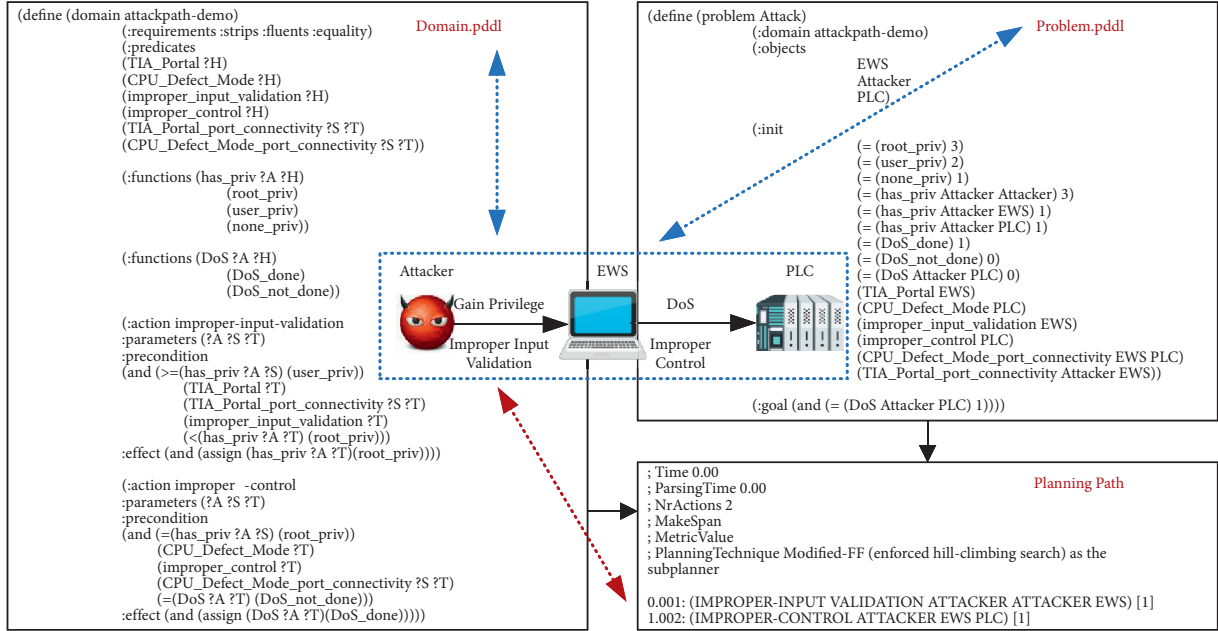


FIGURE 2: Definition of the domain and problem using the PDDL for a simple network scenario.

TABLE 1: Entity information in the graph data model.

| No. | Entity name | Properties |
|-----|--------------------|---|
| 1 | Device | ID, name, type, vendor, version, firmware/OS, fubgraph_ID |
| 2 | Vulnerability | ID, name, type, ATT_vector, CVE_ID, precon, postcon, brief_info |
| 3 | Component | ID, name, type, vendor, version |
| 4 | Domain | ID, name, network, access_rules (service_port) |
| 5 | Tactic & technique | ID, name, brief_info, data_source |

TABLE 2: Relation information in the graph data model.

| No. | Relation name | Descriptions |
|-----|---------------|---|
| 1 | conn | A reachability exists between two devices. |
| 2 | consist | A device belongs to a domain divided by a network. |
| 3 | offer | A component is offered by the devices. |
| 4 | has | A component has a vulnerability. |
| 5 | depend | The tactic and technique depend on a vulnerability. |
| 6 | attack | A device is affected by the vulnerability, leading to an attack. |
| 7 | exploit | A compromised device exploits a vulnerability in another devices. |

5.2. Automatic PDDL Domain and Problem Construction.

As discussed in Section 4.2, it is critical for the planner to define domain and problem files. The domain file encodes predicates and actions. The problem file encodes objects, initial states, and goals. When the planner was run for the different subgraphs, respectively, some pairs of domain and problem files are needed at the same time. As a result, it is necessary to construct these two files in an automatic way, particularly for the situation of large amount subgraphs. To realize the above purpose, we build two templates for domain and problem in PDDL. Combining with the query results from the graph database and the templates, domain and problem files can be generated within a short time, even

if the reachability and device configuration are modified in the previous phases. More details are shown in the pseudocode of Algorithm 2.

5.3. Attack Path Enumeration.

According to a pair of domain and problem PDDL files, the planner provides a solution to find the shortest attack path. Nevertheless, an attack path enumeration strategy is supposed to be developed to look for all attack paths. Referring to the literature [9], a customized algorithm is given by modifying the problem PDDL file to generate new attack paths until an exclusive path set is found. The way of modifying is to automatically

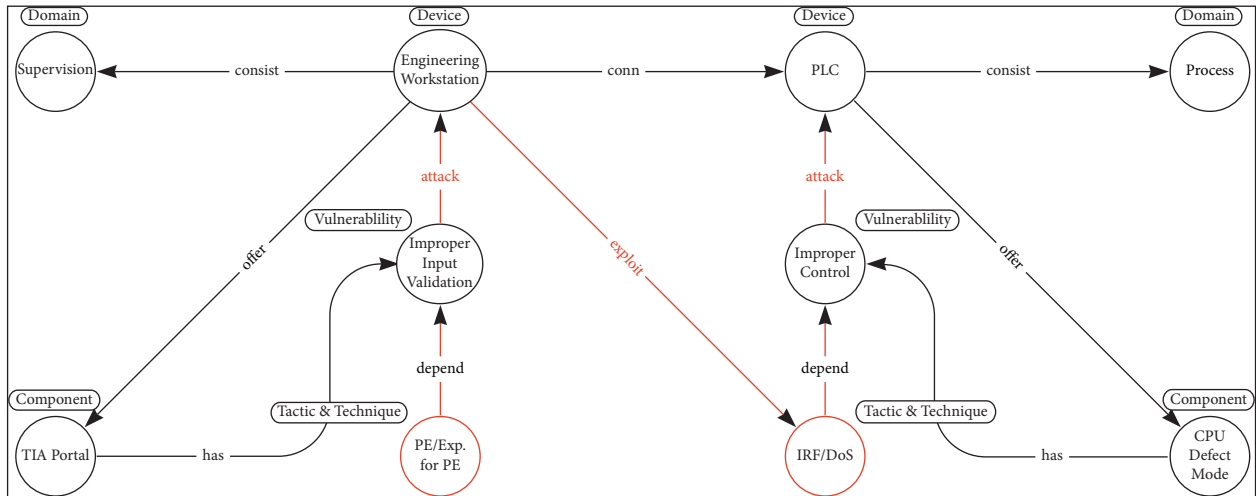


FIGURE 3: Graph data model for a simple network scenario.

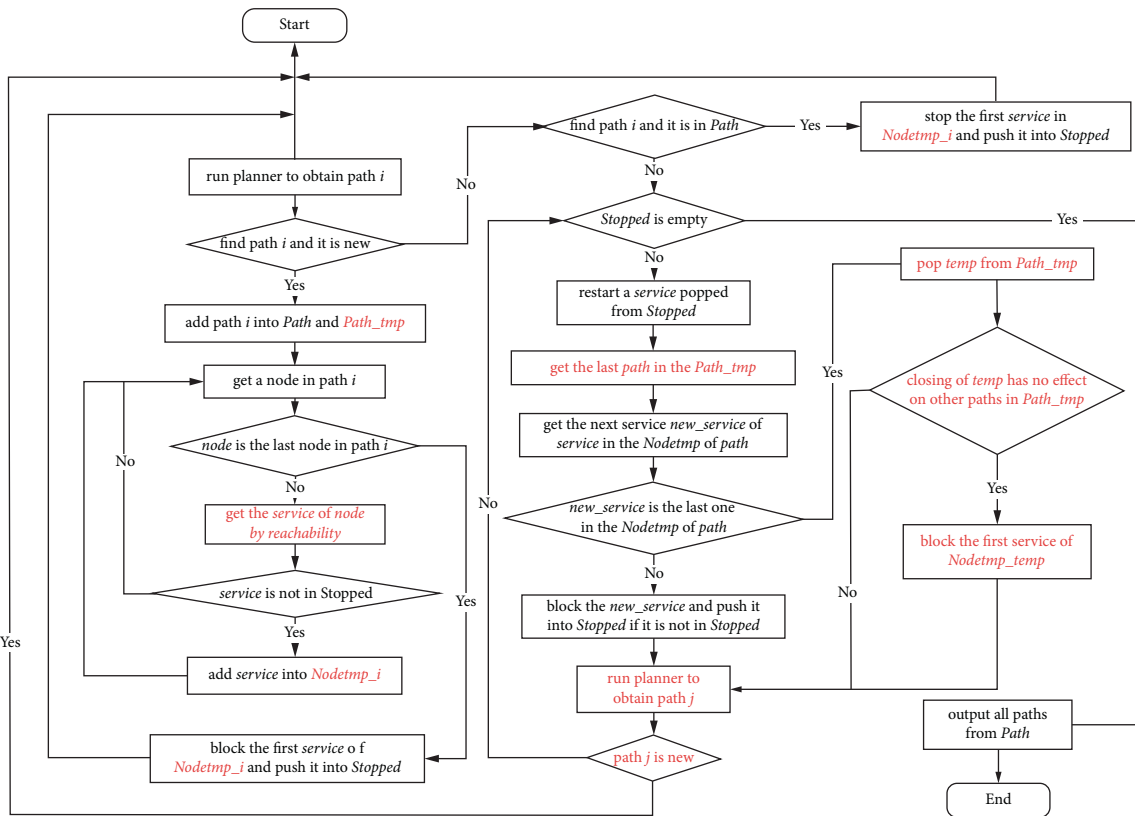


FIGURE 4: Flowchart for the improved attack path enumeration method (the improved parts are marked in red color).

block nodes on the attack path, which is implemented by commenting out some contents encoded in the file that relates to services.

In terms of the customized algorithm, a series of data structures are defined to be used in attack path enumeration. One of them, called *critical nodes*, denotes whether a node is always blocked to generate a new path or not. On the contrary, the other data structure, called *noncritical nodes*, denotes the nodes rely on some critical

nodes, and no new paths are obtained if the nodes are blocked alone. Although these two data structures are applied for indicating how to block nodes, there exist fields of them to be predefined by analyzing the services in a manual way. It is not conducive to enumerate attack path in larger-scale networks. Hence, we remove these two data structures in our improved method. The modification of the attack path enumeration algorithm is shown as a flowchart in Figure 4.

```

Input: device reachability graph  $Gr$ , subgraph size  $subg\_size$ 
Output: all subgraphs
(1) function GENERATE SUBGRAPHS ( $Gr$ ,  $subg\_size$ )
(2)   if nodes_num of  $Gr$  more than  $subg\_size$  then
(3)     push  $Gr$  to  $Qp$ 
(4)     while  $Qp$  is not empty do
(5)       pop a subgraph  $G$  from  $Qp$ 
(6)       find branch nodes  $Ns$  from  $G$ 
(7)       get edges from  $Ns$  and push them into  $Qe$ 
(8)       while  $Qe$  is not empty do
(9)         pop an edge  $qe$  from  $Qe$ 
(10)        find successor subgraph  $Gs$  from edge  $qe$  in  $G$ 
(11)        if nodes in  $Gs$  less than  $subg\_size$  then
(12)          push  $Gs$  into  $Qp$ 
(13)        else
(14)          push  $Gs$  into  $Qo$ 
(15)        output all subgraphs from  $Qo$ 
(16)      else
(17)        output  $Gr$ 
(18)    end function

```

ALGORITHM 1: Device reachability graph partitioning.

```

Input: pddl file template  $domain\_temp$  and  $problem\_temp$ 
         connection object to a graph database  $hg$ ; planning goals
Output: constructed pddl domain and problem files
(1) function GENERATE PDDL FILE ( $domain\_temp$ ,  $problem\_temp$ ,  $hg$ )
(2)   query device nodes, device reachability, vulnerability and component via  $hg$ 
(3)   generate domain file:
(4)     generate predicates of vulnerability, reachability, pre and postconditions
(5)     generate actions of vulnerability from pre- and postconditions of vulnerability
(6)   end
(7)   generate problem file:
(8)     generate objects from device nodes
(9)     generate initially satisfied conditions
(10)    generate goals based on your input
(11)  end
(12)  return generated domain and problem files
(13) end function

```

ALGORITHM 2: Automatic construction of PDDL domain and problem files.

In our improved algorithm, we introduce two new data structures, named *Nodetmp* and *Path_tmp*, respectively, which are used to store the information during the enumeration. The *Nodetmp* is defined to record the services of the nodes in one path, and the *Path_tmp* is defined to record the path, which is not completely processed. The rest of the data structures are in accordance with the definitions in the literature [9].

5.4. Attack Graph Generation. In this part, we integrate the above three algorithms into a multithreading program. Each thread is assigned for one subgraph to call the subprogram of attack path enumeration, and creating and submitting a thread is managed by a thread pool. Particularly, we remove

some irrelevant items for each subgraph from the problem PDDL file to reduce file parsing time. It greatly shortens the time to generate an attack graph in a large scale, and the result is discussed in Section 6.

The input is the number of threads. The output is a complete attack graph, merged by a set of attack graphs of subgraphs. After generating an attack graph, the exploit and attack edges among the nodes of devices and vulnerabilities are added in a graph database as mentioned in Table 2. More details are shown in the pseudocode of Algorithm 3.

6. Case Study

In this part, the proposed automatic planning-based attack path discovery approach is evaluated. At first, an

Input: number of threads `thread_num`
Output: attack graph `AG`; adding exploit and attack edges in a graph database

- (1) create an empty attack graph `AG`
- (2) get `domain.pddl` and `problem.pddl` via GENERATE PDDL FILE (`domain_temp`, `problem_temp`, `hg`)
- (3) get all subgraphs from GENERATE SUBGRAPHS (`G`, `subg_size`)
- (4) create threads pool `threads_pool` and set maxim workers corresponding to `thread_num`
- (5) **foreach** `subgraph` in `subgraphs` **do**
- (6) modify `problem.pddl` and `domain.pddl` based on `subgraph`
- (7) create a thread and bind it to enumerate attack paths using a planner
- (8) submit this thread to `threads_pool`
- (9) **while** `True` **do**
- (10) check the status of threads in `threads_pool`
- (11) **if** all tasks in `threads_pool` have done **do**
- (12) break
- (13) generate subag from paths returned from each thread and merge them into `AG`
- (14) get `ag_edges` from `AG`
- (15) create attack and exploit edges in a graph database according to `ag_edges`

ALGORITHM 3: Attack graph generation in a multithreading manner.

experimental setup is introduced by a hypothetical network topology from IT to OT networks in Section 6.1. Then, attack paths are illustrated, and the corresponding data is stored in the form of graph data in Section 6.2. Finally, we discuss the performance in terms of device reachability graph partitioning, attack path planning, and its enumeration in Section 6.3, which allows us to examine the scalability with the increasing devices in the IT and OT networks.

6.1. Experimental Setup. As shown in Figure 5, a hypothetical network topology is constructed whose structure stems from the real-world practice, but its size is simplified. It is separated into six subnets according to the different functions. The Enterprise Control Network is a corporate network with respect to the product lifecycle management, the resource planning, the business planning, and so on. The Perimeter Network manages servers to provide information for users on the Enterprise Control Network via a variety of services, such as web and mail. The Manufacturing Operations Network is a bridge of information exchanges between control systems and enterprise resources planning systems to support the top-down decision-making. The Process Control Network is used to transmit instructions and data between control and measurement units and Supervisory Control and Data Acquisition (SCADA) devices. The Automatic Control Network is connected to numbers of HMIs and PLCs in fields, which are responsible for logic and control computing tasks to manipulate and regulate sensors or actuators in the Physical Control Network. Among them, the IT networks are made up of the Enterprise Control Network and the Perimeter Network, and other subnets belong to the OT networks [1].

The hypothetical network topology contains twenty heterogeneous devices so as to introduce many services and vulnerabilities, as shown in Tables 3 and 4. The vulnerability information is extracted from descriptions of the NVD (<https://nvd.nist.gov/>) and the ATT & CK ICS (<https://collaborate.mitre.org/attackics/index.php/>

`Main_Page`). Considering the device type and applied technologies, we divide the whole network topology into two parts, namely, Zones A and B. In Zone A, more devices adopt the commercial-off-the-shelf software and hardware, where more vulnerabilities may be exploited for the purpose of lateral movements to the OT networks. Due to factors, such as time and continuity, less security protection devices are deployed in Zone B. Once some devices are compromised in that zone, sophisticated attackers can take multiple measures to launch control process-oriented attacks to affect physical operations. In order to highlight dependencies of vulnerabilities, we define access control rules among services in detail, as shown in Table 5.

6.2. Attack Path Discovery. Based on the model constructed in Section 4.3, we store the experimental data in the form of the graph data. Utilized in this paper, the graph database, HugeGraph (<https://hugegraph.github.io/hugegraph-doc/>), is efficient, universal, and open source. It is fully compatible with Gremlin query language and implements with the Apache TinkerPop3 framework. The stored graph data is the basis of subsequent automatic generation of the PDDL files and the final attack graph generation. To demonstrate the feasibility of our proposed method, we give the results in a reversed order as it is described in Section 3. In this experiment, we define the entry point of the attack as the Manger PC, and its compromised goal is the PLC2 that is a slave station connected to a set of physical equipment.

Figure 6 is a complete attack graph for the experimental environment, which is output by the Graphviz (<http://www.graphviz.org/>) library of the Python. There are 189 attack paths that can reach the attack goal. We separately show the attack paths for Zones A and B in Tables 6 and 7, because of display convenience. Obviously, it is difficult to find a node like the Historian node (Dev12) of the hypothetical network topology, which can be viewed as a cut point in the Graph Theory to partition a network topology.

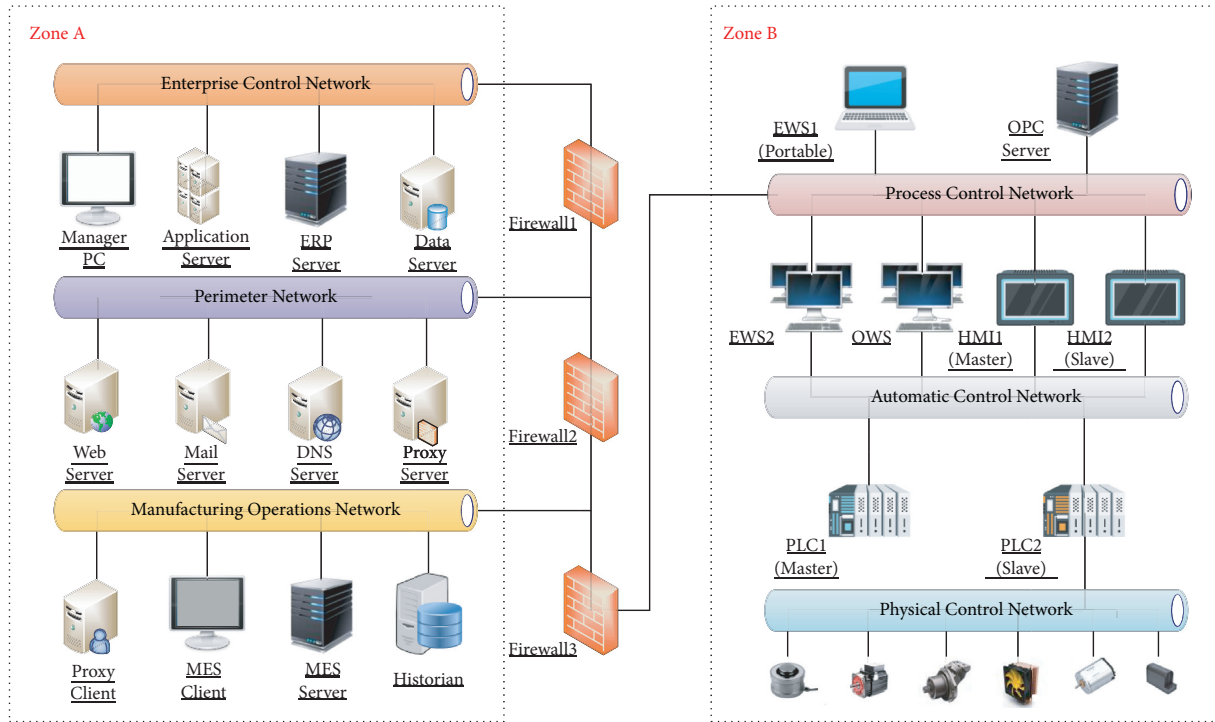


FIGURE 5: A hypothetical network topology from IT to OT networks.

TABLE 3: Device information in the experimental environment.

| Dev.ID | Device name | Port | Vulnerability | Affected component |
|--------|--------------------|------------|--------------------------------------|---|
| Dev1 | Manager PC | — | LNK remote code execution | Icon of the shortcut in windows platform |
| Dev2 | Application server | — | Credentials leak | Connected device login |
| Dev3 | ERP server | 22 | OS command injection | OpenSSH(SCP) |
| Dev4 | Data server | 3389 | BITS improper privilege management | Windows background intelligent transfer service |
| Dev5 | Data server | 3306 | Permissions and access controls | MySQL |
| Dev6 | Web server | 80 | Memory buffer overflow | Internet information services |
| Dev7 | Mail server | 80 | Improper access control | Roundcube |
| Dev8 | DNS server | 53 | DNS server remote code execution | Windows DNS server |
| Dev9 | Proxy server | 8090, 4900 | Path traversal | Lanproxy server |
| Dev10 | Proxy client | 12000 | Plaintext credential | Lanproxy client |
| Dev11 | MES client | 445 | SMBv3 remote code execution | Microsoft server message block protocol |
| Dev12 | MES server | — | Credentials leak | Connected device login |
| Dev13 | MES server | 22 | Kernel improper privilege management | Linux kernel |
| Dev14 | Historian | 80 | SQL server remote code execution | Microsoft SQL server reporting services |
| Dev15 | EWS1 | 445, 139 | Code injection | MSRPC over SMB |
| Dev16 | EWS2 | 3389 | Brute force | Remote desktop services |
| Dev17 | OWS | 445 | SMB remote code execution | Microsoft server message block protocol |
| Dev18 | OPC server | 8080 | Unrestricted upload of file | Apache tomcat |
| Dev19 | HMI1 (master) | 2308, 1033 | Modify configuration project | HMI configuration project in WinCC |
| Dev20 | HMI2 (slave) | 2308, 1034 | Modify configuration project | HMI Configuration project in WinCC |
| Dev21 | HMI1 (master) | 2308, 1033 | Fake MAC address | HMI and PLC communication |
| Dev22 | HMI2 (slave) | 2308, 1034 | Fake MAC address | HMI and PLC communication |
| Dev23 | PLC1 (master) | 102 | Modify parameters/modes | PLC automatic operation/states |
| Dev24 | PLC2 (slave) | 102, 502 | Modify control logic | PLC program project in TIA portal |
| Dev25 | PLC1 (master) | 102 | Plaintext control command | Legacy S7Comm protocol |
| Dev26 | PLC2 (slave) | 102, 502 | Fake MAC address | HMI and PLC communication |
| Dev27 | PLC1 (master) | 102 | Modify parameters/modes | PLC automatic operation/states |
| Dev28 | PLC2 (slave) | 102, 502 | Modify control logic | PLC program project in TIA portal |
| Dev29 | PLC1 (master) | 102 | Plaintext control command | Modbus protocol |
| Dev30 | PLC2 (slave) | 102, 502 | Uncontrolled resource consumption | Protocol common used port |
| Dev31 | PLC1 (master) | 102 | Improper control | CPU defect mode |

TABLE 4: Vulnerability information in the experimental environment.

| Vul.ID | Vulnerability | ATT_Vector | Precondition | Postcondition | Tactics/techniques |
|--------|--------------------------------------|------------|--|-----------------------------|--|
| Vul1 | LNK remote code execution | Local | USB access/crafted LNK files | Execute any code | IA/replication through removable media |
| Vul2 | Credentials leak | Local | Plaintext record file | Credential acquisition | LM/valid accounts |
| Vul3 | OS command injection | Remote | SSH password | Execute any code | LM/remote services |
| Vul4 | BITS improper privilege management | Local | USER login | Administrator (windows) | PE/exploitation for privilege escalation |
| Vul5 | Permissions and access controls | Remote | USER login | Root (linux) | PE/exploitation for privilege escalation |
| Vul6 | Memory buffer overflow | Remote | Crafted URL | Execute any code | IA/exploit public-facing application |
| Vul7 | Improper access control | Remote | Crafted e-mail messages | Execute any code | IA/exploit public-facing application |
| Vul8 | DNS server remote code execution | Remote | Malicious requests | Execute any code | IA/exploit public-facing application |
| Vul9 | Path traversal | Remote | Port scan | Credential acquisition | CA/exploitation for credential access |
| Vul10 | Plaintext credentials | Remote | Credential | Login | LM/valid accounts |
| Vul11 | SMBv3 remote code execution | Remote | USER | Execute any code | LM/exploitation of remote services |
| Vul12 | Kernel improper privilege management | Local | USER login | Root (linux) | PE/exploitation for privilege escalation |
| Vul13 | SQL server remote code execution | Remote | Incorrect page request | Execute any code | IA/exploit public-facing application |
| Vul14 | Code injection | Remote | Crafted RPC request | Execute any code | LM/exploitation of remote services |
| Vul15 | Brute force | Remote | Credential | Login | LM/valid accounts |
| Vul16 | SMB remote code execution | Remote | USER | Execute any code | LM/exploitation of remote services |
| Vul17 | Unrestricted upload of file | Remote | JSP file/HTTP request | Execute any code | IA/exploit public-facing application |
| Vul18 | Modify Configuration project | Remote | Malicious Configuration project | Impair HMI control function | P/modify program |
| Vul19 | Modify control logic | Remote | Malicious control logic | PLC denial of service | P/modify program |
| Vul20 | Modify parameters/Modes | Remote | Malicious operations | PLC denial of service | IPC/modify parameter |
| Vul21 | Plaintext control command | Remote | Crafted control command | PLC denial of service | IPC/unauthorized command message |
| Vul22 | Uncontrolled resource consumption | Remote | High volume of requests | PLC denial of service | IRF/denial of service |
| Vul23 | Fake MAC address | Remote | Scan devices/traffic forward/ Modify data | PLC denial of service | C/man in the middle |
| Vul24 | Improper control | Remote | Crafted packets | PLC denial of service | E/change operating mode |

The paths in the attack graph are a mix of the Device nodes and the Vulnerability nodes, which represents that an attacker can reach a device and exploit vulnerability. As discussed in Section 3, attack graph generated by our proposed approach is to analyze exploitation dependencies, and the corresponding edges are added among nodes in the graph database. Afterward, it is convenient to find attack paths arbitrarily with a fixed entry point and an attacking goal by the Gremlin query. In that way, attack paths in Table 6 are listed from the Attacker to the Historian device. Similarly, attack paths in Table 7 are listed from the Historian device to the PLC2.

6.3. Performance Evaluation. Performance evaluation tests of our proposed approach are carried out in the following environments. The domain and problem of the hypothetical

network topology are described in the PDDL (Version 2.2) (<https://planning.wiki/ref/pddl22>). The attack path planning is executed on the SGPlan (Version 5.2.2). All programs on the four algorithms mentioned in Section 5 are implemented in Python (Version 3.5.2), running on a Linux server with the Intel Xeon Silver 4110 CPU at 2.1 GHz and 125 GB RAM. HugeGraph (Server Version 0.11.2) runs standalone in a Docker (Version 19.03.12) container.

Initially, we implement the method introduced in the literature [9] to validate its scalability. Assuming that a one-to-one correspondence exists between the number of vulnerabilities and the number of devices, changes in the network topology are only reflected in the complexity of the problem file in PDDL that has a great impact on the planning. The results are shown in Table 8. Column 2 on the left concerns the size of each test network topology, while the remaining columns, respectively, focus on the performance

TABLE 5: Device access control rules in the experimental environment.

| Dom.ID | Domain name | Source device | Destination devices |
|--------|-------------|--------------------|--|
| Dom1 | Internet | Manager PC | (Application server, 22) (ERP server, 3389) (web server, 80) (proxy server, 8090&4900) |
| | | Application server | (Data server, 3306) |
| | | ERP server | (Data server, 3306) |
| | | Data server | (Mail server, 80) (DNS server, 53) |
| Dom2 | DMZ | Web server | (Historian, 80) |
| | | Mail server | (Proxy server, 8090 & 4900) |
| | | DNS server | (Proxy server, 8090 & 4900) |
| | | Proxy server | (Proxy client, 1200) |
| Dom3 | Scheduling | Proxy client | (Historian, 80) (MES client, 445) |
| | | MES client | (MES server, 22) |
| | | MES server | (Historian, 80) |
| | | Historian | (OPC server, 8080) |
| Dom4 | Supervision | OPC server | (EWS1, 445 & 139) (EWS2, 3389) (OWS, 445) |
| | | EWS1 | (HMI1, 2308 & 1033) (HMI2, 2308, & 1033) |
| | | EWS2 | (PLC1, 102) (PLC2, 102 & 502) |
| | | OWS | (HMI2, 2308 & 1033) (PLC1, 102) (PLC2, 102 & 502) |
| Dom5 | Process | HMI1 | (PLC1, 102) |
| | | HMI2 | (PLC2, 102) |
| | | PLC1 | (PLC2, 502) |

surrounding the CPU time and memory, including the total planning time in the SGPlan, the parsing time for PDDL files, the enumeration time for all attack paths, and the memory consumption. With the increase of devices (nodes) in the network topology, it is clear that both the running time and the memory consumption grow exponentially. The SGPlan provides the planning time and the parsing time for PDDL files as outputs, and the planning time involves the parsing time and pure solution time for each valid attack path. By summing the planning time and the parsing time, we find that the parsing time accounting for more than 90% dominates in the planning time with the complexity of the problem file in the PDDL growing. Moreover, it is worth noting that the enumeration time is even more unsatisfactory for a small network topology, taking nearly 2 hours to generate all attack paths for a network topology with a size of 41 devices.

To overcome those two shortcomings in the attack path enumeration method using the planner while applying in the large-scale networks, we introduce the graph partition algorithm into our proposed approach. There is a key parameter, namely, the subgraph size `subg_size`, which is set to 10 in Algorithm 1. Performance results of each stage are shown in Table 9. From row 2 to 4 in the table, the results are better than those shown in Table 8 in the case of 21 topology nodes. The reason why the time-consuming process of planning and parsing drastically decreases is that partitioning algorithm on the basis of branch points reduces the complexity of each problem file in PDDL, which helps make it easier for the planner to parse the file and solve a solution for each subgraph. In addition, we further provide running time in the graph partitioning (row 6) and operations with the HugeGraph, such as importing and traversal query (rows 7 and 8), and they take a small proportion in the running time of our proposed approach. The remaining rows list the

information on the attack graph of the hypothetical network topology in Section 6.1.

Finally, we validate that each stage of our proposed approach is suitable for a large-scale network by the experiment. The hypothetical network topology with increasing size of devices and complexity are considered and discussed. It simply achieves that goal by integrally replicating several times those devices contained in the topology to build scenarios of different network sizes, the number of devices ranging from 100 to 1000. Experimental results are shown from Figures 7 to 10. In Figure 7, running time and memory usage of device reachability graph partitioning are shown, considering different numbers of devices. It is observed that the running time is less than 0.2 seconds, even though topological scale reaches more than 1000 devices. Figures 8 and 9, respectively, show the comparison between multithreading and single-threading modes in running time and the memory usage. We set the thread number of the threading pool in Algorithm 3 as 20. Apparently, the time consumption in the multithreading way is less than that in the single-threading way. Simultaneously, its growth ratio is also slower with the increasing of devices. But the cost of our proposed approach has higher memory usage than that in the single-threading way, which is almost linear growth. The overhead of multithreading exists in parsing the problem files and enumerating attack paths. Hence, we remove the irrelevant content encoded in the problem files based on the devices of each subgraph to reduce time of parsing and planning in each thread. It makes it possible to avoid blocking or restarting invalid services in attack path enumeration algorithm as well. What is more, the efficiency of operations, such as importing data and traversal query, determines the process of automatic constructing PDDL files and searching attack paths. Figure 10 shows the running time of the two key operations of the HugeGraph in Algorithms 2 and 3, considering different numbers of devices.

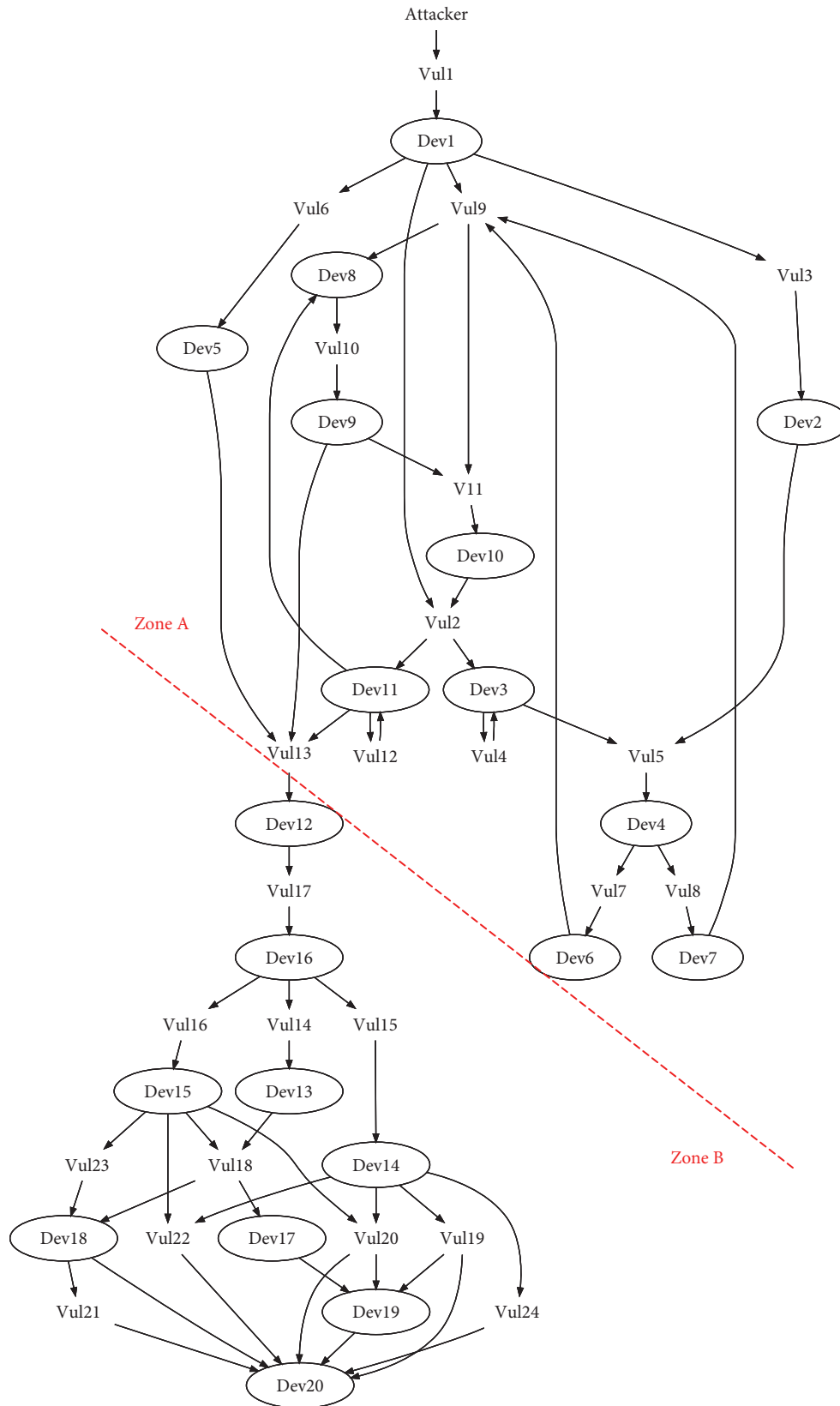


FIGURE 6: Attack graph of the experimental environment.

TABLE 6: Attack paths in Zone A.

| No. | Attack paths |
|-----|--|
| 1 | Attacker → Vul1 → Dev1 → Vul6 → Dev5 → Vul13 → Dev12 |
| 2 | Attacker → Vul1 → Dev1 → Vul9 → Dev8 → Vul10 → Dev9 → Vul13 → Dev12 |
| 3 | Attacker → Vul3 → Dev2 → Vul5 → Dev4 → Vul7 → Dev6 → Vul9 → Dev8 → Vul10 → Dev9 → Vul13 → Dev12 |
| 4 | Attacker → Vul1 → Dev1 → Vul3 → Dev2 → Vul5 → Dev4 → Vul8 → Dev7 → Vul9 → Dev8 → Vul10 → Dev9 → Vul13 → Dev12 |
| 5 | Attacker → Vul1 → Dev1 → Vul2 → Dev3 → Vul4 → Dev3 → Vul5 → Dev4 → Vul7 → Dev6 → Vul9 → Dev8 → Vul10 → Dev9 → Vul13 → Dev12 |
| 6 | Attacker → Vul1 → Dev1 → Vul2 → Dev3 → Vul4 → Dev3 → Vul5 → Dev4 → Vul8 → Dev7 → Vul9 → Dev8 → Vul10 → Dev9 → Vul13 → Dev12 |
| 7 | Attacker → Vul1 → Dev1 → Vul9 → Dev8 → Vul10 → Dev9 → V11 → Dev10 → Vul2 → Dev11 → Vul12 → Dev11 → Vul13 → Dev12 |
| 8 | Attacker → Vul1 → Dev1 → Vul3 → Dev2 → Vul5 → Dev4 → Vul7 → Dev6 → Vul9 → Dev8 → Vul10 → Dev9 → V11 → Dev10 → Vul2 → Dev11 → Vul12 → Dev11 → Vul13 → Dev12 |
| 9 | Attacker → Vul1 → Dev1 → Vul3 → Dev2 → Vul5 → Dev4 → Vul8 → Dev7 → Vul9 → V11 → Dev10 → Vul2 → Dev11 → Vul12 → Dev11 → Vul13 → Dev12 |
| 10 | Attacker → Vul1 → Dev1 → Vul2 → Dev3 → Vul4 → Dev3 → Vul5 → Dev4 → Vul7 → Dev6 → Vul9 → Dev8 → Vul10 → Dev9 → V11 → Dev10 → Vul2 → Dev11 → Vul12 → Dev11 → Vul13 → Dev12 |
| 11 | Attacker → Vul1 → Dev1 → Vul2 → Dev3 → Vul4 → Dev3 → Vul5 → Dev4 → Vul8 → Dev7 → Vul9 → Dev8 → Vul10 → Dev9 → V11 → Dev10 → Vul2 → Dev11 → Vul12 → Dev11 → Vul13 → Dev12 |

TABLE 7: Attack paths in Zone B.

| No. | Attack paths |
|-----|---|
| 1 | Dev12→Vul17→Dev16→Vul14→Dev13→Vul18→Dev18→Dev20 |
| 2 | Dev12→Vul17→Dev16→Vul14→Dev13→Vul18→Dev17→Dev19→Dev20 |
| 3 | Dev12→Vul17→Dev16→Vul15→Dev14→Vul19→Dev20 |
| 4 | Dev12→Vul17→Dev16→Vul15→Dev14→Vul19→Dev19→Dev20 |
| 5 | Dev12→Vul17→Dev16→Vul15→Dev14→Vul22→Dev20 |
| 6 | Dev12→Vul17→Dev16→Vul15→Dev14→Vul24→Dev20 |
| 7 | Dev12→Vul17→Dev16→Vul15→Dev14→Vul20→Dev20 |
| 8 | Dev12→Vul17→Dev16→Vul15→Dev14→Vul20→Dev19→Dev20 |
| 9 | Dev12→Vul17→Dev16→Vul16→Dev15→Vul20→Dev20 |
| 10 | Dev12→Vul17→Dev16→Vul16→Dev15→Vul20→Dev19→Dev20 |
| 11 | Dev12→Vul17→Dev16→Vul16→Dev15→Vul22→Dev20 |
| 12 | Dev12→Vul17→Dev16→Vul16→Dev15→Vul18→Dev18→Dev20 |
| 13 | Dev12→Vul17→Dev16→Vul16→Dev15→Vul23→Dev18→Vul21→Dev20 |

TABLE 8: Performances of attack path enumeration in [9].

| No. | Topo_nodes | Plan_time (s) | Parse_time (s) | Enum_time (s) | Memory (KB) |
|-----|------------|---------------|----------------|---------------|-------------|
| 1 | 11 | 2.23 | 1.66 | 14.72 | 39256 |
| 2 | 21 | 44 | 40.96 | 184.49 | 164360 |
| 3 | 31 | 363.88 | 354.19 | 1398.11 | 644428 |
| 4 | 41 | 1954.15 | 1928.19 | 7319.67 | 1857944 |
| 5 | 51 | 7033.4 | 6973.02 | 26486.55 | 4524044 |

TABLE 9: Performances of our proposed method.

| No. | Metrics | Statistics |
|-----|----------------------|------------|
| 1 | Topo_nodes | 21 |
| 2 | Plan_time (s) | 0.11 |
| 3 | Parse_time (s) | 0.1 |
| 4 | Enum_time (s) | 53.94 |
| 5 | Memory (KB) | 19292 |
| 6 | partition_time (s) | 0.019 |
| 7 | import_time (s) | 0.27 |
| 8 | transversal_time (s) | 0.007 |
| 9 | ag_nodes | 43 |
| 10 | ag_edges | 63 |
| 11 | att_paths | 189 |

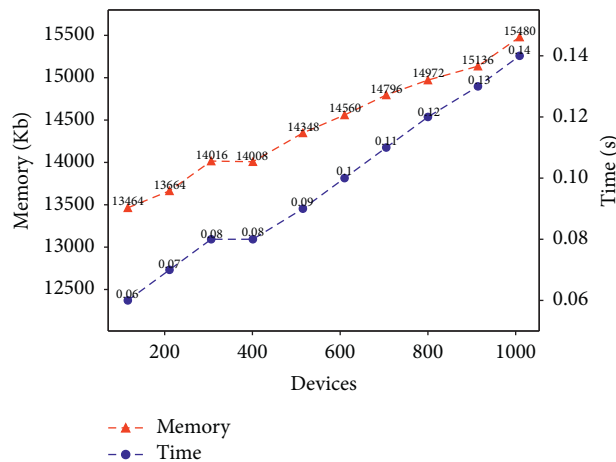


FIGURE 7: Running time and memory usage considering different numbers of devices in Algorithm 1.

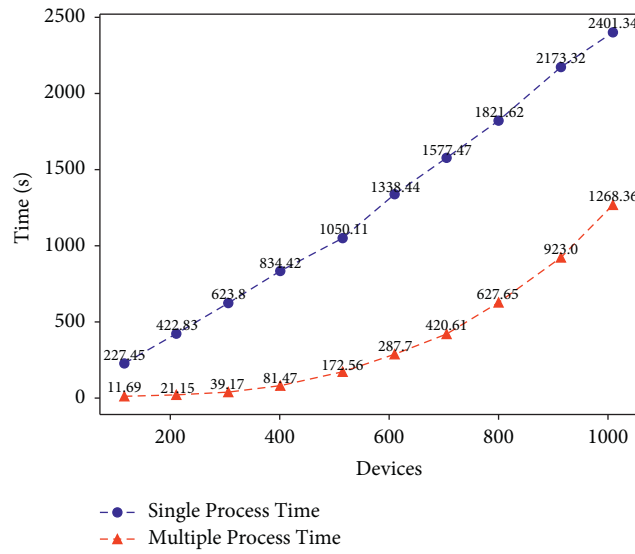


FIGURE 8: Running time comparison of the single and the proposed multiple threading method considering different numbers of devices.

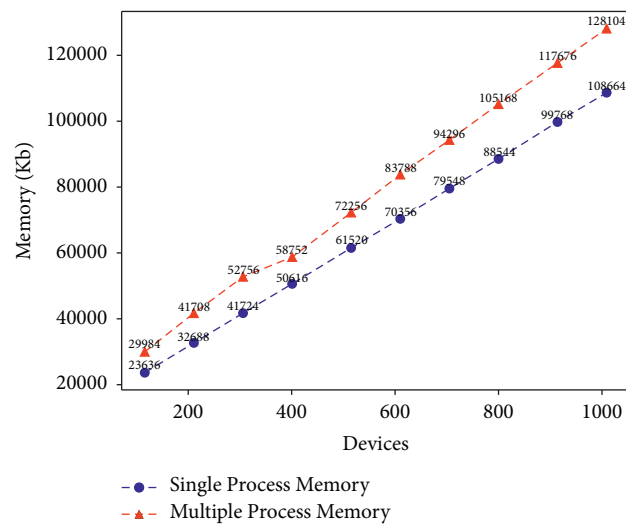


FIGURE 9: Memory usage comparison of the single and the proposed multiple threading method considering different numbers of devices.

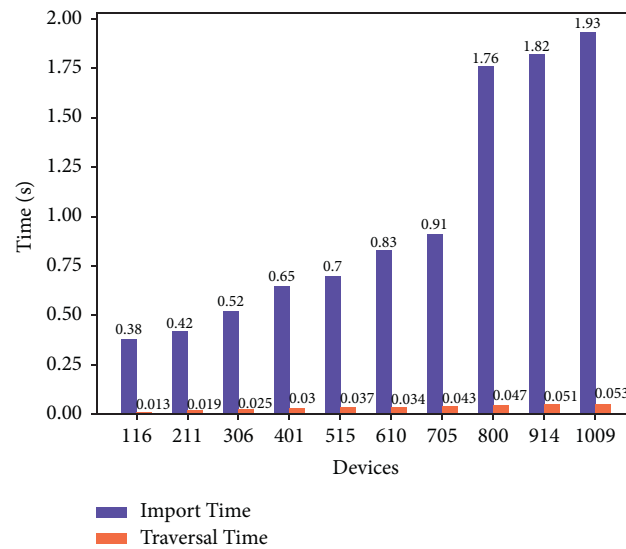


FIGURE 10: Running time of importing and traversal query of the HugeGraph considering different numbers of devices.

7. Conclusion

There is an overwhelming trend that IT and OT networks appear to be deeply integrated into the current industry, whereas their existing security issues still cannot be ignored. Concentrating on attack paths, in contrast with a standalone vulnerability scanning, security assessment takes consideration of dependencies among devices, vulnerabilities, and networks as a whole. Focusing on all attack paths terminated to the specific goal, we present an automatic planning-based approach for the attack graph generation. The conventional planning-based attack path discovery approach is improved with the graph data management, topology partitioning, and the parallel execution, adapted to the large-scale IT and OT networks. The formal data representation adopts the PDDL for describing attack scenarios, which still possesses the advantages in the modeling. Meanwhile, multisource and scattered data is managed by a graph database, providing opportunities for users to query attack paths and corresponding information.

Experimental results indicate that our proposed approach manifests improvements in automation and scalability compared with the conventional planning-based method. Device reachability graph partitioning algorithm helps to reduce the time consuming of parsing the PDDL problem file and planning a single attack path. Calling the attack path enumeration in a multithreading manner has more desirable performance with the number of devices growing. Using the graph database like HugeGraph guarantees the efficiency of importing data and traversal query, which does favor to complete tasks, such as automatic construction of the PDDL files and search attack paths.

In the future work, we attempt to utilize variety of domain-independent AI planners to discovery attack paths, but it is not limited to finding the shortest path in each iteration. The research direction will shift to quantitative security assessment to analyze the attack paths, integrating with probabilities of critical nodes. To further predict attack behaviors, we introduce logical reasoning and the uncertainty theory into the graph data model. Additionally, visual optimization of attack paths is worthy to implement for large-scale IT and OT networks.

Data Availability

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

Conflicts of Interest

All the authors hereby declare no conflicts of interest.

Acknowledgments

This research was funded by the National Key Research and Development Program of China (no. 2018YFB2004200).

References

- [1] R. Paes, D. C. Mazur, B. K. Venne, and J. Ostrzenski, "A guide to securing industrial control networks: integrating IT and OT systems," *IEEE Industry Applications Magazine*, vol. 26, no. 2, pp. 47–53, 2019.

- [2] T. Alladi, V. Chamola, and S. Zeadally, "Industrial control systems: cyberattack trends and countermeasures," *Computer Communications*, vol. 155, pp. 1–8, 2020.
- [3] A. Volkova, M. Niedermeier, R. Basmadjian, and H. D. Meer, "Security challenges in control network protocols: a survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 619–639, 2018.
- [4] R. J. Thomas and T. Chothia, "Learning from vulnerabilities—categorising, understanding and detecting weaknesses in industrial control systems," *Computers & Security*, pp. 100–116, 2020.
- [5] D. Cerotti, D. Codetta-Raiteri, G. Dondossola et al., "A bayesian network approach for the interpretation of cyber attacks to power systems," in *Proceedings of the Italian Conference on CyberSecurity*, Pisa, Italy, February 2019.
- [6] H. S. Lallie, K. Debattista, and J. Bal, "A review of attack graph and attack tree visual syntax in cyber security," *Computer Science Review*, vol. 35, Article ID 100219, 2020.
- [7] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, "A survey of iot-enabled cyberattacks: assessing attack paths to critical infrastructures and services," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3453–3495, 2018.
- [8] J. Zeng, I. Wu, I. Yanyu Chen, I. Rui Zeng, and I. Chengrong Wu, "Survey of attack graph analysis methods from the perspective of data and knowledge processing," *Security and Communication Networks*, vol. 2019, Article ID 2031063, 16 pages, 2019.
- [9] N. Ghosh and S. K. Ghosh, "A planner-based approach to generate and analyze minimal attack graph," *Applied Intelligence*, vol. 36, no. 2, pp. 369–390, 2012.
- [10] S. Khan and S. Parkinson, "Review into state of the art of vulnerability assessment using artificial intelligence," *Guide to Vulnerability Analysis for Computer Networks and Systems*, Springer, Heidelberg, Germany, pp. 3–32, 2018.
- [11] D. R. McKinnel, T. Dargahi, A. Dehghantanha, and K.-K. R. Choo, "A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment," *Computers & Electrical Engineering*, vol. 75, pp. 175–188, 2019.
- [12] T. Alaa, "A2G2V: automatic attack graph generation and visualization and its applications to computer and SCADA networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, pp. 3488–3498, 2019.
- [13] M. Cheminod, L. Durante, L. Seno, and A. Valenzano, "Detection of attacks based on known vulnerabilities in industrial networked systems," *Journal of Information Security and Applications*, vol. 34, pp. 153–165, 2017.
- [14] S. Wu, Y. Zhang, and W. Cao, "Network security assessment using a semantic reasoning and graph based approach," *Computers & Electrical Engineering*, vol. 64, pp. 96–109, 2017.
- [15] M. Inokuchi, Y. Ohta, S. Kinoshita et al., "Design procedure of knowledge base for practical attack graph generation," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Auckland New Zealand, July 2019.
- [16] O. Stan, R. Bitton, M. Ezrets et al., "Extending attack graphs to represent cyber-attacks in communication protocols and modern IT networks," *IEEE Transactions on Dependable and Secure Computing*, 2020.
- [17] T. Gonda, R. Puzis, and B. Shapira, "Scalable attack path finding for increased security," in *Proceedings of the*

- International Conference on Cyber Security Cryptography and Machine Learning*, July 2017.
- [18] B. Bezawada, I. Ray, and K. Tiwary, "AGBuilder: an AI tool for automated attack graph building, analysis, and refinement," in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, July 2019.
 - [19] R. Mirsky, Y. Shalom, A. Majadly, Y. Gal, R. Puzis, and A. Felner, "New goal recognition algorithms using attack graphs," in *Proceedings of the International Symposium on Cyber Security Cryptography and Machine Learning*, June 2019.
 - [20] Z. Yichao, Z. Tianyang, G. Xiaoyue, and W. Qingxian, "An improved attack path discovery algorithm through compact graph planning," *IEEE Access*, vol. 7, pp. 59346–59356, 2019.
 - [21] K. Kaynar and F. Sivrikaya, "Distributed attack graph generation," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 5, pp. 519–532, 2015.
 - [22] H. Li, Y. Wang, and Y. Cao, "Searching forward complete attack graph generation algorithm based on hypergraph partitioning," *Procedia Computer Science*, vol. 107, pp. 27–38, 2017.
 - [23] N. Cao, K. Lv, and C. Hu, "An attack graph generation method based on parallel computing," in *Proceedings of the International Conference on Science of Cyber Security*, August 2018.
 - [24] Y. Chen, Z. Liu, Y. Liu, and C. Dong, "Distributed attack modeling approach based on process mining and graph segmentation," *Entropy*, vol. 22, no. 9, Article ID 1026, 2020.
 - [25] J. Lee, D. Moon, I. Kim, and Y. Lee, "A semantic approach to improving machine readability of a large-scale attack graph," *The Journal of Supercomputing*, vol. 75, no. 6, pp. 3028–3045, 2019.
 - [26] X. Mao, M. Ekstedt, E. Ling, E. Ringdahl, and L. Robert, "Conceptual abstraction of attack graphs-A use case of securiCAD," in *Proceedings of the International Workshop on Graphical Models for Security*, June 2019.
 - [27] M. S. Barik and C. Mazumdar, "A graph data model for attack graph generation and analysis," in *Proceedings of the International Conference on Security in Computer Networks and Distributed Systems*, March 2014.
 - [28] S. Noel, E. Harley, K. H. Tam, M. Limiero, and M. Share, "CyGraph," *Handbook of Statistics*, Elsevier, vol. 35, pp. 117–167, Amsterdam, Netherlands, 2016.
 - [29] N. Polatidis, M. Pavlidis, and H. Mouratidis, "Cyber-attack path discovery in a dynamic supply chain maritime risk management system," *Computer Standards & Interfaces*, vol. 56, pp. 74–82, 2018.
 - [30] B. Yuan, Z. Pan, F. Shi, and Z. Li, "An attack path generation methods based on graph database," in *Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1, June 2020.

Research Article

Detection and Location of Malicious Nodes Based on Homomorphic Fingerprinting in Wireless Sensor Networks

Zhiming Zhang , Yu Yang, Wei Yang, Fuying Wu, Ping Li, and Xiaoyong Xiong

School of Software, Jiangxi Normal University, Nanchang 330027, China

Correspondence should be addressed to Zhiming Zhang; zzm_9650@163.com

Received 8 July 2021; Revised 23 August 2021; Accepted 30 August 2021; Published 24 September 2021

Academic Editor: Ahmed A. Abd El-Latif

Copyright © 2021 Zhiming Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The current detection schemes of malicious nodes mainly focus on how to detect and locate malicious nodes in a single path; however, for the reliability of data transmission, many sensor data are transmitted by multipath in wireless sensor networks. In order to detect and locate malicious nodes in multiple paths, in this paper, we present a homomorphic fingerprinting-based detection and location of malicious nodes (HFDFLMN) scheme in wireless sensor networks. In the HFDFLMN scheme, using homomorphic fingerprint and coding technology, the original data is divided into n packets and sent to the base station along n paths, respectively; the base station determines whether there are malicious nodes in each path by verifying the validity of the packets; if there are malicious nodes in one or more paths, the location algorithm of the malicious node is implemented to locate the specific malicious nodes in the path; if all the packets are valid, the original data is recovered. The HFDFLMN scheme does not need any complex evaluation model to evaluate and calculate the trust value of the node, nor any monitoring nodes. Theoretical analysis results show that the HFDFLMN scheme is secure and effective. The simulation results demonstrate promising outcomes with respect to key parameters such as the detection probability of the malicious path and the locating probability of the malicious node.

1. Introduction

With the rapid development of the Internet of Things, wireless sensor networks (WSNs) are not only widely used in transportation, agriculture, home furnishing, military, environmental monitoring, and other fields [1] but also used in smart city environments [2], smart grid [3], and smart healthcare system [4], and Underwater Sensor Networks (USNs) have become widespread and are being deployed in a wide range of applications ranging from harbor security to monitoring underwater pipelines and fish farms [5] recently. Since WSNs are constructed by a large number of sensor nodes in a wireless and multihop way, and the sensor nodes are restricted by calculation, storage, and communication, they are easy to be captured as malicious nodes by attackers. The existence of malicious nodes is a great threat to the network; by manipulating these malicious nodes, attackers can launch a variety of internal and external attacks [6], for

example, monitoring the important confidential information passing through these malicious nodes, injecting a large number of false data into sensor networks, destroying the normal data aggregation process by tampering with the data, launching various DoS attacks, and so on [7, 8]. Malicious nodes in multipath are more harmful because malicious nodes will send false data or pollution data to nodes in multiple paths at the same time, which is easy to cause the pollution data to continue to spread, thus consuming a large number of valuable resources of intermediate forwarding nodes and ultimately shorten the life cycle of the entire wireless sensor network; therefore, it is very important to detect, locate, and isolate the malicious nodes in multipath.

Detection of malicious nodes has always been a hot topic in wireless sensor networks; many scholars have proposed some effective detection schemes of malicious nodes. The current detection schemes of malicious nodes mainly focus on how to detect and locate malicious nodes in a single path;

however, for the reliability of data transmission, many sensor data are transmitted by multipath in wireless sensor networks [9–13]. In order to detect and locate malicious nodes in multiple paths, in this paper, we present a homomorphic fingerprinting-Based detection and location of malicious nodes (HFDFLMN) scheme in wireless sensor networks. In the HFDFLMN scheme, if the source node wants to send sensor data to the base station (BS), it divides the sensor data into n fragments and then encodes the n fragments to n new fragments; then, using homomorphic fingerprint technology, n packets are generated and sent to the base station along n paths, respectively. After receiving n packets with the same data number from n paths, the base station determines whether there are malicious nodes in each path by verifying the validity of the packets; if there are malicious nodes in one or multiple paths, the location algorithm of the malicious node is implemented to locate the specific malicious nodes in the path; if all the packets are valid, the original data will be recovered.

The main contributions of this paper are as follows. (1) A homomorphic fingerprinting-based detection and location of malicious nodes (HFDFLMN) scheme in wireless sensor networks is presented; using homomorphic fingerprint and coding technology, the HFDFLMN scheme can detect and locate malicious nodes in multiple paths. (2) In order to detect and locate malicious nodes, the HFDFLMN scheme does not need any complex evaluation model to evaluate and calculate the trust value of the node, nor any monitoring nodes. (3) The HFDFLMN scheme can resist the malicious node interfere with the base station to detect malicious nodes. (4) Theoretical analysis results show that the HFDFLMN scheme is secure and effective, and the simulation results show it can detect and locate malicious nodes with high probability by sending a small number of packets.

The rest of the paper is organized as follows. Section 2 introduces the related work. Preliminaries and the system model are described in Section 3. The HFDFLMN scheme is described in Section 4. Proof and analysis of related theorems are described in Section 5. Security analysis is described in Section 6. The performance evaluation is implemented in Section 7. Section 8 concludes this paper.

2. Related Work

At present, scholars have done a lot of work for detecting the malicious nodes in wireless sensor networks and have proposed some effective detection schemes. These schemes can be divided into multihop acknowledgment-based detection schemes, trust evaluation-based detection schemes, and statistics classification-based detection schemes.

Balakrishnan et al. [14] proposed a two-hop acknowledgment detection scheme (TWOACK) based on the checkpoint node. In the TWOACK scheme, each node in the forwarding path is the checkpoint node. If a node i receives a packet, it will send an acknowledgment packet to node j that two hops away from it. If node j does not receive the acknowledgment packet, it suspects the link between i and j to be a malicious link and sends a warning to the source node. However, the TWOACK scheme greatly increases conflict

and collision of network messages. To solve this problem, Xiao et al. [15] proposed a multihop acknowledgment-based detection scheme (CHEMAS). In the CHEMAS scheme, some nodes in the path from the source node to the base station are randomly selected as checkpoint nodes. After the checkpoint node receives a packet, it will send an acknowledgment packet to its upstream node. If an intermediate forwarding node in the path does not receive the specified number of acknowledgment packets, it will suspect that its next-hop node is a malicious node and send a warning to the source node. Although CHEMAS can greatly reduce the conflict and collision of network messages, if two or more malicious nodes are selected as checkpoint nodes in the CHEMAS scheme, the collusion of these malicious nodes will make the CHEMAS scheme invalid. In order to solve this problem, Liu et al. [16] proposed a new scheme based on multihop acknowledgment mechanism (PHACK). In the PHACK scheme, in order to detect and locate malicious nodes, each node in the forwarding path not only needs to forward normal packets but also needs to generate an acknowledgment packet for each packet and send it to the source node along a different path. However, these schemes based on multihop acknowledgment need to transmit a large number of confirmation packets, which will increase high communication overhead and greatly reduce the network life.

To improve the effect of malicious nodes detection, Yang et al. [17] proposed a malicious node detection model based on reputation with enhanced low energy adaptive clustering hierarchy, MNDREL. Based on the enhanced routing protocol, the cluster head nodes are selected and other nodes form different clusters by choosing the corresponding cluster head. By analyzing the reputation value for the parent node evaluated by the child node, the malicious nodes in the network are effectively identified. The MNDREL model outperformed in detecting malicious nodes in WSN with lower false alarm rate; however, the real-time performance of the MNDREL model has to be improved. Xiao et al. [18] proposed a sensor network reputation model based on Gaussian distribution (GRFSN). In this model, the trust value of each node is obtained by calculating the weight sum of direct reputation and indirect reputation, and finally, compared with the trust threshold, if the trust value of the node is less than the trust threshold, the node is a malicious node. This scheme only needs to determine a trust threshold, but the trust threshold is static, and the misjudgment rate of normal nodes being judged as malicious nodes is high. In order to detect the untrusted nodes in the network quickly and effectively and ensure the reliable operation of the network, Zheng et al. [19] proposed a network security mechanism based on trust management to deal with the threats faced by WSNs (DNSMTM). Based on the trusted access of nodes, this mechanism firstly calculates the local trust degree of nodes according to existing interaction behavior and further obtains the comprehensive trust degree of nodes that can reflect the trust degree of nodes, and the detection of malicious nodes is carried out according to the comprehensive trust degree of nodes. The mechanism can effectively detect malicious nodes, with a higher detection

rate, and reduce the energy consumption of nodes. Liao et al. [20] proposed a hybrid strategy monitoring-forwarding game detection scheme to detect selective forwarding attack (MSGSFS). In this scheme, a set of strategies is constructed by integrating factors such as packet loss, data corruption, and forwarding delay. The data sending node and its one-hop neighbor nodes select strategies from the set to perform the monitoring-forwarding game and collect the routing trust value of the suspicious node. In order to locate and isolate malicious nodes in the cluster, a distributed watchdog is run on each cluster head node to monitor and record the forwarding behavior of its one-hop neighbor cluster head node. This scheme can effectively alleviate selective forwarding attack in wireless sensor networks and has less energy consumption. Zhou et al. [21] proposed an improved trust evaluation model based on Bayesian and Entropy (ITEMBB). In this model, the direct trust value of the node is first calculated, and if the direct trust value is not reliable enough, the indirect trust value of the node is calculated. By integrating the direct trust value and the indirect trust value, a comprehensive trust value is obtained, and entropy is used to assign a greater weight to highly trusted nodes. To a certain extent, the model overcomes the limitations of subjective weight allocation, but the problem of static reputation value has not been solved. Zhou et al. [22] combined the neighbor node monitoring and watchdog mechanism to propose a cluster-based selective forwarding attack detection scheme (SMCSF). In this scheme, the nodes in the cluster are divided into three types: cluster head nodes, monitoring nodes, and cluster member nodes; by selecting the monitoring node in the cluster, the monitoring node performs the calculation and adjustment of the comprehensive reputation of the cluster head nodes and cluster member nodes in the cluster. And in this scheme, the monitoring nodes are not only responsible for calculating and adjusting the reputation of the node and judging and detecting malicious nodes in the cluster but also responsible for monitoring whether the cluster head node has malicious behaviors such as data tampering or packet loss during the data forwarding process. Although this scheme can quickly and accurately locate malicious nodes, the responsibility of monitoring nodes is too heavy.

Silva et al. [23] proposed a detecting scheme of malicious nodes based on statistics (IDSBS). The scheme matches and detects the abnormal behavior of nodes through a series of predetermined rules. Because there is no interaction between nodes, the false detection rate of the system is high in the initial stage. Liu et al. [24] proposed a detecting scheme of malicious nodes based on classification (MCMND). In this scheme, first, the multiple attributes of the node are modeled, and then, the known sensor nodes are learned by the multiple classification method based on likelihood. The posterior probability is used to generate a classifier, for any unknown type of nodes, the nodes are classified according to the class with the maximum posterior probability, so as to determine whether a node is a malicious node, but when the number of active nodes in the network is insufficient or the number of packets processed by nodes is small, the false detection rate is high. Aiming at the problem that the

existing malicious node detection methods in wireless sensor networks cannot be guaranteed by fairness and traceability of detection process, She et al. [25] present a blockchain trust model (BTM) for malicious node detection in wireless sensor networks. In BTM, through 3D space, it is realized by using blockchain intelligent contract and WSN quadrilateral measurement for localization of the detection of malicious nodes, and the consensus results of voting are recorded in the blockchain distributed. The model can effectively detect malicious nodes in WSNs and ensure the traceability of the detection process, but the consensus method in the model is the traditional POW workload proof method, which requires relatively large computational power and high energy consumption, so it is not especially suitable for the running environment of wireless sensor networks.

Li et al. [26] proposed a distributed and randomized detection algorithm to locate the attackers who inject polluted packets (IPAs). In this scheme, each node i maintains a set of suspicious nodes. In the beginning, all the neighbors of node i are added to a set of suspicious nodes; if the packets sent by its neighbor nodes are invalid, then the neighbor nodes that send valid packets are deleted from the suspicious nodes set; after n rounds of detection, the nodes in the set of suspicious nodes are malicious neighbors. Although the scheme can effectively detect malicious nodes in the network, it needs n rounds of detection, which will greatly increase the network communication overhead.

To sum up, all kinds of current research schemes have their own characteristics (Table 1). Comparison of advantages and disadvantages of each scheme makes a comparative analysis of relevant work. The detection schemes [14–16] based on multihop acknowledgment need to transmit a large number of acknowledgment packets, which will lead to high communication overhead. The detection schemes [17–22] based on trust evaluation need more monitoring nodes, which greatly increases the overhead of the network. And the current detection schemes of malicious nodes mainly focus on how to detect and locate malicious nodes in a single path. The HFDFLMN scheme proposed in this paper does not need any complex evaluation model to evaluate and calculate the trust value of the node, nor any monitoring nodes, and the HFDFLMN scheme can detect and locate malicious nodes in multiple paths.

3. Preliminaries and System Model

3.1. Preliminaries

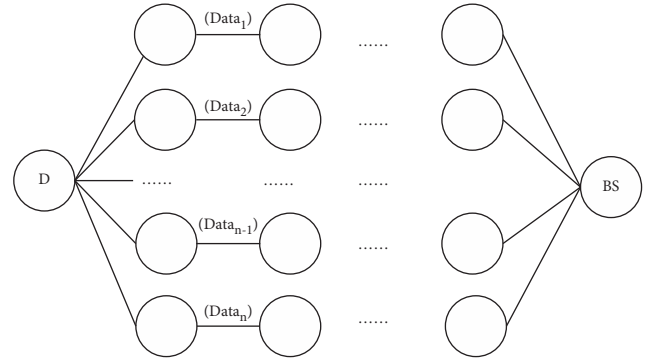
Homomorphic Fingerprinting. Hendricks et al. first proposed homomorphic fingerprinting in [27]. The fingerprinting functions of homomorphic fingerprinting belong to a family of universal hash functions also. Let IF_{q^ω} denote a field of order q^ω , let K be the set of fingerprinting key, and let $P_{q^\omega}: K \rightarrow IF_{q^\omega}[x]$ be a deterministic algorithm that outputs monic irreducible polynomials of prime degree γ with coefficients in IF_{q^ω} ; the polynomials are chosen with probabilities taken over the choice of input $r \in K$ uniformly at random; then a fingerprinting function $f_p(r, d): K \times IF_{q^\omega}^\delta \rightarrow IF_{q^\omega}^\gamma$ can be defined as

TABLE 1: Comparison of advantages and disadvantages of each scheme.

| Schemes | Advantages | Disadvantages |
|-------------|--|---|
| TWOACK [14] | Can be easily added to source routing protocols such as the DSR protocol | Greatly increases conflict and collision of network messages |
| CHEMAS [15] | Reduce the conflict and collision of network messages | Cannot resist the collusive attacks |
| PHACK [16] | Not only can detect a selective forwarding attack but also can recover the routing from the location at which the data were dropped | The scheme produces more ACK packets, which will consume more energy |
| MNDREL [17] | By analyzing the reputation value for the parent node evaluated by the child node, the malicious nodes in the network are effectively identified | The real-time performance of the deletion model has to be improved |
| GRFSN [18] | This scheme only needs to determine a trust threshold | Need complex evaluation model and the trust threshold is static, and the misjudgment rate of normal nodes being judged as malicious nodes is high |
| DNSMTM [19] | Can effectively detect malicious nodes and, with a higher detection rate, reduce the energy consumption of nodes | Need a complex evaluation model and the responsibility of monitoring nodes is too heavy |
| MSGSFS [20] | Can effectively alleviate selective forwarding attack and has less energy consumption | Need a complex game model and it can only solve the malicious packet loss behavior of single-hop nodes |
| ITEMBB [21] | The model overcomes the limitations of subjective weight allocation | Need complex evaluation model and the problem of static reputation value has not been solved |
| SMCSF [22] | Combine the neighbor node monitoring and watchdog mechanism | The responsibility of monitoring nodes is also heavy |
| IDSBS [23] | The collected information and its treatment are performed in a distributed way | The false detection rate of the system is high in the initial stage |
| MCMND [24] | Use multivariate classification to classify nodes | The false detection rate is high |
| BTM [25] | Using a blockchain intelligent contract, which can ensure the traceability of the detection process | The consensus method in the model is the traditional POW workload proof method, which requires relatively large computational power and high energy consumption |
| IPA [26] | Each node maintains a suspicious node set | Need n rounds of detection, which will greatly increase the network communication overhead |

$fp(r, d(x))$: $p(x) \leftarrow P_{q^\omega}(r)$; $\text{return}(d(x) \bmod \cdot p(x))$. A fingerprinting function $fp(r, d): K \times IF_{q^\omega}^\delta \rightarrow IF_{q^\omega}^\gamma$ is homomorphic if $fp(r, d) + fp(r, d') = fp(r, d + d')$ and $b \cdot fp(r, d) = fp(r, b \cdot d)$ for any $r \in K$ and $d, d' \in IF_{q^\omega}^\delta, b \in IF_{q^\omega}^\gamma$. Let (encode, decode) be a linear erasure code with coefficients $b_{ij} \in IF_{q^\omega}$, for $i \in [1, n]$ and $j \in [1, m]$; if $d_1, \dots, d_n \leftarrow \text{encode}^\delta(B)$, then for a homomorphic fingerprinting function $fp(r, d): K \times IF_{q^\omega}^\delta \rightarrow IF_{q^\omega}^\gamma$, the following equation holds: $fp(r, d_i) = \text{encode}_i^\gamma(fp(r, d_1), \dots, fp(r, d_m))$, where $r \in K$ and $i \in [1, n]$.

3.2. Network Model. The sensor network is composed of ordinary nodes, malicious nodes, and base station (BS). Before deployment, each node i is assigned a unique identity ID_i , a random number r , $r \in IF_{2^\omega}$, and a symmetric key $K_{i,BS}$ shared with a base station. After the network is deployed to the target area, all nodes do not move. Adopting the method of [13], each node establishes multiple disjoint paths with the base station, and each node sends the data to the base station through multiple paths, for example. In Figure 1, the source node D and the base station have established n data transmission paths. Assuming that node D wants to send the data to the base station, it first divides the data into n fragments that are different from each other and encodes the n fragments to n new fragments; then, using

FIGURE 1: Source node D transmits data to a base station through multiple paths.

homomorphic fingerprint technology, n packets are generated and sent to the base station along n different paths, respectively. When the base station receives n packets, if all the packets are valid, it will recover the original data.

3.3. Attack Model and Security Goal. The HFDFLMN scheme assumes that any intermediate forwarding nodes can be captured as malicious nodes by the attackers. These malicious nodes can launch pollution attacks by injecting false data into the network, forging or modifying the packets. The HFDFLMN scheme does not consider other attacks such as

selective forwarding attacks but only considers pollution attacks. When the malicious node in the path receives the data, it will forge or modify the data with probability q and then forward it to the next-hop node. In the HFDFLMN scheme, it is assumed that the calculation, storage, and communication capabilities of the base station are not limited, and the attackers can only capture ordinary sensor nodes, but not the base station.

Nowadays, there are several WSN standards (e.g., IEEE 802.15.4) that use different security levels at each layer. For instance, the network part of a packet is signed and encrypted with a network key and a data link layer with a DLL key. When an intermediate node receives a packet to retransmit, the DLL part needs to be verified; if it is not signed, the intermediate node drops the packet. Although the signature and encryption method can verify whether the packet has been modified, it cannot locate the malicious node that modifies the packet. The security goal of the HFDFLMN scheme is not only to verify whether the packet is polluted but also to detect and locate the malicious nodes that launch pollution attacks.

4. Homomorphic Fingerprinting-Based Detection and Location of Malicious Nodes

The HFDFLMN scheme proposed in this paper is divided into five steps: the source node generates the packets, the intermediate node forwards the packet, the base station detects the path of pollution attack, the base station locates the malicious node or malicious link, and base station recovers the original data.

4.1. Generating the Packets

4.1.1. Generating Data Segmentation. If the source node wants to send the data to the base station, it first divides the data into n fragments that are different from each other, namely, $\text{data} = \langle f_1, \dots, f_n \rangle$.

4.1.2. Coding Data Segmentation. Then, the source node generates n linearly independent vectors, the elements of the vectors are randomly picked from the field IF_{2^w} , and the vector is denoted as $[g_{i,1}, \dots, g_{i,n}]$, $i = 1, \dots, n$. According

to the following equation, the source node can get n new fragments, which are denoted as Y_i , $i = 1, \dots, n$.

$$\begin{bmatrix} g_{1,1} & \cdots & g_{1,n} \\ \vdots & \ddots & \vdots \\ g_{n,1} & \cdots & g_{n,n} \end{bmatrix} \times \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}. \quad (1)$$

4.1.3. Generating and Sending the Packets. After coding data segmentation, the source node generates n packets, which are denoted as $\text{Packet}_i = \langle \text{Seq_Number}_k, hp(r, f_i), G_i, Y_i \rangle$, $i = 1, \dots, n$, where the n packets have the same number which is denoted by Seq_Number_k , $hp(r, f_i)$ denotes the fingerprinting of f_i , which is computed by fingerprinting function $hp(\cdot)$ and the random number r , $r \in IF_{2^w}$, $G_i = [g_{i,1}, \dots, g_{i,n}]$ denotes the coding vector of f_i , and Y_i denotes the new data fragment after coding; then, the Packet_i is sent to the base station along n different paths, respectively.

4.2. Forwarding the Packets. All intermediate forwarding nodes maintain a data forwarding table (DFT), which is shown in Table 2, where the Seq_Number field stores the packet number, and the Finger_printing field stores the fingerprinting of f_i , the Encoding_Vector field stores the coding vector of f_i , the Encoded_DataBlock field stores the new data fragment after coding, and the DFT only stores the packets that were forwarded the last three times. When the intermediate forwarding node j receives $\text{Packet}_i = \langle \text{Seq_Number}_k, hp(r, f_i), G_i, Y_i \rangle$, it will delete the first record stored in the DFT and store the currently received packet in the DFT, which makes the DFT only store the packets that were forwarded the last three times and facilitate the base station query, and then it forwards the Packet_i to the next intermediate forwarding node.

4.3. Detecting the Path of Pollution Attack. After the base station receives n packets with the same number from n paths, it first gets $hp(r, f_i)$, Y_i and G_i from the $\text{Packet}_i = \langle \text{Seq_Number}_k, hp(r, f_i), G_i, Y_i \rangle$, $i = 1, \dots, n$ and computes $hp(r, Y_i)$, $i = 1, \dots, n$ respectively; then, it randomly picks t_1, \dots, t_n from the field IF_{2^w} and constructs a new vector $V = [v_1, \dots, v_n]$ according to

$$[v_1, \dots, v_n] = [t_1, \dots, t_n] \begin{bmatrix} G_1 \\ \cdots \\ G_n \end{bmatrix} = \left[\sum_{i=1}^n t_i g_{i1}, \dots, \sum_{i=1}^n t_i g_{in} \right], \quad (2)$$

$$v_1 hp(r, f_1) + \cdots + v_n hp(r, f_n) = t_1 hp(r, Y_1) + \cdots + t_n hp(r, Y_n). \quad (3)$$

The base station can validate the validity of n packets according to equation (3). If equation (3) holds, all the n packets are valid, and it will be performed in Section 4.5 to recover the original data; otherwise, it means that malicious nodes polluted the packets in one path or more paths. Then,

the base station can detect which packet is polluted according to equation (4). If equation (4) does not hold, the Packet_i is polluted, and there are malicious nodes in the path; the base station will execute Algorithm 1 in Section 4.4 to detect and locate the malicious node.

$$hp(r, Y_i) = G_i \times [hp(r, f_1), \dots, hp(r, f_n)]^T, \quad i = 1, \dots, n. \quad (4)$$

4.4. Locating the Malicious Node. When the base station finds the path of pollution attack and the pollution packet $\text{Packet}_m = \langle \text{Seq_Number}_k, hp(r, f_m), G_m, Y_m \rangle$, it assumes that there are m hops in the attack path from the source node to the base station, it is represented by $(n_1, n_2, \dots, n_m, \text{BS})$, where n_1 represents the source node, and the remaining nodes represent intermediate forwarding nodes in the path. In order to locate the malicious nodes in the path, from the source node, the base station first informs each node to send the response packet to the base station along the attack path in turn, and the response packet p_i is generated according to equation (5). In equation (5), ID_i represents the identity of node n_i , and $\text{SPacket}_i = \langle n_i, DF T, \text{Seq_Number}_k \| n_i, DF T, hp(r, f_i) \| n_i, DF T, G_i \| n_i, DF T, Y_i \rangle$, p_{i-1} represents the response packet sent by the previous hop node n_{i-1} , Timestamp represents the timestamp, and $\|$ represents the connection operation.

$$p_i = E_{K_{i,BS}}(ID_i \| \text{SPacket}_i \| p_{i-1} | \text{Timestamp}). \quad (5)$$

After the base station receives the response packet p_m , it will execute Algorithm 1 to locate the malicious node or the malicious link. From back to front, first, it sequentially decrypts p_i with the symmetric key $K_{i,BS}$ shared with the node n_i and gets the ID_i and SPacket_i ; then, from the node n_1 , the base station compares SPacket_i with the pollution packet Packet_m received by the base station in turn; if it is equal, it means that node n_i is a malicious node or the link between node n_i and node n_{i-1} is a malicious link.

4.5. Recovering the Original Data. After the base station receives n linearly independent packets with the same number from n paths, it can validate the validity of n packets according to equation (3); if equation (3) holds, all the n packets are valid, and it will recover the original data. It first gets Y_i and G_i from the $\text{Packet}_i = \langle \text{Seq_Number}_k, hp$

$(r, f_i), G_i, Y_i \rangle, i = 1, \dots, n$, and generates the vector coefficient matrix T , as shown in equation (6). Because n vectors G_i are vector linearly independent, the coefficient matrix T is full rank, and the base station can get the inverse matrix T^{-1} and recover the original Data = $\langle f_1, \dots, f_n \rangle$ according to equation (7).

$$T = \begin{bmatrix} g_{1,1} & \cdots & g_{1,n} \\ \vdots & \ddots & \vdots \\ g_{n,1} & \cdots & g_{n,n} \end{bmatrix}, \quad (6)$$

$$\begin{bmatrix} f_1 \\ \vdots \\ f_2 \end{bmatrix} = T^{-1} \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}. \quad (7)$$

5. Proof and Analysis of Related Theorems

5.1. Proof of Malicious Path Detectability. The base station can validate the validity of n packets according to equation (3); if equation (3) holds, all the n packets are valid, and it will recover the original data; otherwise, the base station can detect which packet is polluted according to equation (4); if equation (4) does not hold, the Packet_i is polluted, and there are malicious nodes in the path of sending the pollution packet. This section will prove the correctness of equations (3) and (4).

Theorem 1. *After the base station receives n linearly independent packets with the same number from n paths, if all the packets it receives are valid, then equation (3) holds; if there are t ($t < n$) packets that are polluted, then equation (3) does not hold.*

Proof

(1) If all the packets received by the base station are valid, then

$$\begin{aligned} v_1 hp(r, f_1) + \cdots + v_n hp(r, f_n) &= hp(r, v_1 f_1) + \cdots + hp(r, v_n f_n) \\ &= hp(r, [v_1, \dots, v_n] [f_1, \dots, f_n]^T) \\ &= hp\left(r, [t_1, \dots, t_n] \begin{bmatrix} G_1 \\ \vdots \\ G_n \end{bmatrix} [f_1, \dots, f_n]^T\right) \\ &= hp\left(r, [t_1, \dots, t_n] \begin{bmatrix} g_{1,1} & \cdots & g_{1,n} \\ \vdots & \ddots & \vdots \\ g_{n,1} & \cdots & g_{n,n} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}\right) \\ &= hp\left(r, [t_1, \dots, t_n] \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}\right) \\ &= hp(r, t_1 Y_1 + \cdots + t_n Y_n) \\ &= t_1 hp(r, Y_1) + \cdots + t_n hp(r, Y_n) \\ &\quad hp(r, v_1 f_1) + \cdots + hp(r, v_n f_n). \end{aligned} \quad (8)$$

That is, Eq. (3) holds.

(2) Assuming that one of the n packets received by the base station is polluted by the forwarding malicious node n_i in the path, and the polluted packet is $Packet'_i = \langle Seq_Number_k, hp(r, f_i^p), G_i^p, Y_i^p \rangle$, where

f_i^p, G_i^p, Y_i^p are false data injected by the malicious node n_i , $hp(r, f_i^p)$ denotes the fingerprinting of f_i^p , computed by fingerprinting function $hp(\cdot)$ and the random number $r \in IF_{2^w}$, then,

$$\begin{aligned}
v_1 hp(r, f_1) + \dots + v_i hp(r, f_i^p) + \dots + v_n hp(r, f_n) &= hp(r, v_1 f_1) + \dots + hp(r, v_i f_i^p) + \dots + hp(r, v_n f_n) \\
&= hp\left(r, [v_1, \dots, v_i, \dots, v_n] [f_1, \dots, f_i^p, \dots, f_n]^T\right) \\
&= hp\left(r, [t_1, \dots, t_n] \begin{bmatrix} G_1 \\ \dots \\ G_i^p \\ \dots \\ G_n \end{bmatrix} [f_1, \dots, f_i^p, \dots, f_n]^T\right) \\
&= hp\left(r, [t_1, \dots, t_n] \begin{bmatrix} g_{1,1} & \dots & g_{1,n} \\ \dots & \dots & \dots \\ g_{i,1}^p & \dots & g_{i,n}^p \\ \dots & \dots & \dots \\ g_{n,1} & \dots & g_{n,n} \end{bmatrix} \begin{bmatrix} f_1 \\ \dots \\ f_i^p \\ \dots \\ f_n \end{bmatrix}\right) \\
&= hp\left(r, [t_1, \dots, t_n] \begin{bmatrix} Y_1^p \\ \dots \\ Y_i^p \\ \dots \\ Y_n^p \end{bmatrix}\right) \\
&= hp(r, t_1 Y_1^p + \dots + t_i Y_i^p + \dots + t_n Y_n^p) \\
&= t_1 hp(r, Y_1^p) + \dots + t_i hp(r, Y_i^p) + \dots + t_n hp(r, Y_n^p) \\
&\quad hp(r, v_1 f_1) + \dots + hp(r, v_i f_i^p) + \dots + hp(r, v_n f_n).
\end{aligned} \tag{9}$$

Because the malicious node n_i has no $\langle f_1, \dots, f_{i-1}, f_{i+1}, \dots, f_n \rangle$ and $\langle G_1, \dots, G_{i-1}, G_{i+1}, \dots, G_n \rangle$, it cannot construct $\langle Y_1^p, \dots, Y_n^p \rangle$ and make $\langle Y_1^p = Y_1, \dots, Y_n^p = Y_n \rangle$; as a result, $t_1 hp(r, Y_1^p) + \dots + t_i hp(r, Y_i^p) + \dots + t_n hp(r, Y_n^p) \neq v_1 hp(r, f_1) + \dots + v_i hp(r, f_i^p) + \dots + v_n hp(r, f_n)$.

So, without considering the link error of the network, if one or more packets are polluted, then Eq. (3) does not hold.

Theorem 1 is proved. \square

Theorem 2. After the base station receives n linearly independent packets with the same number from n paths, if all the packets received by the base station are valid, then equation (4) holds; otherwise, equation (4) does not hold, and the path of sending pollution packet is the malicious path.

Proof

(1) Assuming that $Packet_i = \langle Seq_Number_k, hp(r, f_i), G_i, Y_i \rangle$ is valid, then,

$$\begin{aligned}
G_i \times [hp(r, f_1), \dots, hp(r, f_n)]^T &= [g_{i,1}, \dots, g_{i,n}] \times [hp(r, f_1), \dots, hp(r, f_n)]^T \\
&= g_{i,1} hp(r, f_1) + \dots + g_{i,n} hp(r, f_n) \\
&= hp(r, g_{i,1} f_1) + \dots + hp(r, g_{i,n} f_n) \\
&= hp(r, g_{i,1} f_1 + \dots + g_{i,n} f_n) \\
&= hp\left(r, [g_{i,1}, \dots, g_{i,n}] \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}\right) \\
&= hp(r, Y_i).
\end{aligned} \tag{10}$$

That is, if the packet has not been modified, $G_i \times [hp(r, f_1), \dots, hp(r, f_n)]^T = hp(r, Y_i)$, therefore, (4) holds.

(2) Assuming that one of the n packets received by the base station is polluted by the forwarding malicious node n_i in the path, and the polluted packet is

$\text{Packet}'_i = \langle \text{Seq_Number}_k, hp(r, f_i^p), G_i^p, Y_i^p \rangle$, where f_i^p , G_i^p, Y_i^p are false data injected by the malicious node n_i , $hp(r, f_i^p)$ denotes the fingerprinting of f_i^p , computed by fingerprinting function $hp(\cdot)$ and the random number $r \in IF_{2^\omega}$, then

$$\begin{aligned}
& G_i^p \times [hp(r, f_1), \dots, hp(r, f_i^p), \dots, hp(r, f_n)]^T \\
&= [g_{i,1}^p, \dots, g_{i,i}^p, \dots, g_{i,n}^p] \times [hp(r, f_1), \dots, hp(r, f_i^p), \dots, hp(r, f_n)]^T \\
&= g_{i,1}^p hp(r, f_1) + \dots + g_{i,i}^p hp(r, f_i^p) + \dots + g_{i,n}^p hp(r, f_n) \\
&= hp(r, g_{i,1}^p f_1) + \dots + hp(r, g_{i,i}^p f_i^p) + \dots + hp(r, g_{i,n}^p f_n) \\
&= hp(r, g_{i,1}^p f_1 + \dots + g_{i,i}^p f_i^p + \dots + g_{i,n}^p f_n) \\
&= hp \left(r, [g_{i,1}^p, \dots, g_{i,i}^p, \dots, g_{i,n}^p] \begin{bmatrix} f_1 \\ \dots \\ f_i^p \\ \dots \\ f_n \end{bmatrix} \right) \\
&= hp(r, Y_i^p).
\end{aligned} \tag{11}$$

Because the malicious node n_i has no $\langle f_1, \dots, f_{i-1}, f_{i+1}, \dots, f_n \rangle$, it cannot construct $Y_i^{p'}$ and make $Y_i^{p'} = Y_i^p$; as a result, $hp(r, Y_i^{p'}) \neq hp(r, Y_i^p)$.

So, without considering the link error of the network, if one or more packets are polluted, then (4) does not hold, and the path to send the pollution packet is the malicious path.

Theorem 2 is proved. \square

5.2. Probability Analysis of Legitimate Nodes Being Misjudged as Malicious Nodes. Because of the link error, a legitimate node will be misjudged as a malicious node by forwarding a packet distorted by the link error. This section will analyze the probability of the legitimate node being misjudged as a malicious node in the case of a link error.

Theorem 3. *Assuming that the probability of link error is q , and the number of packets transmitted in the path is S in time period T , the probability of legitimate nodes being misjudged as malicious nodes is*

$$P_m = \sum_{k=1}^S \binom{S}{k} q^k (1-q)^{S-k}. \tag{12}$$

Proof. Let X be the times that misjudged packets are detected, it is obvious that X satisfies the binomial distribution of parameters S and q , that is, $X \sim b(S, q)$, and the distribution law of X is as follows:

$$P\{X = k\} = \binom{S}{k} q^k (1-q)^{S-k}, \quad k = 0, 1, \dots, S. \tag{13}$$

Therefore, the probability of legitimate nodes being misjudged as malicious nodes is

$$\begin{aligned}
P_m &= P\{X \geq 1\} \\
&= \sum_{k=1}^S \binom{S}{k} q^k (1-q)^{S-k}.
\end{aligned} \tag{14}$$

So, Theorem 3 is proved. \square

5.3. Time Complexity Analysis of the Algorithm. If there are malicious nodes in the path, the base station will execute Algorithm 1 in Section 4.4 to detect and locate the malicious node or malicious link, this section will analyze the time complexity of Algorithm 1.

Basic operations of Algorithm 1 are to get the SPacket_i from the response packet and the comparison of SPacket_i with the pollution packet Packet_m . Getting the SPacket_i from the response is mainly to perform m cyclic operations; therefore, the time complexity of basic operation of getting the SPacket_i from the response is $O(m)$. The comparison of SPacket_i with the pollution packet Packet_m is to search whether the pollution packet Packet_m is in the array SPacket_i ; therefore, the time complexity of basic operation of the comparison of SPacket_i with the pollution packet Packet_m is $O(m)$, too. So, the time complexity of the two basic operations is $O(2m)$; that is, the time complexity of Algorithm 1 is $O(n)$.

TABLE 2: Data forwarding table (DFT).

| Seq_Number | Finger_printing | Encoding_Vector | Encoded_DataBlock |
|-------------------------|-----------------|-----------------|-------------------|
| | | | |
| Seq_Number _k | $hp(r, Y_i)$ | G_i | Y_i |
| | | | |

6. Security Analysis

In the HFDLMN scheme, the malicious nodes not only can launch pollution attacks by injecting false data into the network, forging or modifying the packets, but also can also modify or delete the response packet p_i sent by its previous node, so that the malicious nodes can avoid or interfere with the base station to perform the detection of the malicious nodes. Because the pollution attack launched by a malicious node can be detected by (3) and (4), and if the malicious node normally forwards the `[[parms resize(1),pos(50,50),size(200,200),bgcol(156)]]` scheme resists to the malicious nodes avoiding or interfering with the base station to perform the detection of the malicious nodes.

Theorem 4. *Any malicious node can be detected after modifying, deleting, or not sending response packets.*

Proof. In order to interfere with the detection of malicious nodes performed by the base station, when the base station informs each node to send the response packet p_i to the base station along the path, in turn, the malicious node n_i can perform the following operations: (a) Attempt to modify the data of the response packet p_{i-1} sent by its previous node; however, the response packet p_{i-1} is a key chain generated by encrypting the time stamp, p_{i-2} , the identity of the node n_{i-1} , and $SPacket_{i-1}$ with the symmetric key $K_{i-1,BS}$ shared by node n_{i-1} and base station; that is, $p_{i-1} = E_{K_{i-1,BS}}(ID_{i-1} || SPacket_{i-1} || p_{i-2} || Timestamp)$, because the malicious node n_i does not have the symmetric key $K_{i-1,BS}$ shared by the node n_{i-1} and the base station, and it cannot modify the data in the response packet p_{i-1} sent by its previous node. (b) Try to delete the data in the response packet p_{i-1} sent by its previous node, also because the malicious node n_i does not have the symmetric key $K_{i-1,BS}$ shared by the node n_{i-1} and the base station, so it cannot delete the data in the response packet p_{i-1} sent by its previous node. (c) Try not to send its own query response packet to the base station, that is, directly forwards the received response packet p_{i-1} from its previous node to its next node n_{i+1} . In this case, according to Algorithm 1, when the base station tries to decrypt the response packet p_i with the symmetric key $K_{i,BS}$ shared with the malicious node n_i , the malicious node did not send its own response packet, so the base station cannot correctly decrypt the response packet p_i ; therefore, it can be determined that the node n_i is a malicious node. (d) Try to send false data to the base station and interfere with the detection of malicious nodes. For example, the malicious node sends unmodified data to the base station; according to Algorithm 1, the base station can correctly locate the malicious link composed of the malicious node and its next-hop node; similarly, the malicious node can also

send a modified data to the base station; according to Algorithm 1, the base station can correctly locate the malicious link composed of the malicious node and its next-hop node.

In summary, in the HFDLMN scheme, any malicious node can be detected after modifying, deleting, or not sending response packets.

So, Theorem 4 is proved. \square

7. Simulation

In this paper, the performance of the HFDLMN scheme is evaluated from the aspects of the detection probability of malicious path, the location probability of malicious node, and the false detection probability of normal nodes and paths. The simulation experiment environment is carried out on OMNeT++ platform, with 100 nodes randomly distributed in a square area of $400m \times 400m$, each node is assigned a unique *ID*, the nodes will not move after deployment, and the base station is deployed in the center of the area. By adjusting the communication range of each node, each node has at least four neighbor nodes, and each node establishes four disjoint paths to the base station. Some nodes in the network are randomly selected as data source nodes and malicious nodes, and others as intermediate forwarding nodes. The source node sends the packet to the base station by multihop every 1 second, and the length of each packet is 256 bytes. The initial energy of each node is 1J, and the energy consumption of transmission and receiving is 50 nJ/bit. When a malicious node becomes an intermediate forwarding node, it will forge or modify packets with a probability from 0.1 to 0.7. For each set parameter, the average value obtained by 100 simulations is taken. The parameter settings of the experimental simulation are shown in Table 3.

Figure 2 describes the detection probability of a malicious path when the malicious node forges or modifies packets with a probability of 0.1, 0.3, 0.5, and 0.7, and there is a malicious node in one of the four paths from the source node to the base station. From Figure 2, it can be seen that the number of packets that need to be sent to successfully detect malicious paths is related to the probability q of malicious node modifying data. The higher the probability of malicious node modifying data, the less packets need to be sent to successfully detect the malicious path; for example, when the probability q of malicious node modifying data is 0.3, in order to detect the malicious path successfully, the source node needs to send 14 packets; when the probability q of malicious nodes modifying data is 0.5, the base station can successfully detect the malicious path by only sending 9 packets.

Figure 3 describes the detection probability of a malicious path when the malicious node forges or modifies

```

Input: the path of pollution attack , the pollution packet Packetm, and the response packet pm
Output: malicious node or malicious link
For ( $i = m; i > 1; i--$ ) do
   $\mathbf{p}'_i = \mathbf{D}_{\kappa_{i,BS}}(\mathbf{p}_i)$ 
  If cannot get accurately  $\mathbf{p}'_i$  then
    Return  $\mathbf{ID}_{i-1}$ 
  End if
   $\mathbf{SPacket}[i] = \mathbf{SPacket}_i$ 
   $\mathbf{ID}[i] = \mathbf{ID}_i$ 
End for
For( $i = 1; i \leq m; i++$ ) do
  If  $\mathbf{SPacket}[i] == \mathbf{Packet}_m$  then
    Return  $\mathbf{ID}_{i-1}$  and  $\mathbf{ID}_i$ 
  End if
End for

```

ALGORITHM 1: Location of malicious nodes.

TABLE 3: Experimental simulation parameters.

| Parameter | Value or range |
|---|------------------|
| Network deployment area (m) | 400×400 |
| Number of nodes in the network | 100 |
| Initial energy of each node (J) | 1 |
| The energy consumption of the transmission and receiving (nJ/bit) | 50 |
| Time interval of sending packet (S) | 1 |
| The number of malicious nodes | 20 |
| The probability of malicious nodes modifying packets | 0.1–0.7 |
| The probability of link error | 0.005–0.06 |
| The simulation time (S) | 600 |

packets with a probability of 0.1, 0.3, and 0.5, and the number of paths with malicious nodes is 2 and 3. From Figure 3, it can be seen that the number of packets that need to be sent to successfully detect malicious paths is not only related to the probability q of malicious node modifying data but also related to the number of paths with malicious nodes. The higher the probability of malicious node modifying data and the more number of paths with malicious nodes, the less packets need to be sent to successfully detect the malicious path; for example, when the probability q of malicious node modifying data is 0.1 and the number of paths with malicious nodes is 2, in order to detect the malicious path successfully, the source node needs to send 8 packets; when the probability q of malicious node modifying data is 0.3 and the number of paths with malicious nodes is 3, the base station can successfully detect the malicious path by only sending 4 packets.

Figure 4 describes the locating probability of the malicious node when the malicious node forges or modifies packets with a probability of 0.1 and 0.3, and the number of paths with malicious nodes is 1, 2, and 3. From Figure 4, it can be seen that with the probability increase of the malicious node modifying data and the number increase of malicious paths, the probability of successfully locating malicious nodes will increase; for example, when there is a malicious node in only one path and the probability q of

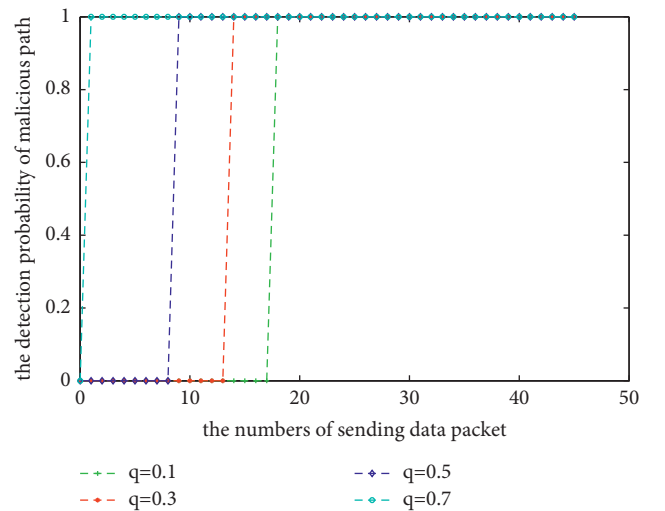


FIGURE 2: The detection probability of a malicious path when there is a malicious node in only one path.

malicious node modifying data is 0.1, the source node sends 15 packets and the probability of successfully locating the malicious node is about 84%; when there are malicious nodes in two paths and the probability q of malicious nodes modifying data is 0.3, the source node only sends 10 packets

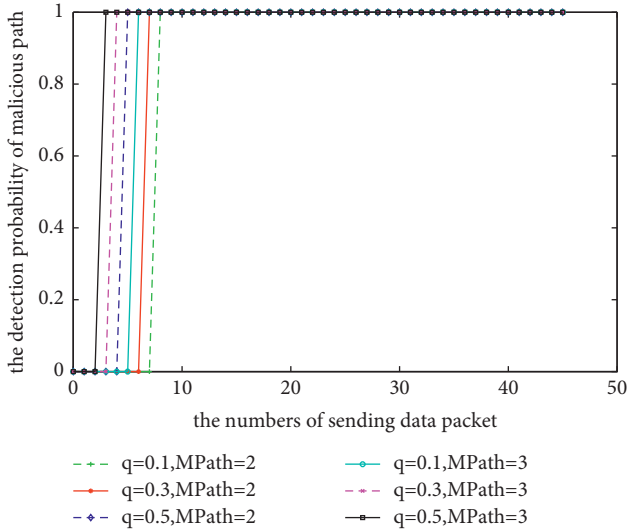


FIGURE 3: The detection probability of a malicious path when there are malicious nodes in multiple paths.

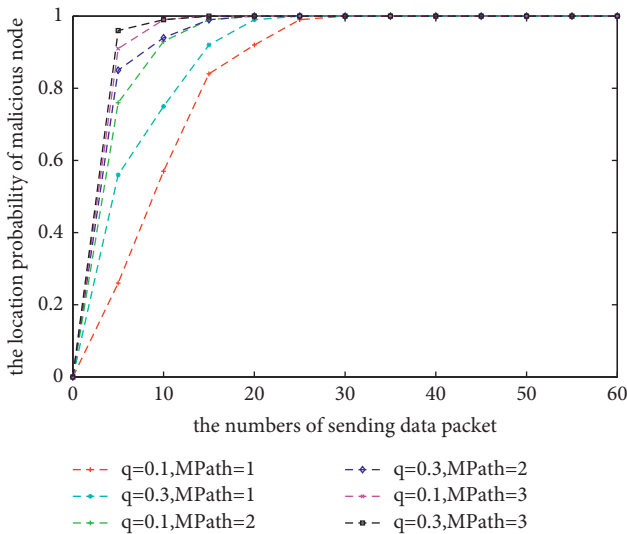


FIGURE 4: The locating probability of the malicious node.

and the probability of successfully locating the malicious node is about 94%.

Because of the link error, a legitimate node will forward a packet distorted by the link error, so that the legitimate node and path are misjudged as malicious node and path. Figure 5 describes the probability of the legitimate node and path being misjudged as malicious node and path when the probability of link error is from 0.005 to 0.06 and the number of packets transmitted in the path is 100 in a certain period of time. From Figure 5, it can be seen that with the probability increase of the link error, the probability of the legitimate node and path being misjudged as malicious node and path will increase; for example, when the probability of link error is 0.01, the probability of the legitimate node and path being misjudged as malicious node and path is about 5%, and when the probability of link error is 0.05, the false detection probability of the legitimate node and path is about 19%.

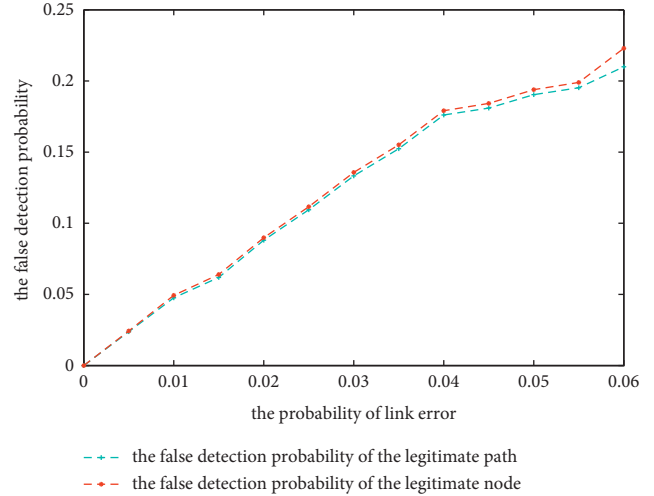


FIGURE 5: The false detection probability of legitimate nodes and paths.

8. Conclusions

To detect and locate malicious nodes in multiple paths, this paper presents a malicious node detection and location scheme based on homomorphic fingerprint and coding technology in wireless sensor networks, HFDLMN. In the HFDLMN scheme, the source node generates n packets and sends them to the base station along n paths, respectively; the base station determines whether there are malicious nodes in each path by verifying the validity of the packets; if there are malicious nodes in one or some paths, the location algorithm of a malicious node is implemented to locate the specific malicious nodes in the path. The HFDLMN scheme does not need any complex evaluation model to evaluate and calculate the trust value of the node, nor any monitoring nodes. Using a key chain, the HFDLMN scheme can resist malicious nodes to avoid or interfere with the base station to detect malicious nodes. Theoretical analysis results show that the HFDLMN scheme is secure and effective, the simulation results demonstrate that the HFDLMN scheme can effectively detect malicious paths and malicious nodes, with a higher detection rate; for example, if there are malicious nodes in two paths and the probability q of malicious nodes modifying data is 0.3, the source node only sends 10 packets and the probability of successfully locating the malicious node is about 94%. In the future, we aim to extend this work into designing a new detection and location of malicious nodes scheme among Internet of Things devices.

Data Availability

No data were used to support this study.

Conflicts of Interest

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

Acknowledgments




This work was supported in part by the National Natural Science Foundation of China under Grant 62002143 and in part by the Natural Science Foundation of Jiangxi Province under Grant 20192BAB217007.

References

- [1] E. Nurellari, D. McLernon, M. Ghogho, and S. Aldalameh, "Distributed binary event detection under data-falsification and energy-bandwidth limitation," *IEEE Sensors Journal*, vol. 16, no. 16, pp. 6298–6309, 2016.
- [2] A. Alanezi, B. Abd-El-Atty, H. Kolivand et al., "Securing digital images through simple permutation-substitution mechanism in cloud-based smart city environment," *Security and Communication Networks*, vol. 2021, Article ID 6615512, 17 pages, 2021.
- [3] U. Baroudi and M. E. Haque, "Ambient self-powered cluster-based wireless sensor networks for industry 4.0 applications," *Soft Computing*, vol. 25, no. 4, pp. 1859–1884, 2021.
- [4] R. Yadav, W. Zhang, I. A. Elgendy, G. Dong, and S. Prakash, "Smart healthcare: RL-based task offloading scheme for edge-enabled sensor networks," *IEEE Sensors Journal*, vol. 2021, Article ID 3096245, 2021.
- [5] A. Chaaf, M. S. A. Muthanna, A. Muthanna, S. Alhelaly, and A. A. A. El-Latif, "REVOHPR: relay-based void hole prevention and repair by virtual routing in clustered multi-AUV underwater wireless sensor network," *Security and Communication Networks*, vol. 2021, Article ID 9969605, 20 pages, 2021.
- [6] J. Grover and S. Sharma, "Security issues in wireless sensor network—a review," in *Proceeding of the 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 397–404, Noida, India, September 2016.
- [7] B. Zhu, V. Addada, S. Setia, S. Jajodia, and S. Roy, "Efficient distributed detection of node replication attacks in sensor networks," in *Proceeding of the 23rd Annual Computer Security Applications Conference (ACSAC '07)*, pp. 257–267, Miami Beach, FL, USA, December 2007.
- [8] R. Rongxing Lu, X. Xiaodong Lin, H. Haojin Zhu, X. Xiaohui Liang, and X. Xuemin Shen, "BECAN: a bandwidth-efficient cooperative authentication scheme for filtering injected false data in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 1, pp. 32–43, 2012.
- [9] A. Liu, Z. Zheng, C. Zhang, Z. Chen, and X. Shen, "Secure and energy-efficient disjoint multipath routing for WSNs," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 7, pp. 3255–3265, 2012.
- [10] G. S. M. D'Souza and R. J. Varaprasad, "Digital signature-based secure node disjoint multipath routing protocol for wireless sensor networks," *Sensors Journal, IEEE*, vol. 12, no. 10, pp. 2941–2949, 2012.
- [11] H. Xu, L. Huang, C. Qiao, Y. Zhang, and Q. Sun, "Bandwidth-power aware cooperative multipath routing for wireless multimedia sensor networks," *IEEE Transactions on Wireless Communications*, vol. 11, no. 4, pp. 1532–1543, 2012.
- [12] R. Marjan, D. Behnam, A. B. Kamalrulnizam, A. R. Shukur, and N. M. Ali, "Interference-aware multipath routing protocol for QoS improvement in event-driven wireless sensor networks," *Tsinghua Science and Technology*, vol. 16, no. 5, pp. 475–490, 2011.
- [13] S. Li and Z. Wu, "Node-disjoint parallel multi-path routing in wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Software and Systems (ICES'05)*, pp. 210–215, Xi'an, China, December 2005.
- [14] K. Balakrishnan, J. Deng, and P. K. Varshney, "TWOACK: preventing selfishness in mobile ad hoc networks," in *Proceedings of the Wireless Communications & Networking Conference*, pp. 2137–2142, IEEE, New Orleans, LA, USA, April 2005.
- [15] B. Xiao, B. Yu, and C. Gao, "CHEMAS: i," *Journal of Parallel and Distributed Computing*, vol. 67, no. 11, pp. 1218–1230, 2007.
- [16] A. Liu, M. Dong, K. Ota, and J. Long, "PHACK: an efficient scheme for selective forwarding attack detection in WSNs," *Sensors*, vol. 15, no. 12, pp. 30942–30963, 2015.
- [17] H. Yang, X. Zhang, and F. Cheng, "A novel algorithm for improving malicious node detection effect in wireless sensor networks," *Mobile Networks And Applications*, vol. 2020, Article ID s11036-019-01492-4, 2020.
- [18] D. Xiao, J. Feng, and Q. Zhou, "Gauss reputation framework for sensor networks," *Journal on Communications*, vol. 29, no. 3, pp. 47–53, 2008.
- [19] G. Zheng, B. Gong, and Y. Zhang, "Dynamic network security mechanism based on trust management in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6667100, 10 pages, 2021.
- [20] H. Liao and S. Ding, "Mixed and continuous strategy monitor-forward game based selective forwarding solution in WSN," *International Journal of Distributed Sensor Networks*, vol. 2015, no. 11, 13 pages, Article ID 359780, 2015.
- [21] Z. Zhou and N. Shao, "An improved trust evaluation model based on Bayesian for WSNs," *Chinese Journal of Sensors and Actuators*, vol. 29, no. 6, pp. 927–933, 2016.
- [22] H. Zhou, Y. Wu, L. Feng, and D. Liu, "A security mechanism for cluster-based WSN against selective forwarding," *Sensors*, vol. 16, no. 9, pp. 1537–1552, 2016.
- [23] D. Silva, M. Martins, B. Rocha, A. Loureiro, L. B. Ruiz, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in *Proceedings of the 1st ACM international workshop on Quality of service*, pp. 16–22, Montreal, Canada, October 2005.
- [24] H. Liu, J. Cui, and H. Dai Hongjun, "Multivariate classification-based malicious node detection for wireless sensor network," *Chinese Journal of Sensors and Actuators*, vol. 24, no. 5, pp. 771–777, 2011.
- [25] W. She, Q. Liu, Z. Tian, J.-S. Chen, B. Wang, and W. Liu, "Blockchain trust model for malicious node detection in wireless sensor networks," *IEEE Access*, vol. 7, pp. 38947–38956, 2019.
- [26] Y. Li and J. C. S. Lui, "Identifying pollution attackers in network-coding enabled wireless mesh networks," in *Proceedings of the 2011 20th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–6, Maui, HI, USA, August 2011.
- [27] J. Hendricks, G. R. Ganger, and M. K. Reiter, "Verifying distributed erasure-coded data," in *Proceedings of 26th ACM Symposium on Principles of Distributed Computing*, pp. 1–8, Portland, OR, USA, August 2007.

Research Article

On Improving the Robustness of MEC with Big Data Analysis for Mobile Video Communication

Jianming Zhao ^{1,2,3}, Peng Zeng ^{1,2,3}, Yingjun Liu ⁴, and Tianyu Wang^{1,2,3}

¹State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China

²Key Laboratory of Networked Control Systems, Chinese Academy of Sciences, Shenyang 110016, China

³Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China

⁴Industry Development and Promotion Center,

Ministry of Industry and Information Technology of the People's Republic of China, Beijing 100846, China

Correspondence should be addressed to Peng Zeng; zp@sia.cn

Received 13 April 2021; Revised 18 May 2021; Accepted 8 June 2021; Published 7 July 2021

Academic Editor: Kai Wang

Copyright © 2021 Jianming Zhao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Mobile video communication and Internet of Things are playing a more and more important role in our daily life. Mobile Edge Computing (MEC), as the essential network architecture for the Internet, can significantly improve the quality of video streaming applications. The mobile devices transferring video flow are often exposed to hostile environment, where they would be damaged by different attackers. Accordingly, Mobile Edge Computing Network is often vulnerable under disruptions, against either natural disasters or human intentional attacks. Therefore, research on secure hub location in MEC, which could obviously enhance the robustness of the network, is highly invaluable. At present, most of the attacks encountered by edge nodes in MEC in the IoT are random attacks or random failures. According to network science, scale-free networks are more robust than the other types of network under the random failures. In this paper, an optimization algorithm is proposed to reorganize the structure of the network according to the amount of information transmitted between edge nodes. BA networks are more robust under random attacks, while WS networks behave better under human intentional attacks. Therefore, we change the structure of the network accordingly, when the attack type is different. Besides, in the MEC networks for mobile video communication, the capacity of each device and the size of the video data influence the structure significantly. The algorithm sufficiently takes the capability of edge nodes and the amount of the information between them into consideration. In robustness test, we set the number of network nodes to be 200 and 500 and increase the attack scale from 0% to 100% to observe the behaviours of the size of the giant component and the robustness calculated for each attack method. Evaluation results show that the proposed algorithm can significantly improve the robustness of the MEC networks and has good potential to be applied in real-world MEC systems.

1. Introduction

MEC is defined as providing IT service environment and cloud computing capability at the edge of mobile network [1–9]. In the view of the service providers, the network is actually divided into three parts: wireless access network, mobile core network, and application network. Among them, the wireless access network is composed of base stations, which are responsible for the access of mobile terminals [10–12]. The mobile core network is composed of a bunch of high-performance routers and servers, which are responsible for connecting the wireless base station to the

external network [13–15]. The application network is where all kinds of application servers work, in fact, all kinds of data centres, servers, and even PCs [16, 17]. The server providers are basically only in charge of the wireless access network and the mobile core network. The application network is usually in the hands of OTT. These three kinds of networks transfer data alternately between the user terminal and the application server to meet the various Internet needs of users. However, with the emergence of various new types of services, such as AR/VR, connected cars, and so on, this traditional network structure is gradually overburdened [18–21]. Therefore, the emergence of MEC, that is, network

services “sink” to the wireless access network side closer to users, brings about three benefits: (1) The transmission delay perceived by users is significantly reduced; (2) network congestion is controlled remarkably; and (3) more network information and network congestion control functions can be opened to developers.

There are many service scenarios for MEC. In the white paper “Mobile Edge Computing-A Key Technology towards 5G” of ETSI [22], the following typical scenarios are listed:

- (1) *Augmented Reality (AR)*. Augmented reality (AR) is a technology that uses additional information generated by computer to enhance or expand the real-world scene that users see. The MEC server caches the AR audio and video content that needs to be pushed. Based on the location technology and geographic location information, it corresponds to the way of one by one. According to the application request initiated by the terminal, MEC server judges the application content through deep packet analysis, determines to push AR content combined with location information, and sends it to the user. On the one hand, MEC solution reduces content delay and improves user experience through content localization; on the other hand, it greatly enhances the application effect and value of AR based on location.
- (2) *Intelligent Video Acceleration*. On the Internet, media and file transfer is usually in the form of stream or HTTP download based on TCP protocol. The change of channel environment, terminal access, and departure will lead to the change of link capacity. TCP may not be able to quickly adapt to the rapid changes of Ran, so using MEC for video acceleration can solve this kind of problem.
- (3) *Connected Cars*. MEC servers can be deployed on LTE base stations along the road, receiving and analyzing local information from on-board applications and road sensors, so as to transmit some emergency information to other vehicles in the region.
- (4) *Convergence Gateway of Internet of Things*. IOT devices are usually resource constrained in terms of processor and memory capacity, so it is necessary to use aggregation gateway to aggregate all kinds of IOT device information, which can reduce the response time of analysis and processing.

During the process of transferring video flow, mobile devices could be easily attacked, which would seriously influence the function of the whole systems. In this paper, in order to improve the robustness of MEC network for mobile video communication, we proposed a novel method for hub location to generate a more robust structure of the network. The MEC networks discussed in this paper include the edges nodes for edge computation and the information transferred between the nodes. To optimize the structure of the network, we use an optimization algorithm to overall consider the capability of edge nodes and the amount of the information.

The main contributions of this work are summarized as follows:

- (1) We address the problem of improving the robustness of MEC networks. We can show that MEC networks with different structures perform significantly uniquely under the same attacks.
- (2) We propose a well-tuned optimization algorithm to improve the robustness of MEC networks.

The rest of this paper is organized as follows. Section 2 provides the definition of the robustness of the MEC network, and the relationship between the robustness and the structure. In Section 3, we present the optimization algorithm, which can improve the robustness of the MEC networks. Section 4 shows the performance of our algorithm, we evaluate the robustness of each generated network under different kinds of attacks, and the relative size of the giant component and the robustness under different attacks are reported in this part. Finally, conclusions are given in Section 5.

2. Background

2.1. Definition of Network Robustness. In this paper, we discuss the problem of improving the robustness of an MEC network. Connectivity is one of the most popular characteristics of network structure and function, and the size of giant component in the network is used to evaluate the connectivity. Therefore, the robustness could be evaluated by the size of the giant component after attacks. The same network shows different robustness under different attacks. In paper [23], the robustness of a network under certain attacks is defined as follows:

$$R = \frac{1}{|N|} \sum_{Q=1}^N GCsize(Q), \quad (1)$$

where $|N|$ is the number of nodes in the network, Q is the attack on the network, and $GCsize(Q)$ means the size of the giant component under the attack of Q . Figure 1 shows the effect of two different attacks on the same example network. Figure 1(a) is the example network, and Figures 1(b-c) are under two different attacks of one node in the network. As we can see, the attack in the middle subfigure results in the size of the network of only one node, and the attack in the right subfigure leads to the size of the network of four nodes. This reflects the fact that the same network will show different robustness under different attacks of the same scale. In MEC networks, the size of the giant component means the subset of the network, where the information could reach each edge node. The parameter “ R ” displays the accumulated influence of the attacks on the MEC network, in terms of the giant component. As known by intuition, the area with the function of information transportation can effectively evaluate the robustness of the MEC network. In the basis of the analysis, the parameter R would be used for the evaluation of our method to improve the robustness of the MEC

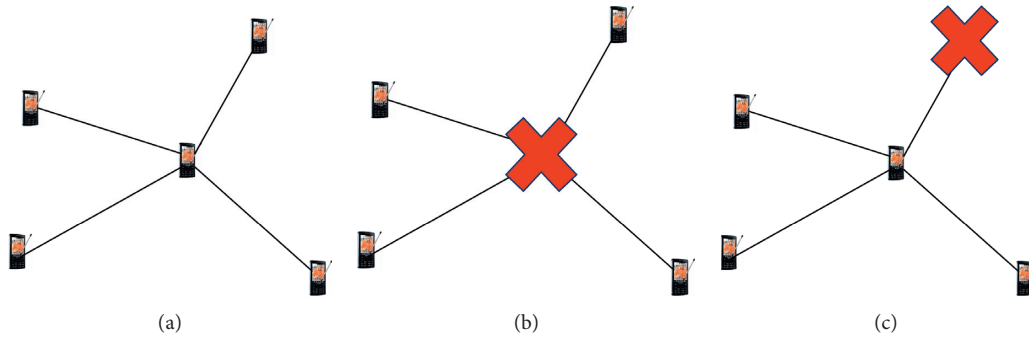


FIGURE 1: Different attacks on an example network. The example network and two situations with different attacks on the network.

networks. Therefore, designing a robust network according to the potential attacks is an important work.

2.2. Popular Attack Method. In MEC networks for mobile video communication, the mobile edge nodes are exposed to an open environment, which could cause random damages to the nodes [24]. Therefore, the first attack method mentioned here is random attack, in which the nodes would be damaged randomly.

One of the simplest metrics in complex network theory is the degree. The degree of a node i (deg_i) is the number of the node i 's links and is defined in terms of the adjacency matrix A as [25]

$$\text{deg}_i = \sum_{j \in N} a_{ij}. \quad (2)$$

The adjacency matrix storage structure uses a one-dimensional array to store the edge information for each vertex, so that all the points together represent the adjacency relationship between the vertices in the graph with a matrix. A matrix is actually a two-dimensional array. As a method to attack networks, degree-based methods first attack nodes with larger degree. In real MEC networks, the edge nodes with larger degree are often connected to more other nodes, which reflect the importance of the nodes to some degree. The degree-based methods are significantly efficient even for very large-scale networks.

Betweenness is another very popular network metric. The information exchange between two nodes that are not directly adjacent depends on the nodes on the path connecting these two nodes. The betweenness of a node can describe the importance in terms of exchanging information, and the betweenness of node i is defined as [26, 27]

$$b_i = \sum_{x, y \in N, x \neq y} \frac{n_{xy}(i)}{n_{xy}}, \quad (3)$$

where n_{xy} means the number of the shortest paths between node x and y and $n_{xy}(i)$ is the number of the shortest paths between node x and y through node i . Similarly to degree-based methods, in betweenness-based methods, the nodes with larger betweenness are attacked first. The ones with larger betweenness would be more important for information transforming in real MEC networks. Betweenness-

based methods are used more frequently in real-world applications, since they could often identify more important nodes than degree-based methods.

Based on the attack methods above, we evaluate the methods on a small example network with only thirteen nodes. Figure 2 shows the networks under different attacks with two nodes. As we can see, after the optimal attack on the example network, the size of the giant component would be 5; the size of the giant component under degree-based method is 7; and the size of the giant component under betweenness-based method is 7. To sum up, it can be seen that different attack methods lead to different effectiveness of the attack. Therefore, we will evaluate the networks designed in our paper under different attack methods.

3. Proposed Algorithm

Given the vulnerability of the MEC network under attacks, we proposed to improve the robustness of MEC for mobile video communication with big data. Since the mobile devices need to send and receive video data, the capacity of each device and the size of the video data should be taken into consideration in the algorithm. Generally, the structure and function of MEC network should be related to two aspects: the video flow transferred in the network and the type of attacks that the network is facing with.

In this paper, we define the MEC network to be $G(E, V)$, where E means the mobile device nodes in the network and V means the links in the network. If there is a link between two nodes, this means there is a video flow transferring between them. The attacks used in the paper are also classified into two aspects: random attack and human intentional attacks. When the MEC network is under random attack, the mobile devices will break down randomly, maybe due to hardware damage or being out of power. When it comes to human intentional attacks, the attackers would aim at some important nodes, which would influence the network function and structure seriously. We use some popular intentional attack methods here to test the effectiveness of our algorithm. However, different network structure could show different robustness under the same type of attacks, and the same network could behave differently under different types of attacks. Therefore, in this paper, the structure of the MEC network can adapt to the change of attack types at any time. Firstly, two typical kinds of complex networks

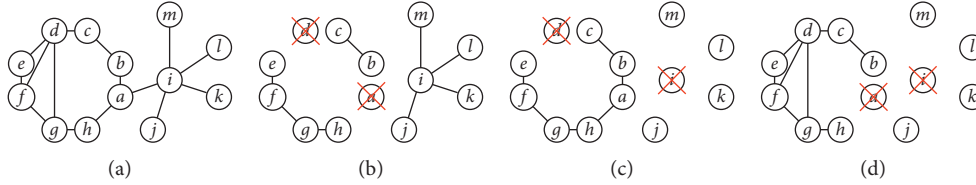


FIGURE 2: The effectiveness of different attack methods. (a) Original network. (b) Optical method (5). (c) Degree-based method (7). (d) Betweenness-based method (7).

with different structure would be introduced for the use of our algorithm.

- (1) *BA Scale-Free Network* [28, 29]. In this model, the construction of BA networks consists of two steps:
 - (1) Starting with a connected network with m_0 nodes, a new node is generated in the network each time, which would build m ($m \leq m_0$) links with the existing nodes
 - (2) When a new node is selecting an existing node to build a link, the probability of its connection with node i is $p_i = k_i / \sum_j k_j$

After time t , the above steps will generate a network with $N = t + m_0$ nodes. In this paper, we set the parameters $N = 200$ and $N = 500$, m is 2 or 3 or 4, which means the scale of the MEC network.

- (2) *WS Small-World Network* [30]. In this model, the construction of WS networks also consists of two steps:
 - (1) Starting from a regular network: consider a network with N nodes, which are surrounded by a ring, in which each node is connected to its left and right $k/2$ nodes.
 - (2) Rewiring randomly: each link in the network is randomly rewired with probability p , that is, one endpoint of the link remains unchanged, and the other endpoint is taken as a randomly selected node in the network. It stipulates that any two different nodes can only have one link at most, and each node cannot have a link connected to itself.

In the WS network model, it is a regular network when $p = 0$, and it is a random network when $p = 1$. The characteristic of the network could be controlled through the variety of parameter p .

According to the knowledge of network science, BA networks are more robust under random attacks, while WS networks behave better under human intentional attacks. Therefore, we should change the structure of the network accordingly, when the attack types is different. Besides, in the MEC networks for mobile video communication, the capacity of each device and the size of the video data influence the structure significantly. When the capacity of a node is used up, there could be no more video data it can deal with.

We show the process of our algorithm in Figure 3 and the pseudocode of our algorithm in Algorithm 1. Our

algorithm consists of three building blocks: (1) collect information of nodes and attacks, (2) generate networks, and (3) test the effectiveness of the network. Each step is discussed in detail as follows:

Block 1: Collect Information of Nodes and Attacks. First, collect the information of the nodes' capacity and the video flow, since this information would directly influence the structure and function of the networks. Then, as discussed before, since the robustness of the same network under different types of attacks is unique, the types of the attack need to be analyzed before constructing the network.

Block 2: Generate Networks. In network science, BA networks and WS networks are two typical kinds of networks, and they behave differently under the same attacks. Therefore, we decide to generate BA or WS networks according to the information of the MEC. When generating the networks, the capacity and the video flow need to be concerned. This means that if one node total is beyond its capacity, it will not be connected with new nodes.

Block 3: Test the Effectiveness of the Network. After generating networks, we need to use popular methods to attack them and to see if the robustness is good enough. If not, we would improve the parameter in the process of generating networks.

4. Results and Discussion

In this section, we evaluate the effectiveness of our algorithms on some artificial MEC networks with 200 nodes ($n = 200$) and 500 nodes ($n = 500$). Besides, the capacity of each node is 200 or 500 in this section, and the video flow to be transferred on each node varies from 1 to 4. To show the performance of our algorithm, we evaluate the robustness of each generated network under different kinds of attacks. The relative size of the giant component and the robustness under different attacks are also reported in this part.

4.1. Random Network Generation. In this section, we would generate the MEC networks for mobile video communication under different types of attacks. As introduced above, different types of networks behaves unique under different types of attacks: the BA network is more robust under random attacks, and the WS network behaves better under intentional attacks.

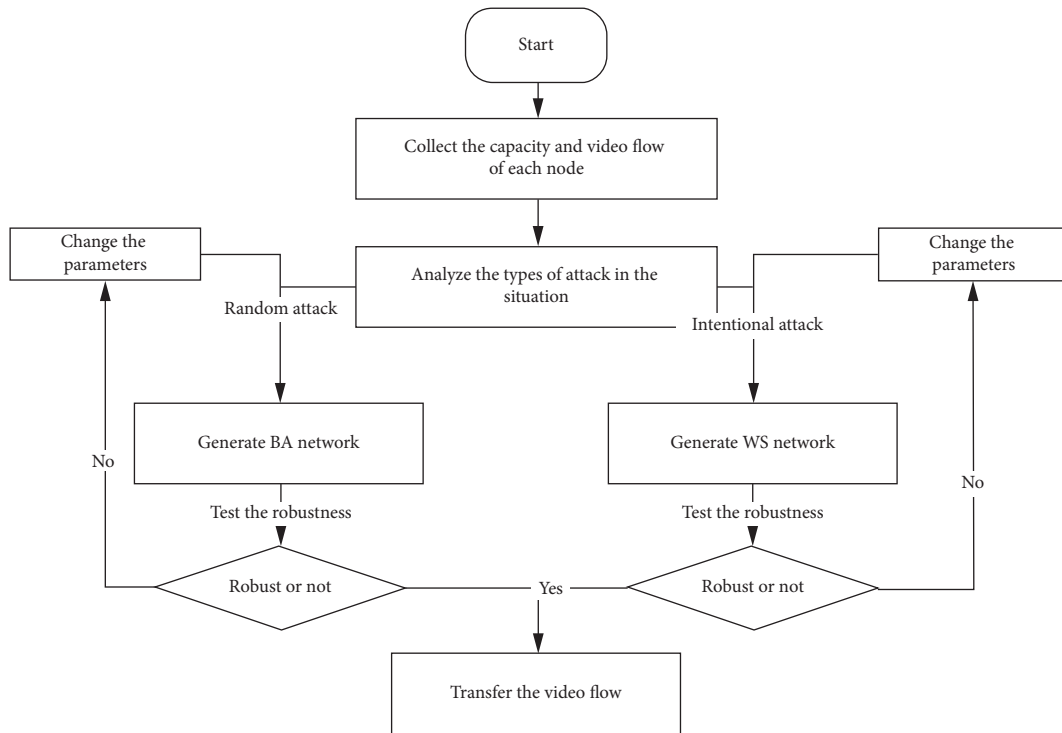


FIGURE 3: The process of our algorithm.

```

Robust MEC algorithm;
Collect the capacity and video flow of each node;
Analysis the types of attacks in the situation of mobile video communication;
while the robustness of the networks is not good enough
  Change the parameter of generating network method
  if the attacks are random attacks then:
    Generate BA networks
    for each node do
      if sum of neighbours' video flow is larger than the node's capacity then
        regenerate the network
    else if the attacks are intentional attacks then:
      Generate WS networks
      for each node do
        if sum of neighbours' video flow is larger than the node's capacity then
          regenerate the network
  Test the robustness under different attacks.
Return the generated networks.
    
```

ALGORITHM 1: The robust MEC algorithm.

4.1.1. *Random Attacks.* Firstly, for random attacks, we would use our algorithm to generate BA networks. In this paper, we set the number of nodes in the network to be 200 and 500, and the parameter m to be 2, 3, and 4. Table 1 and Figure 4 show the details of the generated networks.

Table 1 displays basic statistics for the three BA networks generated with our algorithm. In Table 1, “ave. deg” and “ave. betw” mean the average degree and average

betweenness of all nodes, respectively; ASPL stands for the average of the distance between all node pairs in the networks; CC represents the clustering coefficient of the networks, which can show the degree of nodes being in the same cluster; the nodes in the same community usually have the same characteristics. As we can see from the table, though the numbers of the nodes in each network are the same, the structures of these networks are rather diverse. With the

TABLE 1: Basic statistics for the BA networks used in our study.

| Metrics | 200BA (2) | 200BA (3) | 200BA (4) |
|-------------|-----------|-----------|-----------|
| Node number | 200 | 200 | 200 |
| Link number | 400 | 600 | 800 |
| Ave. deg | 3.95 | 5.9 | 7.83 |
| Ave. betw | 0.012 | 0.0097 | 0.0082 |
| ASPL | 3.38 | 2.91 | 2.61 |
| CC | 0.10 | 0.07 | 0.12 |
| Community | 12 | 10 | 10 |
| Metrics | 500BA (2) | 500BA (3) | 500BA (4) |
| Node number | 500 | 500 | 500 |
| Link number | 1000 | 1500 | 2000 |
| Ave. deg | 3.984 | 5.964 | 7.936 |
| Ave. betw | 0.0058 | 0.0044 | 0.0039 |
| ASPL | 3.91 | 3.18 | 2.93 |
| CC | 0.048 | 0.058 | 0.067 |
| Community | 16 | 13 | 11 |

Ave. deg means average degree; ave. betw is average betweenness; ASPL stands for the average shortest path length; and CC represents clustering coefficient.

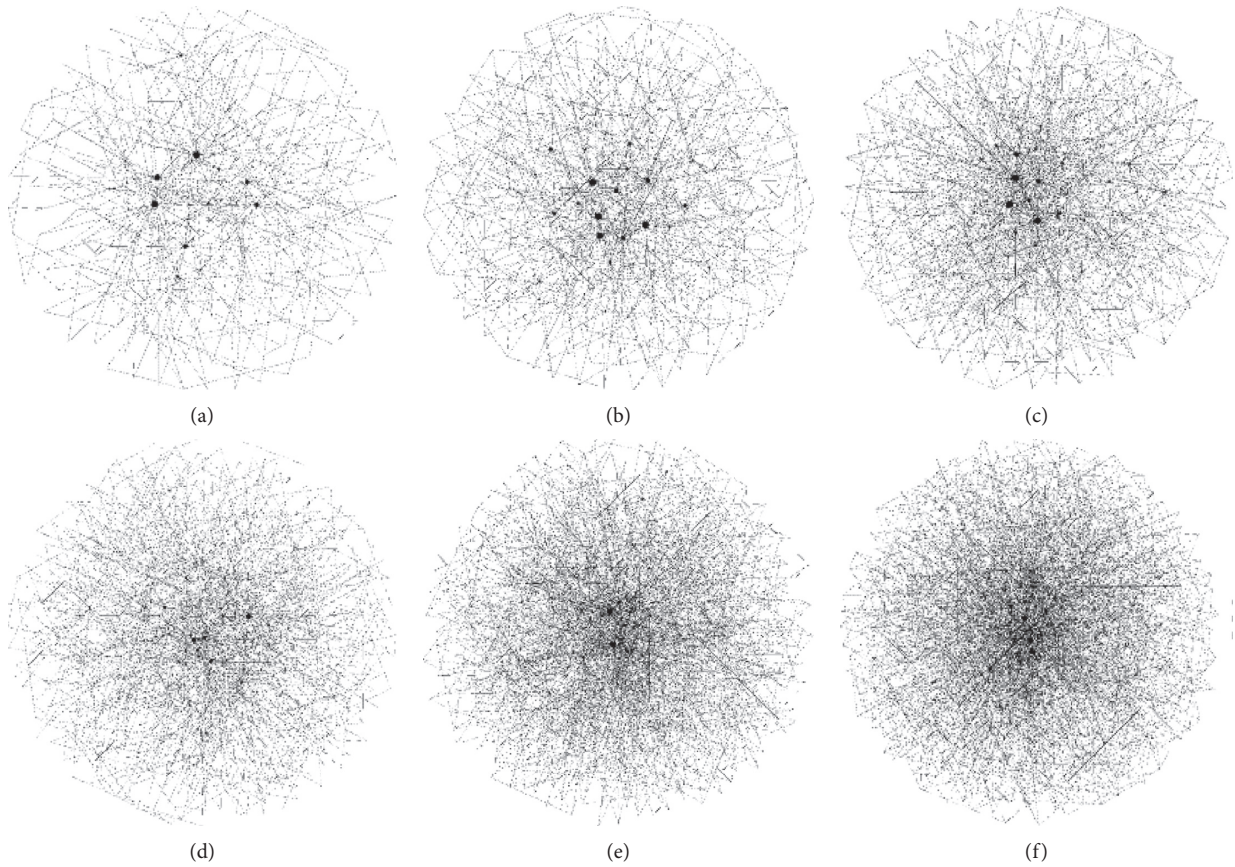


FIGURE 4: Structure of the BA networks.

increase of the link number, the average betweenness and the ASPL would be smaller. The community structure is detected by Louvain method in this paper.

Figure 4 visualizes the six BA networks, where the size of a node is proportional to its degree. The hub nodes are obvious in the network. As seen from the figure, the parameter in the algorithm can significantly influence the structure of the network.

4.1.2. Intentional Attacks. For intentional attacks, our algorithm would generate a series of WS networks. In WS network model, there are three parameters: number of nodes, initial number of each node's neighbours, and probability of each node rewiring. In order to evaluate the effectiveness of our algorithm in experiment with different parameters, we set the number of nodes to be 200 and 500, initial number of each node's neighbours to be 4, and probability of each node rewiring to be 20%, 30%, and 40%.

Table 2 shows basic statistics for the three WS networks generated with our algorithm. As we can see from the table, though the numbers of the nodes in each network are the same, with the increase of the rewiring probability, the average betweenness and the ASPL would be smaller. The difference of WS networks is less than BA networks.

Figure 5 visualizes the six WS networks, where the size of a node is proportional to its degree. The structure of these three WS networks looks similar, and there is no obvious hub node.

4.2. Robustness Test. To test the robustness of the networks, we will attack the networks with different attack methods. The attack methods used in this paper is random attack, degree-based attack, and betweenness-based attack. Specially, for the random attack, we attack the network randomly for ten times and choose the one with the best effectiveness. For intentional attacks, we attack the networks with two most popular methods: degree-based and betweenness-based methods.

Figure 6 and Table 3 report the size of the giant component and the robustness for the six BA networks. In the evaluation process, we increase the attack scale from 0% to 100% to observe the behaviours of the size of the giant component and the robustness calculated for each attack method.

In Figure 6, as we can see, on the whole ranges from 0% to 100% of the whole network, the generated MEC networks (BA networks) are more robust under random attacks. At the beginning of the attacks, the intentional attack methods (degree-based and betweenness-based methods) can rapidly influence the structure of the networks. For example, for BA (2) network, the size of the giant component is nearly 0 when 17% nodes are attacked, while the random attacks could hardly influence the structure of the network. Through comparing the effectiveness on different BA networks, it can

TABLE 2: Basic statistics for the WS networks used in our study.

| Metrics | 200WS (0.2) | 200WS (0.3) | 200WS (0.4) |
|-------------|-------------|-------------|-------------|
| Node number | 200 | 200 | 200 |
| Link number | 800 | 800 | 800 |
| Ave. deg | 3.99 | 3.99 | 3.99 |
| Ave. betw | 0.021 | 0.018 | 0.017 |
| ASPL | 5.09 | 4.47 | 4.28 |
| CC | 0.28 | 0.13 | 0.12 |
| Metrics | 500WS (0.2) | 500WS (0.3) | 500WS (0.4) |
| Node number | 500 | 500 | 500 |
| Link number | 2000 | 2000 | 2000 |
| Ave. deg | 3.99 | 3.99 | 3.99 |
| Ave. betw | 0.011 | 0.0090 | 0.0083 |
| ASPL | 6.24 | 5.47 | 5.16 |
| CC | 0.26 | 0.18 | 0.10 |
| Community | 22 | 19 | 20 |

Ave. deg means average degree; ave. betw is average betweenness; ASPL stands for the average shortest path length; and CC represents clustering coefficient.

be concluded that as the parameter of BA network increases, the networks would show better robustness.

When it comes to Table 3, the robustness of generated MEC networks (BA networks) against random attacks and two intentional attacks is shown. In all generated networks, the robustness against random attacks is the largest while that against degree-based attacks is the smallest. It should be noticed that since hub nodes are the most important characteristics of BA networks, the hub-attack method (degree-based attack) is the most effective.

Figure 7 and Table 4 report the size of the giant component and the robustness for the six WS networks. In the evaluation process, we also increase the attack scale from 0% to 100% to observe the behaviours of the size of the giant component and the robustness calculated for each attack method.

For Figure 7, when the attack range is smaller than 30%, there is no obvious difference between random and intentional attacks, which reflects the fact that the generated MEC networks are relatively robust under intentional attacks. Comparing with Figure 6, WS networks show better robustness than BA networks, since these WS networks have more links than most of BA networks.

In Table 4, similarly to the results on BA networks, the WS networks are more robust under random networks. However, it should be noticed that the difference between random attacks and intentional attacks is rather small. This means that the MEC networks (WS networks) generated for intentional attacks show good robustness against intentional networks. Compared with BA4, which contains the same number of nodes and links, WS networks' robustness has been improved a lot. WS40 network's robustness is nearly 21.1% larger than BA networks. The results show that our algorithm could generate robust MEC networks for mobile video communication under both random attacks and intentional attacks.

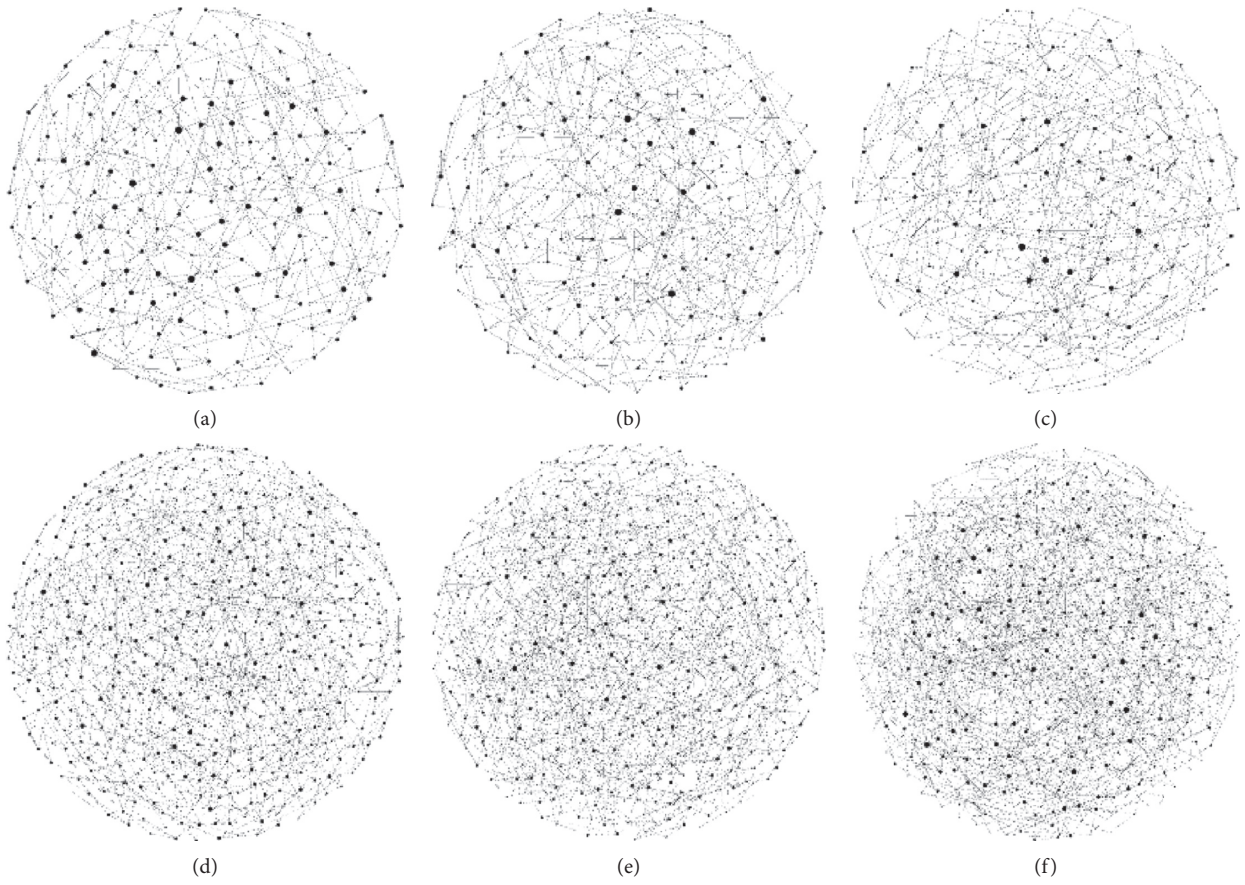


FIGURE 5: Structure of the WS networks.

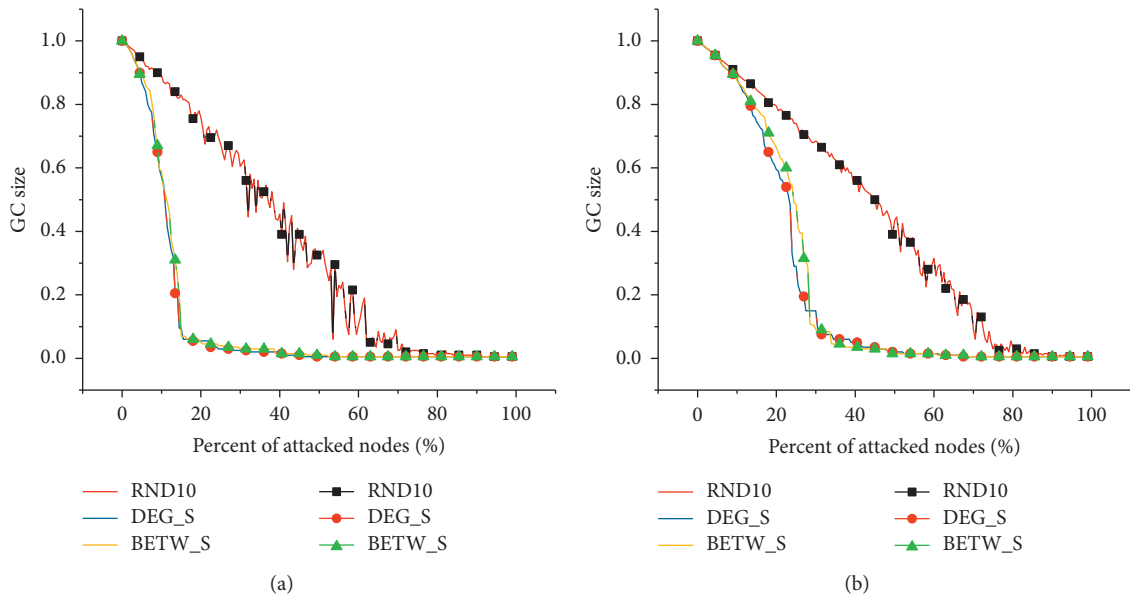


FIGURE 6: Continued.

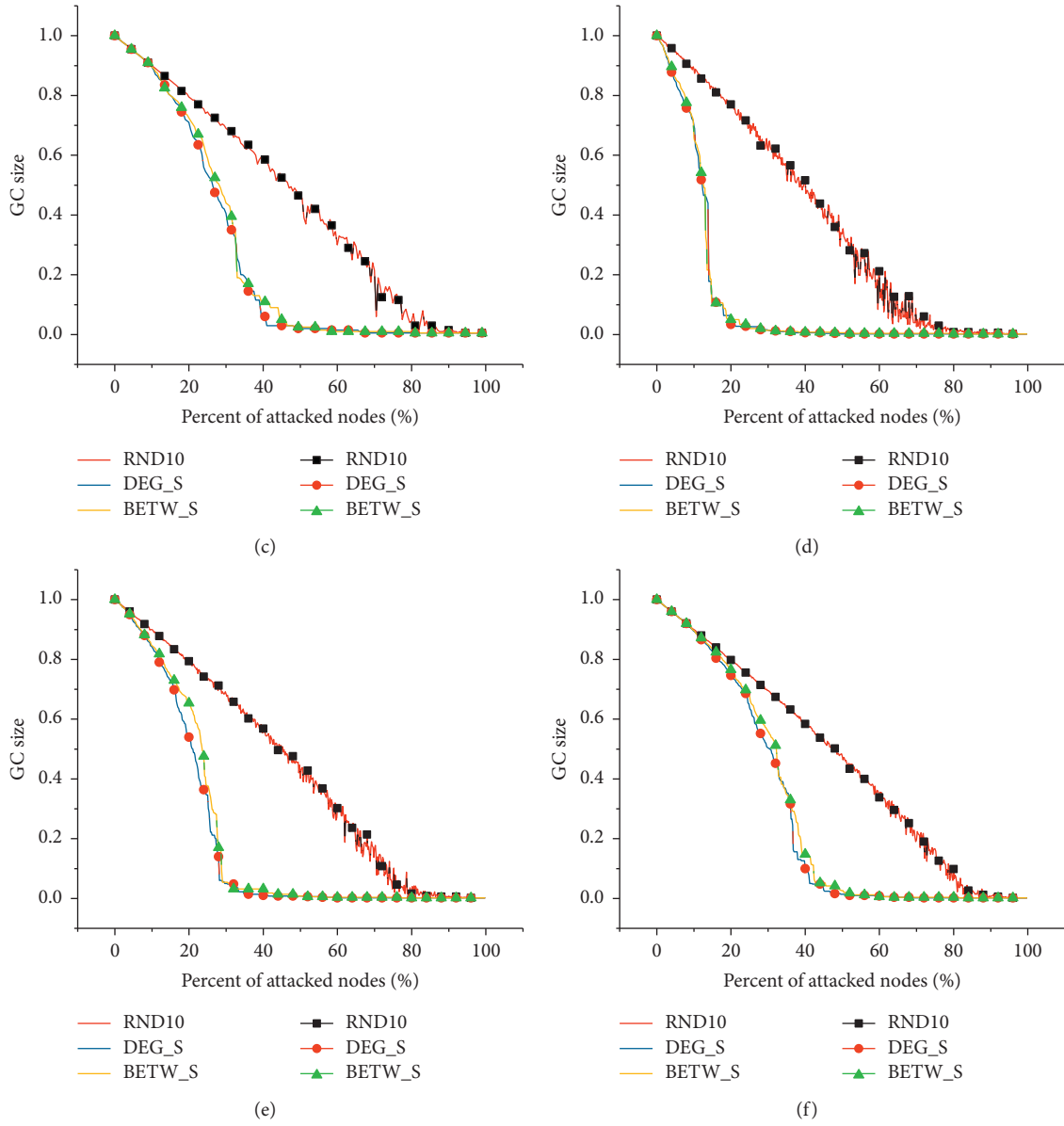


FIGURE 6: Results of the BA networks.

TABLE 3: Robustness of BA networks under different attacks.

| Method | 200BA2 | 200BA3 | 200BA4 |
|--------|--------|--------|--------|
| RND | 0.367 | 0.431 | 0.454 |
| Deg | 0.116 | 0.219 | 0.261 |
| Betw | 0.122 | 0.230 | 0.270 |
| Method | 500BA2 | 500BA3 | 500BA4 |
| RND | 0.383 | 0.433 | 0.460 |
| Deg | 0.118 | 0.198 | 0.279 |
| Betw | 0.120 | 0.214 | 0.291 |

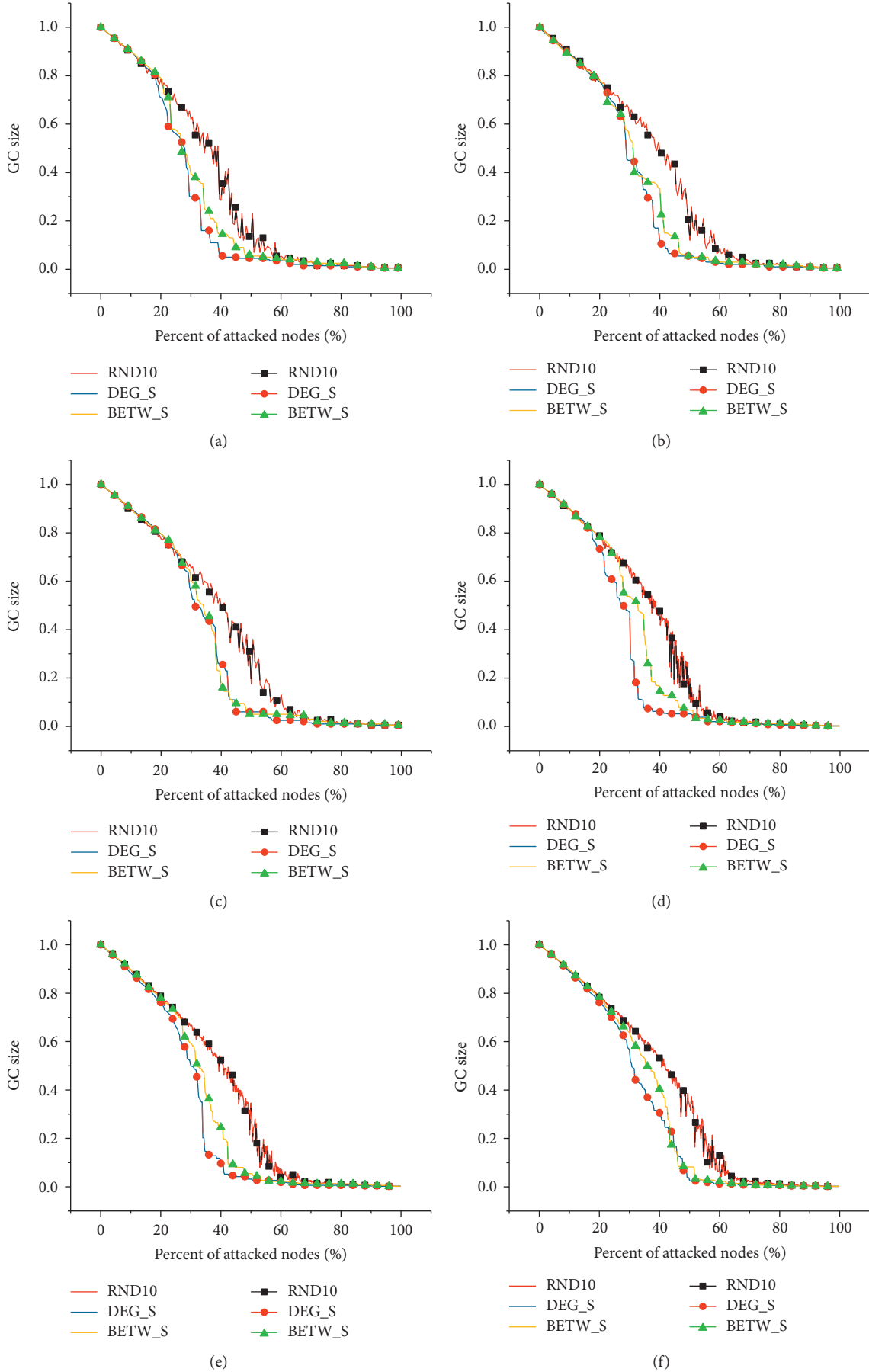


FIGURE 7: Results of the WS networks.

TABLE 4: Robustness of WS networks under different attacks.

| Method | 200WS20 | 200WS30 | 200WS40 |
|--------|---------|---------|---------|
| RND | 0.343 | 0.372 | 0.378 |
| Deg | 0.268 | 0.294 | 0.316 |
| Betw | 0.292 | 0.310 | 0.322 |
| Method | 500WS20 | 500WS30 | 500WS40 |
| RND | 0.350 | 0.372 | 0.383 |
| Deg | 0.261 | 0.278 | 0.308 |
| Betw | 0.302 | 0.308 | 0.329 |

5. Conclusions

In this paper, we designed an MEC network generating algorithm for mobile video communication to improve the security and robustness. After evaluation on artificial networks, our algorithm could significantly improve the robustness of the networks under either type of attacks (random attacks or human intentional attacks). Our algorithm provides a view to improving the security of mobile video communication and robustness of MEC networks.

In future work, our algorithm can also be improved in the following ways:

- (1) Design an algorithm for more types of attacks. For the current algorithm, we only concern on two types of attacks: random or intentional. However, in real-world MEC systems, the attacks may be a mixture of the two types. Therefore, it is necessary to make the algorithm applicable for more situations.
- (2) Collect real-world data. In this paper, we evaluate our algorithm on some artificial networks, and it shows effectiveness. However, in order to prove that our algorithm can be applied to real-world MEC networks, it is necessary to collect some real-world data for experiments.

Data Availability

In this manuscript, the data used to support the findings of this study are simulation data and generated by the NetworkX library in Python. Actually, we have sketched the basic technological process in this manuscript, but some contents and specific parameters of this process may be not completely open details. Therefore, if other researchers want to verify the results, replicate the analysis, or conduct secondary analyses, the corresponding author or first author can be contacted. The requests for the data will be considered by them after a confidentiality agreement.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by the National Key R&D Program of China: “Key Technologies for Information Security of Measurement and Control Equipment” (Grant no.

2018YFB2004200) and the 2020 Industrial Internet Innovation and Development Project: “MEC Network Security Protection Technology and Product.”



References

- [1] S. Wang, X. Zhang, Y. Zhang, and L. Wang, “A survey on mobile edge networks: convergence of computing, caching and communications,” *IEEE Access*, vol. 99, p. 1, 2017.
- [2] H. Hu, H. Shan, C. Wang et al., “Video surveillance on mobile edge networks-A reinforcement-learning-based approach,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4746–4760, 2020.
- [3] K. Peng, C. Victor, M. Leung, X. Xu, L. Zheng, and J. Wang, “A survey on mobile edge computing: focusing on service adoption and provision,” *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 8267838, 16 pages, 2018.
- [4] X. Zhang, W. Wu, S. Yang, and X. Wang, “Falcon: a blockchain-based edge service migration framework in MEC,” *Mobile Information Systems*, vol. 2020, Article ID 8820507, 17 pages, 2020.
- [5] M. Wan, J. Li, Y. Liu, J. Zhao, and J. Wang, “Characteristic insights on industrial cyber security and popular defense mechanisms,” *China Communications*, vol. 18, no. 1, pp. 130–150, 2021.
- [6] Y. Gong and Y. Mo, “Qualitative analysis of commercial services in MEC as phased-mission systems,” *Security and Communication Networks*, vol. 2020, Article ID 8823952, 11 pages, 2020.
- [7] M. Cui, Y. Fei, and Y. Liu, “A survey on secure deployment of mobile services in edge computing,” *Security and Communication Networks*, vol. 2021, Article ID 8846239, 8 pages, 2021.
- [8] Q. Cao, Q. Wu, B. Liu, S. Zhang, and Y. Zhang, “An optimization method for mobile edge service migration in cyberphysical power system,” *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6610654, 12 pages, 2021.
- [9] Y. Mao, C. You, J. Zhang, and K. Huang, “A survey on mobile edge computing: the communication perspective,” *IEEE Communications Surveys & Tutorials*, vol. 99, p. 1, 2017.
- [10] H. Kim, M. Bae, W. Lee, and H. Kim, “Adaptive decision of wireless access network for higher user satisfaction,” *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 3427238, 19 pages, 2018.
- [11] J. Huang, V. Subramanian, R. Agrawal, and R. Berry, “Joint scheduling and resource allocation in uplink OFDM systems for broadband wireless access networks,” *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 2, pp. 226–234, 2009.
- [12] D. Tipper, T. Dahlberg, H. Shin, and C. Charnsripinyo, “Providing fault tolerance in wireless access networks,” *IEEE Communications Magazine*, vol. 40, no. 1, pp. 58–64, 2002.
- [13] A. D. Zayas, B. G. Garcia, and P. Merino, “An end-to-end automation framework for mobile network testbeds,” *Mobile Information Systems*, vol. 2019, Article ID 2563917, 8 pages, 2019.
- [14] J. He and W. Song, “Smart routing: fine-grained stall management of video streams in mobile core networks,” *Computer Networks*, vol. 85, pp. 51–62, 2015.
- [15] S. Racz, M. Telek, and G. Fodor, “Call level performance analysis of 3rd generation mobile core networks,” in

- Proceedings of the IEEE International Conference on Communications*, IEEE, Glasgow, Scotland, 2007.
- [16] T. Szyrkowiec, A. Autenrieth, and W. Kellerer, "Optical network models and their application to software-defined network management," *International Journal of Optics*, vol. 2017, Article ID 5150219, 9 pages, 2017.
 - [17] E. Lakiotakis, C. Liaskos, and X. Dimitropoulos, "Application-network collaboration using SDN for ultra-low delay teleorchestras," in *Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, Heraklion, Greece, July 2017.
 - [18] B. Priya, R. Sri, A. Nimmagadda, and K. Garudkar, "Mobile edge communication an overview of MEC in 5G," in *Proceedings of the 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pp. 271–276, IEEE, Coimbatore, India, March 2019.
 - [19] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.
 - [20] Y. Yu, "Mobile edge computing towards 5G: vision, recent progress, and open challenges," *China Communications*, vol. 13, no. Supplement2, pp. 89–99, 2016.
 - [21] E. Garcia-Palacios, "Mobile edge computing towards 5G: vision, recent progress, and open challenges," *China Communications*, vol. 13, no. Supplement2, pp. 89–99, 2016.
 - [22] ETSI, *Mobile Edge Computing-A Key Technology Towards 5G*, Vol. 11, ETSI White Pap, Sophia Antipolis, France, 2015.
 - [23] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. Herrmann, "Mitigation of malicious attacks on networks," *Proceedings of the National Academy of Sciences*, vol. 108, no. 10, pp. 3838–3841, 2011.
 - [24] Q. Li, S. Meng, S. Zhang, J. Hou, and L. Qi, "Complex attack linkage decision-making in edge computing networks," *IEEE Access*, vol. 7, no. 99, pp. 12058–12072, 2019.
 - [25] T. Opsahl, F. Agneessens, and J. Skvoretz, "Node centrality in weighted networks: generalizing degree and shortest paths," *Social Networks*, vol. 32, no. 3, pp. 245–251, 2010.
 - [26] P. Holme, B. J. Kim, C. N. Yoon, and S. K Han, "Attack vulnerability of complex networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 65, no. 5, Article ID 056109, 2002.
 - [27] D. Ding, J. Cao, Overview of network security of cyber-physical systems," *Information and Control*, vol. 48, no. 5, pp. 513–521, 2019.
 - [28] Y.-y. Zhu, W. Li, and X. Cai, "Opinion evolution on a BA scaling network," *Physica A: Statistical Mechanics and Its Applications*, vol. 392, no. 24, pp. 6596–6602, 2013.
 - [29] R. Yin, F. Zhang, Y. Xu, L. Liu, and X. Li, "A security routing algorithm against selective forwarding attacks in scale-free networks," *Procedia Computer Science*, vol. 174, pp. 543–548, 2020.
 - [30] V. Latora and M. Marchiori, "Efficient behavior of small-world networks," *Physical Review Letters*, vol. 87, no. 19, Article ID 198701, 2001.

Research Article

V-Lattice: A Lightweight Blockchain Architecture Based on DAG-Lattice Structure for Vehicular Ad Hoc Networks

Xiaodong Zhang ^{1,2}, Ru Li ^{1,2}, Wenhan Hou,^{1,2} and Hui Zhao^{1,2}

¹Inner Mongolia Key Laboratory of Wireless Networking and Mobile Computing, Hohhot 010021, China

²College of Computer Science, Inner Mongolia University, Hohhot 010021, China

Correspondence should be addressed to Ru Li; csliru@imu.edu.cn

Received 28 March 2021; Revised 19 April 2021; Accepted 7 May 2021; Published 30 May 2021

Academic Editor: Yuanlong Cao

Copyright © 2021 Xiaodong Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the development of wireless communication technology and the automobile industry, the Vehicular Ad Hoc Networks bring many conveniences to humans in terms of safety and entertainment. In the process of communication between the nodes, security problems are the main concerns. Blockchain is a decentralized distributed technology used in nonsecure environments. Using blockchain technology in the VANETs can solve the security problems. However, the characteristics of highly dynamic and resource-constrained VANETs make the traditional chain blockchain system not suitable for actual VANETs scenarios. Therefore, this paper proposes a lightweight blockchain architecture using DAG-lattice structure for VANETs, called V-Lattice. In V-Lattice, each node (vehicle or roadside unit) has its own account chain. The transactions they generated can be added to the blockchain asynchronously and parallelly, and resource-constrained vehicles can store the pruned blockchain and execute blockchain related operations normally. At the same time, in order to encourage more nodes to participate in the blockchain, a reputation-based incentive mechanism is introduced in V-Lattice. This paper uses Colored Petri Nets to verify the security of the architecture and verifies the feasibility of PoW anti-spam through experiment. The validation results show that the architecture proposed in this paper is security, and it is feasible to prevent nodes from generating malicious behaviors by using PoW anti-spam.

1. Introduction

With the development of wireless communication technology and the automobile industry, the Vehicular Ad Hoc Networks (VANETs) have developed significantly, which brings many conveniences to humans in terms of safety and entertainment. Through the collaboration of Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrians (V2P), and Vehicle-to-Cloud (V2C) communication, VANETs can enhance driving safety and traffic efficiency and realize Intelligent Traffic System (ITS) better. In ITS, a lot of sensor data need to be transferred between nodes. In particular, video data collected by cameras, such as traffic accident information and road condition information, need to be transmitted to other nodes through mobile video communication technology. However, in the process of

communication between the nodes in the VANETs, security and privacy are the main concerns [1], including the security protection issues (such as integrity and correctness) of the VANETs data in transmission process, the security issues (such as non-tampering) of VANETs data stored in the data center, the access control issues (such as identity verification), and privacy protection issues.

The traditional approach to solve security problems is based on centralized approach which requires a trusted central entity. However, through a trusted central entity, there is a single point of failure problem. Moreover, the data stored in a trusted central do not have a technical method to ensure its security (nontampering, traceability), and they are only guaranteed at the legal level. There are also some approaches based on the distributed approach, but the lack of trust between distributed entities makes it difficult to implement in actual environments.

Blockchain is a distributed technology. It uses cryptography and hash functions to store data in a chain to ensure that data are tamper-resistant and traceable. And the technology uses a consensus protocol to ensure data consistency. At present, the use of blockchain is very extensive, including supply chain management, agriculture, Internet of Things (IoT), artificial intelligence (AI), and autonomous vehicles. It can be seen that using blockchain technology in the VANETs environment can solve the security problems and the dependence on trusted central entities.

At present, there have been many studies on using blockchain in the VANETs environment, mainly including the architecture [1–4], authentication mechanism [5–7], privacy protection [8–10], trust management [11–13], certificate management [14, 15], and data sharing [16–18]. These studies mainly focus on the architecture of the combination of blockchain and the VANETs and how to use the blockchain to solve the security issues in the VANETs environment. Moreover, they are all based on the traditional chain blockchain structure. However, the VANETs is self-organizing and highly dynamic. Moving vehicles will gather at intersections to form a multidomain network. The data sharing between V2V and V2I is mostly temporary and dynamic, which make it difficult to synchronize data between nodes and resulting in more forks and low consensus efficiency. At the same time, vehicles in the VANETs usually have limited resources. The compute and storage capabilities are usually not particularly strong. So, a complete blockchain cannot be run and stored on the vehicle. It can be seen that the characteristics of highly dynamic and resource-constrained of the VANETs make the traditional chain blockchain system not suitable for actual VANETs scenarios.

With the development of blockchain technology, the structure of the blockchain has also evolved from a traditional chain structure to a directed acyclic graph (DAG) structure. In the DAG structure, the constituent unit is a transaction which does not need to be packaged into blocks. Moreover, in the DAG structure, each transaction can point to multiple previous transactions, allowing the blockchain to generate forks. Therefore, the DAG structure can make transactions reach consensus parallelly, which greatly improves the speed of processing transactions, that is, TPS. It can be seen that through the DAG structure, the consensus efficiency and transaction throughput of the system can be improved, meanwhile the problem of low scalability can be solved. Currently, the researches using DAG structure include IOTA [19], Byteball [20], InterValue [21], Nano [22], and JURA [23]. These researches use DAG structure to build a blockchain that can run stably for a long time. It shows that the DAG structure can replace the traditional chain structure to show better performance. In particular, the DAG-lattice structure is used in both the Nano and JURA projects. In the DAG-lattice structure, each node account has its own account chain, and only the node itself can add transactions to its own account chain. So, the transactions between accounts can be added to the blockchain asynchronously and parallelly. Moreover, in the lattice DAG structure, the node can prune the transactions from the account chain of the node out of communication and only keep their latest

transactions without affecting the system consensus. It can be seen that using the DAG-lattice structure in the VANETs environment can solve the problems caused by high dynamics and resource constraints.

Therefore, this paper proposes a lightweight blockchain architecture using DAG-lattice structure for VANETs, called V-Lattice. This architecture is general and suitable for highway and urban road. In this architecture, each node has its own account chain. The transactions they generated can be added to the blockchain asynchronously and parallelly, and resource-constrained vehicles can store the pruned blockchain and execute blockchain related operations normally. At the same time, in order to incentivize vehicle and roadside unit (RSU) to participate in the blockchain network actively and without malicious intent, this paper uses a reputation-based incentive mechanism; that is, each node participating in the blockchain has a reputation score, and the reputation score of nodes can be updated dynamically based on the node's behavior. The main contributions to this paper are as follows.

- (1) Based on the DAG-lattice structure, this paper proposes a lightweight blockchain architecture for VANETs. The lightweight nature is reflected in two aspects. One is the small amount of calculation. In this paper, the PBFT consensus algorithm is used, which is not discriminatory in computing power. The other is the small storage capacity. Vehicles with limited storage capacity store pruned blockchain instead of full blockchain.
- (2) This paper introduces a reputation-based incentive mechanism to encourage more nodes to participate in the blockchain, and through updating the node's reputation score dynamically, it motivates normal nodes and punishes malicious nodes.
- (3) This paper proposes a consensus method that can conduct asynchronous consensus on transactions generated by nodes.
- (4) Aiming at the lightweight blockchain architecture for VANETs proposed, this paper describes how common applications in the VANETs work under this architecture.

The rest of the paper is organized as follows. Section 2 describes research on the architecture introducing blockchain technology in VANETs and the current using of the DAG-lattice structure; Section 3 introduces the relevant components of the architecture proposed in this paper; Section 4 introduces the proposed lightweight blockchain architecture in detail; Section 5 describes the malicious attack scenarios that may occur in this architecture and analyzes how this paper deals with these malicious attacks; Section 6 verifies the proposed architecture; finally, the full paper is summarized in Section 7.

2. Related Work

2.1. Research on the Architecture of Using Blockchain in VANETs. At present, there have been many researches on

proposing the security architecture for VANETs using blockchain. These studies mainly include two categories. One is that the nodes (vehicle and RSU) in VANETs do not participate in the blockchain, and the other is that the nodes in VANETs participate in the blockchain.

In the research case where the nodes in VANETs do not participate in the blockchain, the blockchain is deployed as a storage service in nodes or platforms which maintain the operation of the blockchain outside the VANETs. Rahman et al. [4] proposed a secure Internet of Vehicles (IoV) framework that allowed vehicle data to be stored in blockchain and off-chain repositories for secure sharing with one's community of interest. In this framework, the blockchain provides storage as a service. Xu et al. [24] introduced a new framework for secured intelligent vehicle data sharing, namely, biometric blockchain (BBC). In the BBC, each vehicle can be connected to the BBC-based platform and store their IV-BC through the BBC-based cloud service. Kulathunge and Dayarathna [25] proposed a VANET communication framework based on blockchain functions. In this architecture, data are stored in the Hyperledger Fabric blockchain, and the VANETs nodes communicate with the blockchain through the REST protocol.

However, the blockchain runs on nodes outside the VANETs as a distributed storage environment, which can only guarantee the security of the data in the storage process and cannot improve the security of the data in the communication process through blockchain technology. Therefore, researchers have proposed some architectures in which the nodes in the VANETs participate in the blockchain. In these architecture studies, the blockchain runs in vehicles or RSUs in the VANETs, and the operation of the blockchain is maintained by the vehicles or RSUs. Yuan and Wang designed a seven-layer conceptual model for ITS in [26] and proposed a B2ITS framework. Dorri et al. [27] proposed a blockchain-based distributed privacy protection and security architecture for smart vehicles. The nodes in the architecture are clustered, and only the cluster heads (CHs) are responsible for managing the blockchain and executing its core functions. Singh and Kim [28] proposed an Intelligent Vehicle-Trust Point (IV-TP) mechanism for IV communication among IVs using blockchain technology. In this mechanism, the IV-TP data are managed through the blockchain. Leiding et al. [29] proposed a transparent, self-managed, and decentralized system which combined Ethereum and VANETs. In this system, each entity has an Ethereum address, and RSU provides Ethereum-based applications which have been deployed to the Ethereum blockchain. Sharma et al. [2] proposed a vehicle network architecture based on blockchain in the smart city (Block-VN). Block-VN includes controller nodes, miner nodes, and ordinary nodes. The ordinary nodes can send service request messages to miner nodes (vehicles) or controller nodes. Jiang et al. proposed a blockchain-based distributed VANETs architecture in [30]. According to different application purposes, the blockchain is divided into 5 types. The various blockchains do not communicate with each other, and they also have 5 different types of blockchain nodes.

The current architecture researches are considered from the perspective of application. The use of blockchain can solve some security issues, privacy, and so on. Moreover, the proposed architectures mainly study that the blockchain runs on which entities, what information is stored on the entities, and how the entities interact to ensure security in the VANETs. However, they do not consider the high dynamics of the VANETs and the problems caused by the use of blockchain under the condition of limited node resources.

2.2. Usage of DAG-Lattice Structure. The DAG-lattice structure is a kind of DAG structure. The first to use this structure was the Nano project [22]. In this project, each account has its own blockchain. The sending transaction and receiving transaction are separated, which can provide almost instantaneous transaction speed and unlimited scalability. Moreover, the transaction tracks the account balance, so that blockchain can be pruned without affecting the performance and security. In view of the limitations of current blockchain technology and the scalability requirement of having millions of TPS in the future, the JURA team proposed the decentralized JURA [23], in which novel data structure Fusus, PoU consensus mechanism, verifiable random function technology, dynamic monitored and distributed sharding, and artificial intelligence is used. The essence of Fusus is the DAG-lattice structure. The transaction records of accounts form the lattice, and the transaction records of each account are organized by DAG structure.

Both Nano and JURA are cryptocurrencies based on blockchain. However, some researchers have used the DAG-lattice structure in a noncryptocurrency environment. Zhou et al. [31] used the DAG-lattice structure in data tokenization and proposed the lattice blockchain model, namely, DLattice. This model has a double-directed acyclic graph (Double-DAG) structure, and each account is composed of a Token-Chain and a Data-Tree.

At present, the DAG-lattice structure has not been used in the VANETs environment.

3. The Components of V-Lattice

This section defines the basic components of V-Lattice from four aspects: node, account, transaction, and ledger.

3.1. Node. The node is software that runs on entities in the blockchain network. It follows the relevant protocols of the blockchain and can run all or part of the blockchain-related operations, including generating transaction, verifying transaction, transaction consensus, and storing transaction. In the VANETs environment, the entities are mainly mobile vehicles and RSUs fixed on both sides of the road. When a blockchain node is running on a moving vehicle, the mobility of the vehicle causes the vehicle to frequently join or leave the blockchain network. Moreover, the moving vehicle often forms small clusters at intersections, which makes it difficult to maintain a unified ledger among vehicles; when the blockchain node is running in fixed RSUs, the RSUs are

fixed and can maintain a stable network. However, because the blockchain ledger is stored on the RSUs, it cannot be guaranteed to keep communicating with the RSUs during the movement of vehicles. When the vehicles cannot communicate with the RSUs through a single hop or multiple hops, it is impossible to obtain data from blockchain or synchronize the transactions to the RSUs in real time.

Therefore, both vehicles and RSUs participate in the blockchain network as blockchain node and jointly maintain the blockchain in V-Lattice. The node running on the vehicle is called vehicle node, denoted as $VNode_i \in \{VNode_1, VNode_2, \dots, VNode_N\}$, where N is the number of vehicles. The node running on RSU is called RSU node, denoted as $RNode_i \in \{RNode_1, RNode_2, \dots, RNode_M\}$, where M is the number of RSUs.

3.2. Account. The account is the identity of a blockchain node participating in the system. For ease of management, each blockchain node has only one account. Each account consists of a public key and private key pair $\{P_r, S_r\}$, where the public key P_r is publicly available on the entire network to identify each account, and the private key S_r is kept by the account itself and is not publicly disclosed.

The public and private key pair of the account is generated by the Trust Center (TC). The trusted center is an absolutely trusted infrastructure that will not be maliciously attacked and used for generating public and private key pair, authenticating identity, and issuing certificates. Before joining the blockchain network, vehicle nodes and RSU nodes firstly need to generate public and private key pair through the TC. The public and private key pair of the vehicle account is generated by the fixed attributes of the vehicle (vehicle license plate, vehicle type, vehicle manufacturer, vehicle owner information, etc.). The public and private key pair of the RSU account is generated by the fixed attribute (number, etc.) of the RSU.

The V-Lattice contains vehicle accounts and RSU accounts, which represent vehicle nodes and RSU nodes, respectively. The vehicle account is denoted as $VAC_i \in \{VAC_1, VAC_2, \dots, VAC_N\}$, where N is the number of vehicles. RSU account is denoted as $RAC_i \in \{RAC_1, RAC_2, \dots, RAC_M\}$, where M is the number of RSUs.

3.3. Transaction. The transaction is a series of operations that cause state change in the blockchain. In a traditional blockchain, multiple transactions need to be packaged into a block firstly. It takes a certain amount of time to pack a block, and the packed block is generally relatively large, which is not conducive to transmission in a highly dynamic and bandwidth limited VANETs environment. Therefore, a transaction is regarded as a block in V-Lattice. Transaction and block can be used interchangeably.

In the traditional blockchain, a chain is maintained between all accounts. One operation is a transaction, and a transaction may involve two accounts. However, for a blockchain with a lattice structure, each account maintains its own account chain. In order to reduce the coupling and

enable transactions to operate asynchronously between account chains, a transaction under the traditional blockchain is separated into a sending transaction and a receiving transaction in V-Lattice, which is stored in sending and receiving account chains, respectively.

In the VANETs environment, the main behaviors of nodes include that node creates account, node broadcasts messages, node obtains messages from other nodes, and RSU updates node's reputation score. Therefore, the transaction types involved in V-Lattice include creating account transaction ($T_{\text{account_create}}$), sending message transaction ($T_{\text{message_send}}$), sending reputation score transaction ($T_{\text{reputation_send}}$), and receiving transaction (T_{receive}). $T_{\text{account_create}}$ is used to create the node's account; $T_{\text{message_send}}$ and T_{receive} appear in pairs to complete a message transmission together; $T_{\text{reputation_send}}$ and T_{receive} appear in pairs to complete a reputation score updating operation together. In particular, $T_{\text{account_create}}$ only involves one account, so there is no need for T_{receive} .

3.4. Ledger. The ledger is the data maintained by all accounts, and each account has an account chain. For each account, the transaction content on the account chain includes message-related transactions and reputation-related transactions. In order to improve transaction processing speed, facilitate transaction query, and process transactions asynchronously, the V-Lattice divides the account chain into Message Chain (MC) and Reputation Chain (RC). The ledger is stored by the structure of a directed acyclic graph (DAG). The DAG-lattice structure of the ledger is shown in Figure 1.

The genesis block is generated by the system during initialization, and it is the parent block of all creating account transaction. $T_{\text{account_create}}$ is the genesis transaction of the account chain. The MC and RC are, respectively, linked to $T_{\text{account_create}}$. $T_{\text{message_send}}$ and T_{receive} exist in pairs, and $T_{\text{message_send}}$ must be quoted by the corresponding T_{receive} . Similarly, $T_{\text{reputation_send}}$ and T_{receive} exist in pairs, and $T_{\text{reputation_send}}$ must be quoted by the corresponding T_{receive} . The change of node's reputation score is caused by the behavior of the node. In this architecture, the behavior of the node refers to whether the node sends the correct message. In other words, the behavior of the node can be reflected by $T_{\text{message_send}}$. Therefore, $T_{\text{reputation_send}}$ must quote the corresponding $T_{\text{message_send}}$ that resulted in the reputation score changing.

Through the DAG-lattice structure shown in Figure 1, the transactions can be performed asynchronously between accounts. At the same time, the transactions on MC and RC can also be performed asynchronously within the same account.

4. Proposed V-Lattice

This section introduces the proposed V-Lattice in detail.

4.1. The Overview of V-Lattice. In V-Lattice, vehicles, and RSUs together form a blockchain network. When the

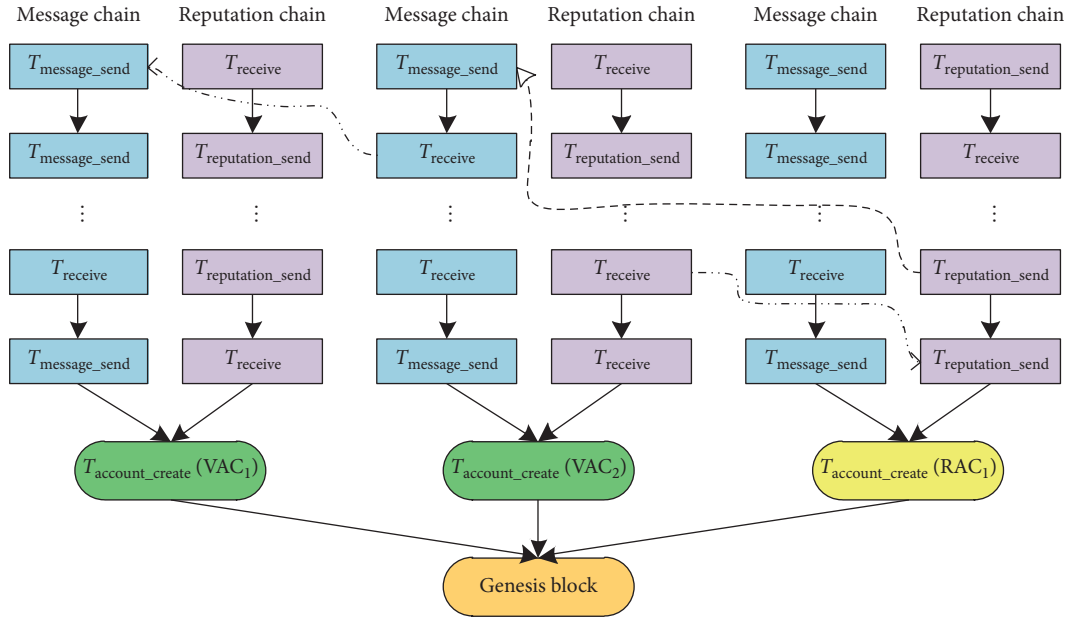


FIGURE 1: The DAG-lattice structure of ledger.

vehicles and the RSUs join the blockchain network, firstly they need to obtain the public and private key pair through the trusted center and then create account according to public key. The communication method between vehicles or between vehicles and RSUs is wireless communication, such as DSRC and 5G. The communication method between RSUs is wire communication. They communicate through the Gossip protocol to ensure the consistency of the blockchain. Figure 2 shows the network structure diagram.

The entities of this architecture mainly include Trust Center, vehicle, and RSU. The overall workflow chart between entities is shown in Figure 3.

Trusted Center, as an absolutely trusted entity, is used to generate public and private key pairs for vehicles and RSUs. Moreover, it is also used for assisting RSUs to verify the account information created by blockchain nodes.

Vehicle, as a lightweight node, has limited storage capacity and stores the pureed blockchain. In order to verify transactions and participate in consensus, the pureed blockchain stored in the vehicle should at least contain the creating account transaction and the latest message-related transactions and reputation-related transactions of the vehicle itself and other surrounding vehicles. However, as the vehicle is moving, the surrounding vehicles constantly change, which cause the vehicles to frequently update the locally pureed blockchain. Therefore, in order to prevent this problem and to improve the overall security of the blockchain, vehicles are encouraged to store more transactions.

RSU, as a full node, has the characteristics of large storage capacity and strong computing power. It can perform all blockchain functions and store all transaction records. The update of the reputation score requires a large amount of calculations, and the

transaction information of other vehicles is required, so it is completed by the RSU. For the message sent by the node, the RSU needs to judge whether the message is true or false (the method of judging whether the message sent by the node is true or false is mainly realized through trust management in the VANETs. This part is not the focus of this paper and will not be discussed in detail). If it is found that a fake message is sent by a node, its reputation is reduced; otherwise, its reputation is increased. For nodes with high reputation score, there are more opportunities to generate blocks and get rewards. Meanwhile, the messages generated by them are processed firstly. For nodes with low reputation score, the messages will not be accepted by other nodes and they cannot get priority services. This can help encourage the vehicle to stay normal.

In V-Lattice, the operations performed by the vehicle include generating transactions, verifying transactions, forwarding transactions, and participating in consensus. The operations performed by RSU include generating transactions, verifying transactions, forwarding transactions, participating in consensus, and updating node's reputation score.

4.2. Transaction Structure

4.2.1. *Creating Account Transaction.* To create an account, the blockchain node needs to send a creating account transaction as shown in Table 1. The type field identifies the type of transaction, and its value is account_create. The account field indicates the address of the account to be created. The difficulty field is the difficulty of Proof of Work (PoW), and the nonce is a random number that meets the difficulty of PoW. The difficulty and nonce will be described in detail in Section 4.5. The timestamp field indicates the

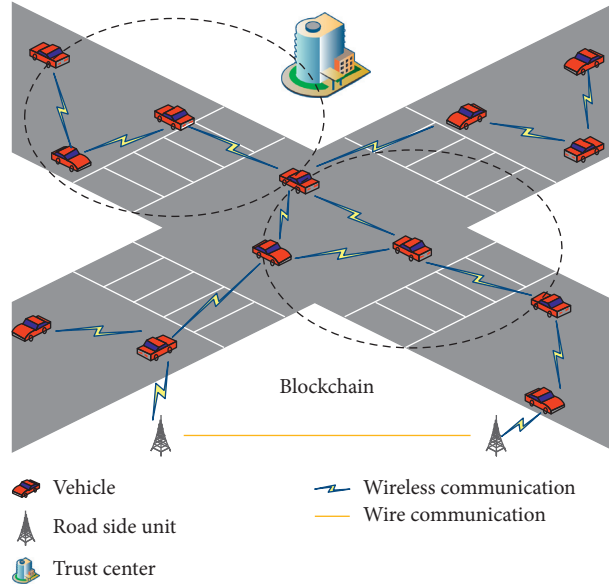


FIGURE 2: The network structure diagram.

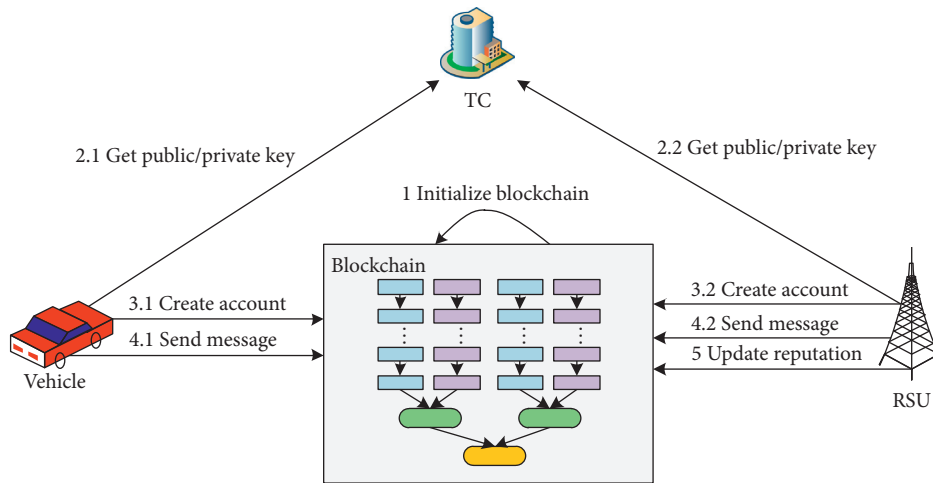


FIGURE 3: Overall workflow chart.

time when the transaction is created. The signature field is the signature of the transaction by the node creating transaction. Through signature, on the one hand, the node cannot deny its signature. On the other hand, it can also determine whether the transaction content has been tampered by malicious nodes, thereby ensuring the integrity of the information transmission process. The signature is generated by the transaction producer using his own private key S_r to encrypt the hash value of the transaction content (except the signature field), that is, $\text{signature} = E_{S_r} \{H_1(\text{type}, \text{account}, \text{nonce}, \text{difficulty}, \text{timestamp})\}$, where H_1 is the hash algorithm.

4.2.2. Sending Message Transaction. When the node sends message content, it firstly needs to send a sending message transaction as shown in Table 2. The type field identifies the

TABLE 1: The structure of creating account transaction.

| |
|--|
| $T_{\text{account_create}} \{$ $\text{type: account_create,}$ $\text{account: f6b4c0685f...acc5f7918c,}$ nonce: 154883521, $\text{difficulty: 0000000111...1111111111,}$ $\text{timestamp: 1606128999,}$ $\text{signature: af0c8bb4d5...b3786dd6ac}$ $\}$ |
|--|

type of transaction, and the value is message_send. The previous field identifies the hash value of the previous block. The source field is the address of the account that sent the transaction, and the destination field is the address of the account that receive the transaction. The msg field is used to indicate the message sent by node, including desc field and

TABLE 2: The structure of sending message transaction.

```

T_message_send {
  type: message_send,
  previous: f69bb1178e...e5fdc9c7d1,
  source: f6b4c0685f...acc5f7918c,
  destination: 4d9c22f58f...7936b855da,
  nonce: 4565123156,
  difficulty: 000000111...1111111111,
  timestamp: 1606133657,
  msg: {desc, content {plaintext_cont, ciphertext_cont}},
  signature: c8dbb6c772...6c46aabbf2
}

```

content field. The desc field is the description information of content, and content field refers to the specific message content to be sent which includes public content plaintext_cont field and encrypted content ciphertext_cont field. The desc field and plaintext_cont field are displayed in plaintext in the transaction, which can be directly read by other nodes, whereas the ciphertext_cont field is displayed in ciphertext in the transaction. Only after the sender authorizes it, it can be decrypted and read through the decryption key. The signature field is generated by the transaction producer using his own private key S_r to encrypt the hash value of the transaction content (except for the signature field), namely, $\text{signature} = E_{S_r}\{H_1(\text{type, previous, source, destination, nonce, difficulty, timestamp, msg})\}$, where H_1 is the hash algorithm.

4.2.3. Sending Reputation Score Transaction. When RSU updates the reputation score of nodes, it needs to send a sending reputation score transaction as shown in Table 3 to the nodes. The type field identifies the type of transaction, and its value is reputation_send. The destination field indicates the account address of the node (vehicle or RSU) whose reputation score needs to be updated. The associate field is the hash value of the corresponding transaction that causes the node's reputation score to change. It is used to associate the change of node's reputation score with the node's behavior. The reputation field records the reputation score of the node, where desc field describes the update method of the node's reputation score, and the score field records the updated reputation score of the node. The timestamp_dest field is the timestamp when the node whose reputation score is updated signs the transaction. The signature field and signature_dest field, respectively, represent the signature of the transaction producer (i.e., the RSU updating the node's reputation score) and the node whose reputation score is updated. As the reputation score update is completed by the RSU, and the reputation score needs to be recorded on the Reputation Chain of the node whose reputation is updated, so the RSU and the node whose reputation is updated need to sign the transaction jointly. Through the joint signature method, the updated node can not only track the change of reputation score but also can prevent the updated node from modifying the reputation score, and at the same time can prevent the RSU from being attacked and maliciously updating the node's reputation score. The signature field is generated by the

TABLE 3: The structure of sending reputation score transaction.

```

T_reputation_send {
  type: reputation_send,
  previous: 2174a3b1cd...5a57e16be9,
  source: f6b4c0685f...acc5f7918c,
  destination: 4d9c22f58f...7936b855da,
  associate: d24c9bfe35...fcfebebe64,
  nonce: 423426631123,
  difficulty: 000000111...1111111111,
  timestamp: 1606133830,
  reputation: {desc, score},
  signature: 3af4cdace5...b326438866,
  timestamp_dest: 1606133840,
  signature_dest: b9bec112d8...63e65bd294
}

```

transaction producer using his own private key to encrypt the hash value of the transaction content (except for the signature field and signature_dest field), that is, $\text{signature} = E_{S_r}\{H_1(\text{type, previous, source, destination, associate, nonce, difficulty, timestamp, reputation})\}$, where S_r is the private key of the transaction producer and H_1 is the hash algorithm. The signature_dest field means that the node whose reputation score is updated uses its own private key to encrypt the hash value of the transaction content (except for the signature field and signature_dest field), that is, $\text{signature}_{\text{dest}} = E_{D^{S_r}}\{H_1(\text{type, previous, source, destination, associate, nonce, difficulty, timestamp, reputation, timestamp}_{\text{dest}})\}$, where D^{S_r} is the private key of the node whose reputation is updated, and H_1 is hash algorithm.

4.2.4. Receiving Transaction. When node receives a sending message transaction or a sending reputation score transaction, it needs to send a receiving transaction as shown in Table 4. The type field is used to identify the type of the receiving transaction. If a receiving transaction is initiated for a sending message transaction, the type field is message_receive. And if a receiving transaction is initiated for a sending reputation score transaction, the type field is reputation_receive. The destination field indicates the address of the account that receives the transaction, which is consistent with the content of the source field. The associate field points to the hash value of the corresponding sending transaction and is used to associate the receiving transaction with the sending transaction. The signature field is generated by the transaction producer using his own private key to encrypt the hash value of the transaction content (except the signature field), that is, $\text{signature} = E_{S_r}\{H_1(\text{type, previous, source, destination, associate, nonce, difficulty, timestamp})\}$, where S_r is the private key of the transaction producer, and H_1 is the hash algorithm. Only after the receiving transaction and the corresponding sending transaction are both agreed and added to the blockchain can an operation be considered complete.

4.3. Transaction Verification. After blockchain node receives the transaction T sent by the node K , ($K \in VNode_i \parallel RNode_i$),

TABLE 4: The structure of receiving transaction.

| |
|--|
| T_{receive} { type: reputation_receive/message_receive, previous: 34676fa04c...97c2d3e2a2, source: f6b4c0685f...acc5f7918c, destination: f6b4c0685f...acc5f7918c, associate: 1a6a65ede0...62d93672a1, nonce: 551354523, difficulty: 000000111...1111111111, timestamp: 1606133858, signature: abf7727736...9854bae5d9 } |
|--|

it needs to verify the transaction T . The verification content includes (1) whether the transaction is received for the first time; (2) the nonce value; (3) the timestamp; (4) the signature of the transaction; (5) the associated transaction. The specific verification procedures of transaction T are shown in Figure 4.

4.3.1. Whether the Transaction Is Received for the First Time.

Upon receiving a transaction T , first it is necessary to verify whether the transaction is received for the first time. The verification method is to check the recent historical transaction information under the account that sent the transaction T to determine whether there is a copied transaction. If there is a copied transaction, that is, the transaction has been received before, it may be a copied transaction generated by a malicious node, and the transaction needs to be discarded.

4.3.2. The Nonce Value.

The correctness of the nonce value can determine whether the blockchain node K has completed proof of work. Only when the node has completed the proof of work can it send transactions to the blockchain network. So, the verification of the correctness of the nonce value is very important. The method of verification is to combine the type field, the difficulty field, the timestamp field, and the nonce field of the transaction to calculate the hash value and determine whether the Hash (type, difficulty, timestamp, nonce) meets the PoW difficulty. If the nonce value meets the value of the difficulty field in the transaction, it means that the node has completed the corresponding proof of work; otherwise, the transaction is discarded.

4.3.3. The Timestamp.

The purpose of verifying the timestamp is to determine whether the node K sends the transaction immediately after it generates the transaction and to prevent malicious nodes from generating pre-computed POW attacks (detailed in Section 5.3). The method to verify the correctness of the timestamp is to verify whether the time interval Δt between the timestamp of the transaction timestamp or $\text{timestamp}_{\text{dest}}$ and the time of receipt of the transaction Time satisfies $\Delta t < \text{Threshold}_T$, where $\Delta t = \text{Time} - \text{timestamp} \parallel \text{timestamp}_{\text{dest}}$. If Δt exceeds the threshold Threshold_T , it means that the node does not

send the transaction immediately and the verification failed. It should be noted that, for the sending reputation score transaction, it contains the timestamp and $\text{timestamp}_{\text{dest}}$. If the transaction is issued by the RSUs which update the reputation score, the timestamp will be verified. If the transaction is issued by the node whose reputation score is updated, then verify the $\text{timestamp}_{\text{dest}}$.

Under normal circumstances, the determinants of the time interval Δt include the PoW time t_T^{pow} of the transaction, the queue time t_T^{que} of the transaction in the node, and the transmission time t_T^{trans} from the sending node to the destination node of the transaction. Therefore, the threshold Threshold_T is expressed as follows:

$$\text{Threshold}_T = t_T^{\text{pow}} + t_T^{\text{que}} + t_T^{\text{trans}} + \eta_T. \quad (1)$$

t_T^{pow} is the time required for the node to complete PoW for the transaction. This time usually varies and is specifically related to the node's reputation score and computing power. However, the computing power between nodes is usually similar, so the main influencing factor of this time is the reputation score of the node.

t_T^{que} is the queuing time of transactions in the node, which is determined by the frequency of the sending packets and the frequency of generating transactions. If there is no malicious attack (e.g., Denial of Service), the frequency of transactions generated by the node is generally not too high.

t_T^{trans} is the transmission time of the transaction from the sending node to the receiving node, and this time is related to the network topology.

η_T is the error time. Since there may be error in the calculation of the times t_T^{pow} , t_T^{que} and t_T^{trans} , they need to be corrected by η_T . The value of η_T needs to be determined according to the real network environment and historical experience. If η_T is too small, it will cause normal transactions to fail to be verified. On the contrary, if the value of η_T is too large, transactions generated by malicious nodes may be mistaken for normal transactions.

4.3.4. The Signature of the Transaction.

By verifying the signature of the transaction, it can be determined that the transaction is indeed generated by the producer of the transaction rather than forged by other malicious nodes. The method to verify the signature is firstly to calculate the hash value of the transaction (except for the signature field or $\text{signature}_{\text{dest}}$ field) $\text{hash}_1 = H_1(\text{type, account, nonce, difficulty, timestamp})$ (here we take the creating account transaction as an example), where H_1 is the hash algorithm, which is the same hashing algorithm as when signing. Then, it needs to use the public key P_r of the transaction producer to decrypt the signature and obtain the decrypted hash value $\text{hash}_2 = D_{P_r}\{\text{signature}\}$. Finally, it needs to judge whether hash_1 and hash_2 are the same. If $\text{hash}_1 = \text{hash}_2$, the signature

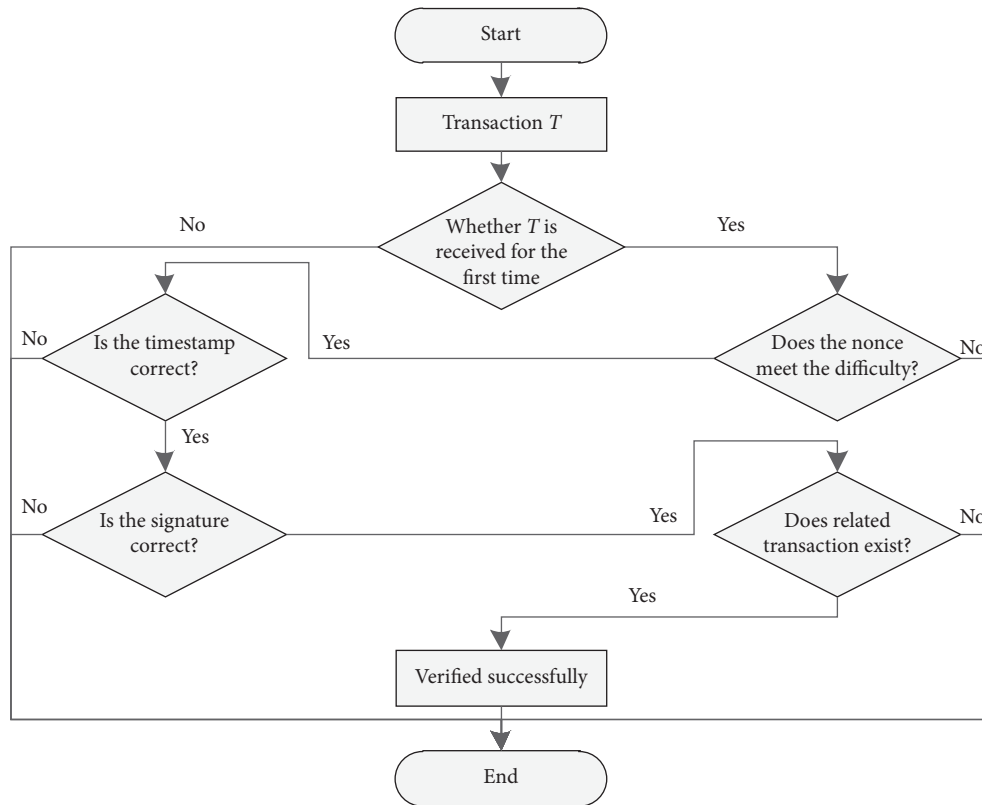


FIGURE 4: Transaction verification flow chart.

verification is passed; otherwise, the transaction is discarded. The specific verification process is as shown in Figure 5.

It is particularly important to note that, for the sending reputation score transaction, if the transaction is generated by the RSU that updates the node's reputation score, it only needs to verify the signature. If the transaction is sent by the node whose reputation score is updated, the signature and the signature_{dest} need to be verified at the same time.

4.3.5. The Associated Transaction. For the creating account transaction and sending message transaction, this process can be ignored because they do not involve related transactions. However, for the sending reputation score transaction, it is necessary to associate the corresponding transaction that causes the reputation score change. For the receiving transaction, the corresponding sending message transaction and the sending reputation score transaction need to be associated. Therefore, after receiving the sending reputation score transaction and receiving transaction, it is necessary to verify whether the associated transaction has been added to the blockchain. If the associated transaction exists, the transaction verification is passed; otherwise, the transaction is discarded.

Transactions that have passed the above 5-step verification are added to the transaction candidate set. The transaction candidate set is used to temporarily store verified transactions and wait for consensus.

4.4. Consensus Scheme. Consensus algorithm is the core part of the blockchain, and it can guarantee the consensus of distributed blockchain nodes. Common consensus algorithms currently include Proof of Work (PoW) used in Bitcoin system, Proof of Stake (PoS) used in Ethereum, and Practical Byzantine Fault Tolerance (PBFT) used in Hyperledger. By comparing PoW, PoS, and PBFT, Ayaz et al. [32] believe that PBFT has the most potential to become the consensus algorithms in the VANETs environment. The PoW consensus algorithm is discriminatory in computational power, and the POS consensus algorithm is discriminatory in stakes owned. However, the PBFT consensus algorithm is based on voting, which is not discriminatory in computational power and stakes owned. It can be seen that it is suitable for use in the VANETs environment. Therefore, PBFT-based consensus algorithm is adopted in this architecture.

The PBFT consensus algorithm has relatively high communication complexity, and its complexity is $O(n^2)$, where n is the number of nodes participating in the consensus. As you can see, performance degrades dramatically when there are many nodes in the network. Moreover, in PBFT consensus algorithm, if the byzantine nodes exceed 1/3 of the total number of participating consensus nodes, the consensus algorithm will fail. Based on these two considerations, this architecture selects nodes within M hop around the transaction producer rather than all nodes in the network, which can improve consensus efficiency. As a parameter in the consensus process, M is dynamically

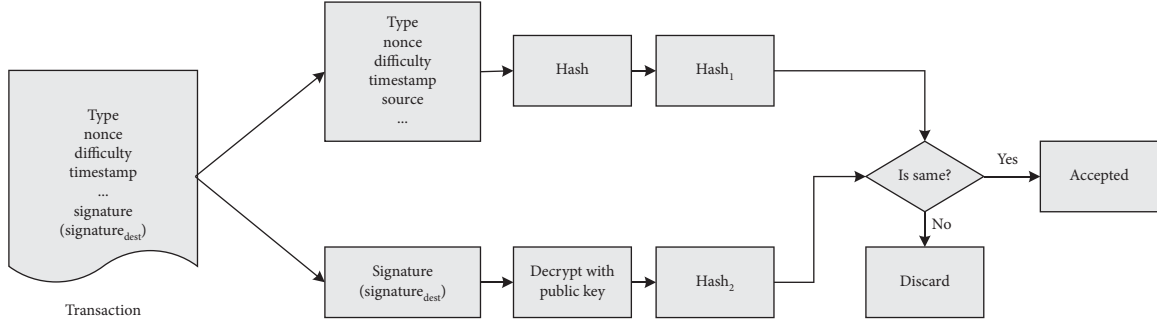


FIGURE 5: The signature verification process of the transaction.

adjusted according to the network environment. At the same time, node with reputation score greater than a certain threshold T is selected to participate in the consensus, so as to avoid the byzantine nodes participating in the consensus from exceeding 1/3 of the total number of nodes. If a node participating in the consensus is attacked and behaves maliciously, the RSU can quickly find and reduce its reputation score, thus preventing the attacked node from participating in the consensus process.

In this architecture, transaction consensus can be asynchronous, that is, transactions of different accounts, or transactions related to messages and reputation scores of the same account can be asynchronously agreed. Therefore, in the consensus process, all nodes can become the master node and present a consensus on transaction. But for a certain transaction, only one node can become the master node to avoid the occurrence of forks. However, in the traditional PBFT consensus algorithm, the selection method of the master node has the problem of forecasting in advance. In order to avoid a malicious attack on a certain transaction, the master node proposing a certain transaction should not be predicted in advance.

The verifiable random function VRF [33] maps the input value to a verifiable output. It is characterized by not being able to predict the random result and verifying the correctness of the result. In Algorand [34] and Ouroboros Praos [35], the authors use VRF to select the node proposing block, which has high security. Therefore, VRF is used to determine the master node of the consensus in this paper and PBFT consensus algorithm is improved. To achieve asynchronous consensus, this paper appends transaction information (the account||source field and the timestamp field) to the VRF verification information. Transaction can be uniquely identified by additional transaction information. The detailed procedures are as follows.

- (1) Each node G_i of scheduled generation block selects a transaction from the local transaction candidate set, uses the timestamp of transaction and the private key $G_i^{S_r}$ of node G_i as the input of VRF to generate the hash output of hash_value and proof_value through the VRF calculation function VRF_HASH and VRF proof function VRF_Proof. The calculation method of hash_value and proof_value is shown in the following:

$$\text{hash_value} = \text{VRF_HASH}(G_i^{S_r}, \text{timestamp}), \quad (2)$$

$$\text{proof_value} = \text{VRF_Proof}(G_i^{S_r}, \text{timestamp}). \quad (3)$$

- (2) The node G_i sends the VRF verification message $\{\text{hash_value}, \text{proof_value}, \text{account}||\text{source}, \text{timestamp}, \text{account}_{G_i}\}$ to the surrounding nodes in the prepare phase of the PBFT consensus algorithm, where the account_{G_i} is the account of the node G_i . In particular, it should be noted that the nodes participating in the consensus are both the master node and the client node in this architecture. Therefore, the request phase is not involved in this architecture.
- (3) After the surrounding nodes as VRF verifier receiving $\{\text{hash_value}, \text{proof_value}, \text{account}||\text{source}, \text{timestamp}, \text{account}_{G_i}\}$, firstly they need to verify whether $\text{hash_value} = \text{VRF_P2H}(\text{proof_value})$ is true. If true, then zero-knowledge proof is performed through the VRF verification function $\text{VRF_Verify}(G_i^{P_r}, \text{timestamp}, \text{proof_value})$, where $G_i^{P_r}$ is the public key of the node G_i . If the verification result is true, this means that the hash_value is generated by the node G_i through timestamp.
- (4) Among the verified VRF verifiers for the same transaction $\{\text{account}||\text{source}, \text{timestamp}\}$, only the node whose hash_value meets a certain condition is selected as the master node to perform other stages of the PBFT consensus, including prepare stage, commit stage, and reply stage. It should be noted that the client is not involved in this architecture, so the message is sent to the master node during the reply phase.

Through the above process, the master node can be randomly generated without being predicted in advance. At the same time, the asynchronous transaction consensus can be achieved.

As the vehicle moves, it may not be able to communicate with the RSU. In this case, the transaction that has reached consensus can only synchronize with the surrounding vehicles but cannot synchronize with the RSUs in real time. Therefore, it is necessary to use the Delay Tolerant Network (DTN). The vehicle firstly saves the transaction that has

reached consensus locally, and when it moves to the communication range of the RSU, it synchronizes the transaction to the RSU. In this way, all the transactions that have reached consensus can eventually be synchronized with the RSUs.

4.5. PoW Anti-Spam. Before nodes send a transaction, they first need to compute a random number satisfying a particular difficulty to complete the proof of work. Transactions cannot be sent until PoW is complete. When other nodes receive a transaction, they first need to verify whether nonce value is correct. Unlike PoW in Bitcoin, it is only used as an anti-spam tool in this paper. Since the amount of calculation is consumed during PoW, this can prevent malicious nodes from always sending malicious messages to occupy network bandwidth and perform DoS attacks.

In this paper, the difficulty of POW is related to the node's reputation score. The higher the node's reputation score, the lower the difficulty of POW. On the contrary, the lower the node's reputation score, the higher the difficulty of POW. The specific calculation method is shown in the following equations:

$$N_{zero} = \lfloor e^{-(\vartheta \cdot \text{Score} + \mu)} * N_{hash} \rfloor, \quad (4)$$

$$\text{difficulty} \leftarrow \underbrace{0 \dots 0}_{N_{zero}} + (2^{N_{hash} - N_{zero}} - 1)B, \quad (5)$$

where N_{hash} is the length of the hash value, which depends on the specific hash algorithm (e.g., the MD5 algorithm is 128 bits, the SHA-1 algorithm is 160 bits, and the SHA-256 algorithm is 256 bits). The specific selection of hash algorithms needs to be determined according to the application requirements. N_{zero} is the number of the first 0 in the POW difficulty string. The Score is the node's reputation score. ϑ and μ are two preset parameters, which are used to control the change rate and upper limit of POW difficulty. $\lfloor \cdot \rfloor$ indicates that the number is rounded down to the integer part. $(\cdot)B$ means convert a decimal number into a binary number.

As shown in Figure 6, POW is relatively difficult when the node's reputation score is very low. But with the increase of reputation score, the difficulty of PoW decreases gradually.

After the difficulty is determined, the PoW can be carried out for the transaction. The method is to try random numbers until a random number is found that satisfies the difficulty. The specific PoW method is shown in the following:

$$\text{Hash}(\text{type}, \text{difficulty}, \text{timestamp}, \text{nonce}) \leq \text{difficulty}. \quad (6)$$

4.6. Application Cases. In the VANETs environment, the communication between vehicles or between vehicles and RSUs is mainly used to improve traffic safety and efficiency and realize intelligent traffic management. Therefore, in the proposed architecture, application cases mainly include node creates accounts, node broadcasts messages, node obtains messages from other nodes, and RSU updates nodes'

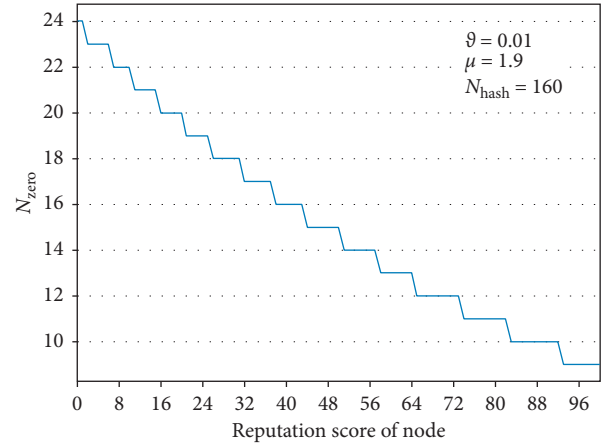


FIGURE 6: The relationship between PoW difficulty and reputation score.

reputation score. Under this architecture, the working methods of these four application cases are as follows.

4.6.1. The Node Creates Accounts. When a node joins the blockchain for the first time, an account needs to be created. The specific procedures are as follows:

- 1) Node $A \in VNode_i \parallel RNode_i$ registers information in the trusted center and obtains the public and private key pair $\{A^P_r, A^S_r\}$.
- 2) Node A generates a creating account transaction $T_{\text{account_create}}$, signs the transaction, and sends it to the RSU. The account field is the public key A^P_r of node A .
- 3) RSU $B \in RNode_i$ that receives $T_{\text{account_create}}$ verifies whether the public key account A^P_r has been registered in the TC.
- 4) If the verification is successful, $T_{\text{account_create}}$ will be verified, reached consensus, and added to the account chain of the account as the genesis transaction of the account.

4.6.2. The Node Broadcasts Messages. Vehicles or RSUs need to frequently broadcast some warning messages, road condition messages, and so on to improve traffic safety and efficiency. The specific procedures are as follows:

- 1) Node $A \in VNode_i \parallel RNode_i$ generates a sending message transaction $T_{\text{message_send}}$, signs it, and broadcasts it to surrounding nodes. It should be noted that the broadcast message does not have a definite destination node, and the transaction corresponding to the message needs to be stored in the node A 's Message Chain. So, the destination field and the source field have the same content, that is, the account address of node A . The description information $\text{info}_{\text{desc}}$ of the message is recorded in the desc field of msg, and the content info_{pub} that is publicly accessible by other nodes is recorded in the

plaintext_cont field. The content $\text{info}_{\text{auth}}$ that needs to be authorized for other nodes to access is encrypted and recorded in the ciphertext_cont field, that is, $\text{ciphertext_cont} = E_{A^{Pr}}(\text{info}_{\text{auth}})$.

- (2) After receiving $T_{\text{message_send}}$ sent by node A , other nodes verify it, reach a consensus, and add it to the node A 's Message Chain.
- (3) After $T_{\text{message_send}}$ is added to the blockchain, node A generates a receiving transaction T_{receive} , signs it, and sends it to the network.
- (4) Once T_{receive} is received, other nodes verify the transaction, reach a consensus, and add it to the node A 's Message Chain.

4.6.3. The Node Obtains Messages from Other Nodes.

Node $B \in \text{VNode}_i \parallel \text{RNode}_i$ in network needs to obtain content from another node $A \in \text{VNode}_i \parallel \text{RNode}_i$ or obtain access to encrypted content in the Message Chain from node A . For node A , decrypting the ciphertext on the blockchain every time, encrypting it with the public key of node B , and then sending it to node B would undoubtedly cause a lot of trouble for node A as the data owner. Therefore, using proxy reencryption technology [36] in this paper adopts, node A only needs to encrypt the ciphertext with the reencryption key and send it to node B . The specific procedures for node B to obtain a message from node A are as follows:

- (1) Node B generates a sending message transaction $T_{\text{message_send}}$, signs it, and broadcasts to the surrounding nodes. The destination field is the account address of node A , and the desc field in the msg field records the content name that will be obtained from node A or the hash value of a block in the node A 's Message Chain.
- (2) After receiving $T_{\text{message_send}}$ sent by node B , other nodes verify it, reach a consensus, and add it to the node B 's Message Chain.
- (3) Node A generates a receiving transaction T_{receive} , signs it, and sends it to the network. At the same time, node A generates ciphertext $\text{Data}_{\text{Encry}}$ based on what node B needs to obtain. The procedures of generating $\text{Data}_{\text{Encry}}$ are as follows:
 - (a) Node A generates a reencryption key $\text{Re} - \text{Key}_{A \rightarrow B}$ based on its own public key A^{Pr} and public key B^{Sr} of node B .
 - (b) Node A generates the data Data_s sent to node B . If the content that node B needs to obtain is common content C_{comm} which is not on the blockchain, then C_{comm} needs to be encrypted with its own public key; that is, $\text{Data}_s = E_{A^{Pr}}(C_{\text{comm}})$. If the content that Node B needs to obtain is content M_{encry} on the blockchain, then $\text{Data}_s = M_{\text{encry}}$.
 - (c) Node A encrypts the data Data_s sent to node B by the reencryption algorithm; that is, $\text{Data}_{\text{Encry}} = \text{Re} - \text{Enc}(\text{Data}_s, \text{Re} - \text{Key}_{A \rightarrow B})$.

- (4) Node A generates a sending message transaction $T_{\text{message_send}}$, signs it, and broadcasts it to surrounding nodes. The destination field is the account address of node B , and the ciphertext $\text{Data}_{\text{Encry}}$ of the message that needs to be sent to node B is recorded in the ciphertext_cont field.
- (5) After receiving $T_{\text{message_send}}$ sent by node A , other nodes verify it, reach a consensus, and add it to the node A 's Message Chain.
- (6) Node B decrypts $\text{Data}_{\text{Encry}}$ with its own private key to obtain the required content and at the same time generates a receiving transaction T_{receive} and sends it to the network.
- (7) After receiving T_{receive} sent by node B , other nodes verify it, reach a consensus, and add it to the node B 's Message Chain.

4.6.4. The RSU Updates Nodes' Reputation Score. The RSU $A \in \text{RNode}_i$ in network can update the reputation score of node $B \in \text{VNode}_i \parallel \text{RNode}_i$ according to its behavior, so as to motivate or punish node B . The specific procedures for node A to update the reputation score of node B are as follows:

- (1) Node A generates a sending reputation score transaction $T_{\text{reputation_send}}$, signs it, and sends it to node B . The destination field is the account address of node B . The associate field is the hash value of the corresponding transaction that causes the reputation score to change. The desc field of the reputation field describes the method of updating the reputation score, and the score field records the updated reputation score of node B .
- (2) When node B receives $T_{\text{reputation_send}}$, it first verifies the transaction. After the verification is passed, node B records the current time in the timestamp_dest field, signs the transaction again, and broadcasts it to surrounding nodes.
- (3) After receiving $T_{\text{reputation_send}}$ sent by node B , other nodes verify it, reach a consensus, and add it to the node B 's Reputation Chain.
- (4) After $T_{\text{reputation_send}}$ is added to the blockchain, node B generates a receiving transaction T_{receive} , signs it, and sends it to the network.
- (5) After receiving T_{receive} sent by node B , other nodes verify it, reach a consensus, and add it to the node B 's Reputation Chain.

During the reputation score update process, the RSU and the updated node are required to sign transaction simultaneously. The RSU signs the sending reputation score transaction and sends it to the corresponding vehicle. The corresponding vehicle signs again and then sends the transaction to the network to be added to blockchain. However, if the node (especially a moving vehicle) leaves the communication range of the current RSU when its reputation is updated, the forwarding is done with the assistance of other RSUs; if the node completely leaves the network (in

an offline state), the RSU saves the relevant transaction temporarily until the node joins the network and then updates. If a node performs malicious behavior and does not sign the sending reputation score transaction, the node will be punished and pulled into the blacklist so that it can no longer generate new transactions.

5. Security Analysis

This section describes the malicious attack scenarios that may occur in architecture and analyzes how this architecture deals with these malicious attacks.

5.1. The Sybil Attack. A malicious node may create multiple accounts and launch sybil attack on the network. However, in this architecture, public key, and private key pairs of vehicle and RUS are issued by TC, which is confirmed when the account is created. So, the malicious node cannot create large numbers of accounts to carry out the sybil attack.

5.2. The Transaction Flooding Attack. A malicious node may send a large number of malicious transactions to the network, which affects other nodes' judgment on the correctness of messages in the network. In this architecture, the node needs to complete the PoW before sending a transaction. The difficulty of PoW is related to the node's reputation score. After a node sends a malicious transaction, it will be discovered by the RSU and its reputation score will be reduced. This makes it increasingly difficult for node to initiate malicious attack. The node will abandon the attack voluntarily until PoW consumes more computational resources than the benefits of launching an attack.

5.3. The Precomputed PoW Attack. The timestamp is generated automatically by the system and cannot be changed by the node that generates the transaction. However, malicious node can save transactions that have completed PoW locally without sending them to the surrounding nodes. After a period of waiting, transactions within this period are sent out simultaneously, and then the network or node is attacked. In this architecture, after the node receives a transaction, it first verifies the transaction including the verification of the timestamp. If a malicious node waits for a period of time before sending a transaction, the interval between the timestamp of the transaction and the current time exceeds the threshold, and the verification cannot be passed. As a result, the transaction is discarded and cannot generate a valid attack.

5.4. The Conspiracy Attack. Two or more malicious nodes in the network may join together to provide false voting information during consensus process to conduct a conspiracy attack. The PBFT consensus method is used in this architecture. Consensus results can be affected when 1/3 of the malicious nodes participating in the consensus conduct a conspiracy attack. However, in this architecture, nodes within the M hop range around the production transaction

node and with a reputation score greater than a certain threshold are selected to participate in the consensus. Because this architecture uses reputation score to motivate and punish nodes. Malicious nodes usually have low reputation scores, thus greatly reducing the probability of conspiracy attacks.

6. Architecture Verification

This section verifies proposed architecture, including the security verification of using Petri nets and the feasibility verification of PoW anti-spam through experiments.

6.1. Security Verification Based on Petri Net. Petri net is a mathematical tool for researching system. It can be used to describe and analyze asynchronous and concurrent system, as well as design and optimize system. Colored Petri Net (CPN) is a kind of advanced Petri Net that combines Petri Net and programming language. A token represents an object with a set of attributes. In CPN, each color set represents a different resource in the system, and the color set can be of multiple data types. At the same time, CPN supports hierarchical modeling, allowing the model to be partitioned into manageable parts that can be reused. Therefore, this paper adopts a top-down method to establish a hierarchical CPN model for the proposed V-Lattice.

6.1.1. The CPN Model. This paper defines the CPN through nine-tuples, namely, $CPN = (P, T, A, \Sigma, V, C, G, E, I)$. The top layer CPN model is shown in Figure 7, and the transition Verify Transaction is substitution transition as shown in Figure 8. The detailed introduction of the nine-tuple is as follows:

P is set of places, which is identified by an ellipse in figure. The upper right corner of each place is the initial function I , and the lower right corner is the color set function C . There are 6 places in the top layer CPN model, and 8 places in the substitution transition Verify Transaction.

T is the set of transitions that are identified by a rectangle in figure, in which the double-line rectangles represent substitution transition. The guard function G is located in the upper left corner of each transition. It is not involved in this model. There are 5 transitions in the top layer CPN model and 5 transitions in the substitution transition Verify Transaction.

A is the set of arcs that are used to connect places and transitions. Each arc contains an arc expression function E .

Σ is the set of color sets, and V is the set of variables. In this model, the definition of set of color sets and set of variables are shown in Table 5.

As shown in Figure 7, in the top layer CPN model, the Generate transition generates transactions which need to be sent and adds them to the list of transactions. The Send transition broadcasts the next transaction to be sent to the

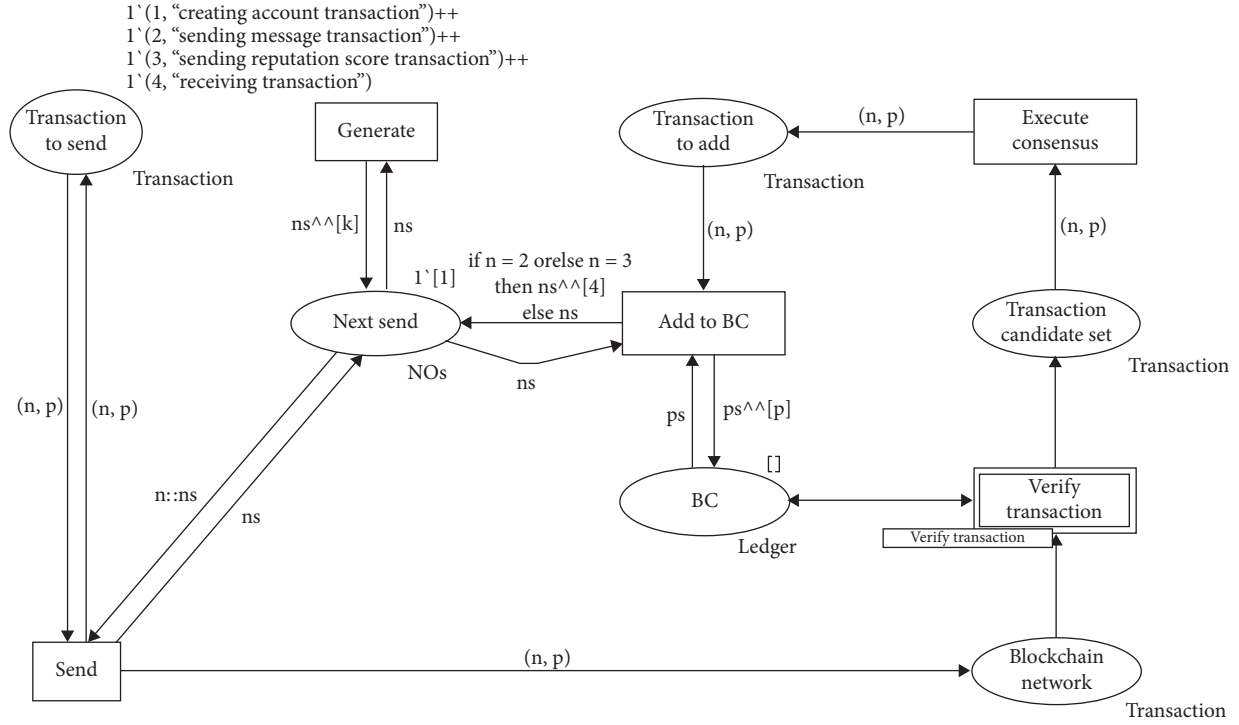


FIGURE 7: The top layer CPN model of V-Lattice.

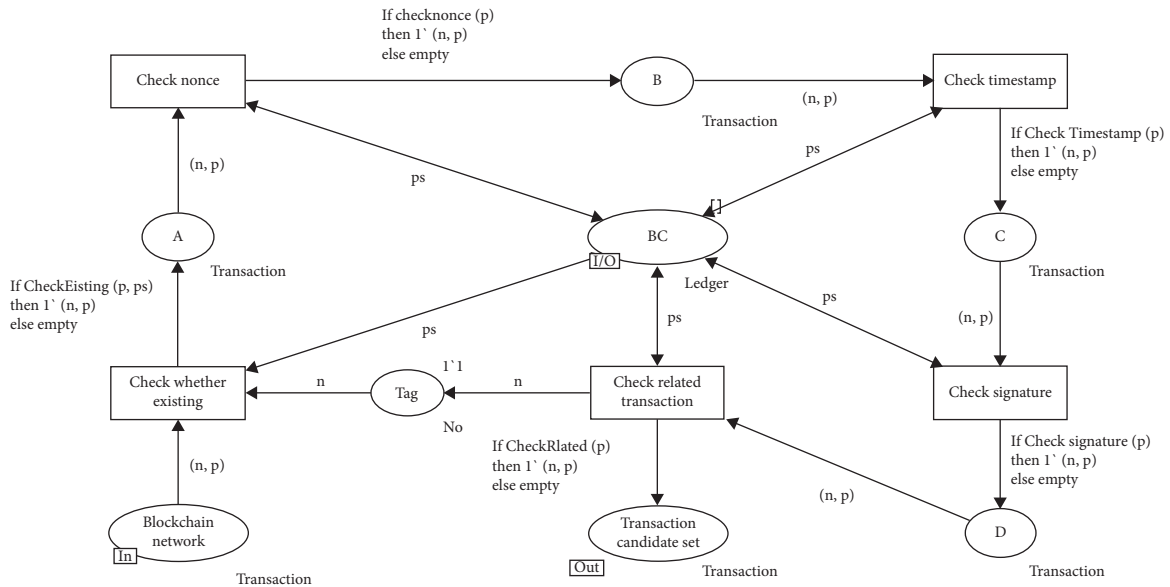


FIGURE 8: The subpage model of the Replacement Verify.

TABLE 5: The definition of set of color sets and set of variables.

| Set of color sets | Description | Set of variables |
|--|-------------------------------------|------------------|
| Colset NO = int | Identify different transaction type | n |
| Colset NAME = string | Transaction name | p |
| Colset TRANSACTION = product NO * NAME | Information transmitted | |
| Colset ledger = list NAME | Ledger in the blockchain | ps |
| Colset NOs = list NO | List of transactions to be sent | ns |
| Colset range = int with 1..3 | Transaction type randomly generated | k |

blockchain network. The Verify Transaction substitution transition is used to verify the transaction and add the verified transaction to the transaction candidate set. The Execute Consensus transition selects transactions from the transaction candidate set and reaches consensus. The Transaction to add transaction adds consensus-completed transactions to the blockchain. At the same time, for the sending message transaction and sending reputation score transaction, a receiving transaction is generated and added to the list of transactions that need to be sent.

In the subpage model of the Replacement Verify as shown in Figure 8, the Check whether existing transition checks whether transaction already exists in the blockchain. The Check nonce transition checks whether nonce value in the transaction is correct. The Check timestamp transition is used to check whether timestamp in the transaction is correct. The Check signature transition is used to check whether transaction is correctly signed by the sender of the transaction. The Check related transaction transition is used to check whether related transaction information is correct and has been added to the blockchain.

6.1.2. Security Analysis. The number of nodes in VANETs is finite, so the transactions generated by the Generate transition are also finite. There exists an integer M and the number of tokens in any place in the CPN model will not exceed M . Therefore, the CPN is bounded, that is, M -secure. The specific value of M depends on the speed with which transactions are generated and processed in the network. Modeling and simulating the network using CPN tools show that all transitions in the CPN are alive and can be fired indefinitely; that is, there is no dead transition. It can be seen that the workflow of architecture is secure. At the same time, through the analysis of CPN state space, it is found that every sending message transaction and sending reputation score transaction has a corresponding receiving transaction in the blockchain ledger, which ensures the existence of their pairing. It can be seen that the availability of architecture is secure.

In a word, using CPN to model architecture, it can be concluded that the architecture proposed in this paper is security.

6.2. Feasibility Verification of PoW Anti-Spam. In this section, the impact of different difficulty values on PoW time is analyzed by experiments, and the feasibility of PoW anti-spam is verified. The experiment in this paper is carried out in the environment of 2.8 GHz processor and 8 GB memory. At different difficulty values, 500 experiments were carried out independently, and then the average value was taken as the experimental result.

First, this paper verifies the impact of difficulty value on the times to find the nonce. As shown in Figure 9, the abscissa represents difficulty value (ranging from 1 to 24), and the ordinate represents the average times to find the nonce. It can be concluded from Figure 9 that the times for attempting to find a nonce increase exponentially as the difficulty value increases.

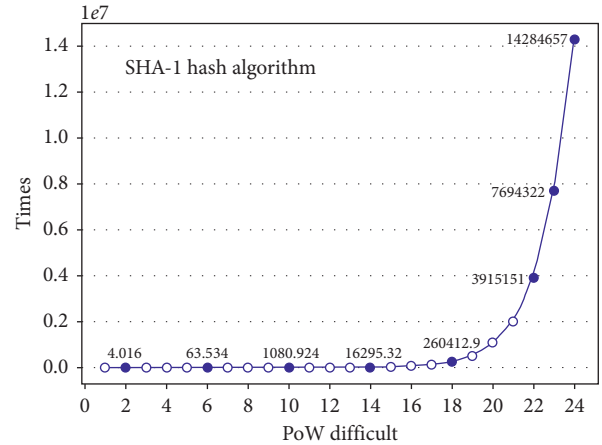


FIGURE 9: The relationship between difficulty and times to find the nonce.

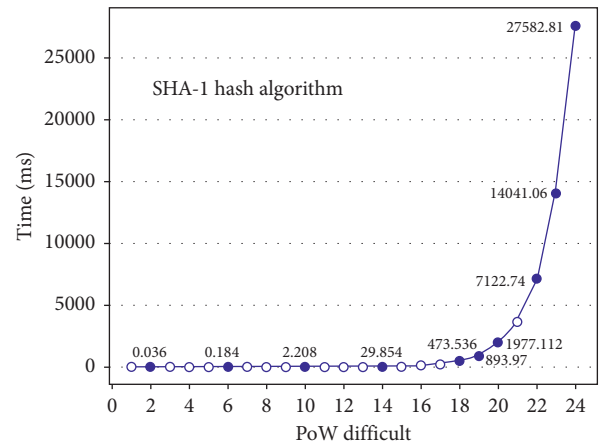


FIGURE 10: The relationship between difficulty and time to find the nonce.

This paper then verifies the impact of difficulty value on time required to find the nonce. As shown in Figure 10, abscissa represents difficulty value (ranging from 1 to 24), and ordinate represents average time required to find the nonce. It can be concluded from Figure 10 that time required to find a nonce increase exponentially as the difficulty value increases. When difficulty value is small, time required to complete the PoW is also short, basically in milliseconds level, which satisfies the needs of VANETs applications. When difficulty value is greater than or equal to 20, time required to complete the PoW reaches the second level. The messages sent by the nodes cannot satisfy the real-time requirements in the VANETs environment, especially for safety-related messages. However, for nodes with high difficulty values, their reputation scores are relatively low, and generated messages are also with low credibility. It can be seen that it is feasible to use PoW anti-spam in this architecture, and it can prevent nodes from performing malicious behaviors without affecting the performance of VANETs.

The average number of times to find nonce is not directly related to the hardware environment but is related to the selected hash algorithm. However, for the same hash algorithm, the average number of times is basically the same for all nodes. The main factor that affects time to find nonce is the computational power of the node. The more computational power, the less time required for the node to complete proof of work. Therefore, in order not to affect the performance of VANETs, it is necessary to dynamically adjust ϑ and μ shown in Section 4.5 according to the computational power of the nodes in the network.

7. Conclusions

This paper proposes a lightweight blockchain architecture using DAG-lattice structure for VANETs, called V-Lattice. In V-Lattice, each node (vehicle or roadside unit) has its own account chain, the transactions they generated can be added to the blockchain asynchronously and parallelly, and resource-constrained vehicles can store the pruned blockchain and execute blockchain related operations normally. In order to encourage more nodes to participate in the blockchain, a reputation-based incentive mechanism is introduced in V-Lattice. At the same time, this paper proposes an asynchronous consensus scheme for transactions generated by nodes and describes how common applications (including node creates accounts, node broadcasts messages, node obtains messages from other nodes, and RSU updates nodes' reputation score) in the VANETs work under this architecture. This paper uses Colored Petri Nets to verify the security of the architecture and verifies the feasibility of PoW anti-spam through experiment. The validation results show that the architecture proposed in this paper is security and it is feasible to prevent nodes from generating malicious behaviors by using PoW anti-spam.

In future work, we will further study how to achieve efficient and asynchronous consensus on transactions under the architecture. At the same time, a blockchain pruning scheme will be designed to ensure that the blockchain can run normally on resource-constrained vehicles.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61862046, in part by the Inner Mongolia Autonomous Region Science and Technology Achievements Transformation Project under Grant CGZH2018124, and in part by the Science and Technology Program of Inner Mongolia Autonomous Region under Grant 2019GG376.

References

- [1] S. Sharma, K. K. Ghanshala, and S. Mohan, "Blockchain-based internet of vehicles (IoV): an efficient secure ad hoc vehicular networking architecture," in *Proceedings of the 2019 IEEE 2nd 5G World Forum (5GWF)*, pp. 452–457, IEEE, Dresden, Germany, September 2019.
- [2] P. K. Sharma, S. Y. Moon, and J. H. Park, "Block-VN: a distributed blockchain based vehicular network architecture in smart city," *Journal of Information Processing Systems*, vol. 13, no. 1, pp. 184–195, 2017.
- [3] M. Awais Hassan, U. Habiba, U. Ghani, and M. Shoaib, "A secure message-passing framework for inter-vehicular communication using blockchain," *International Journal of Distributed Sensor Networks*, vol. 15, no. 2, 2019.
- [4] M. A. Rahman, M. M. Rashid, S. J. Barnes, and S. M. Abdullah, "A blockchain-based secure internet of vehicles management framework," in *Proceedings of the 2019 UK/China Emerging Technologies (UCET)*, pp. 1–4, Glasgow, UK, August 2019.
- [5] X. Wang, P. Zeng, N. Patterson, F. Jiang, and R. Doss, "An improved authentication scheme for internet of vehicles based on blockchain technology," *IEEE Access*, vol. 7, pp. 45061–45072, 2019.
- [6] D. Zheng, C. Jing, R. Guo, S. Gao, and L. Wang, "A traceable blockchain-based access authentication system with privacy preservation in VANETs," *IEEE Access*, vol. 7, pp. 117716–117726, 2019.
- [7] C. Wang, J. Shen, J.-F. Lai, and J. Liu, "B-TSCA: blockchain assisted trustworthiness scalable computation for V2I authentication in VANETs," *IEEE Transactions on Emerging Topics in Computing*, p. 1, 2020.
- [8] F. Gao, L. Zhu, M. Shen, K. Sharif, Z. Wan, and K. Ren, "A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks," *IEEE Network*, vol. 32, no. 6, pp. 184–192, 2018.
- [9] L. Li, J. Liu, L. Cheng et al., "CreditCoin: a privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.
- [10] H. Li, L. Pei, D. Liao, G. Sun, and D. Xu, "Blockchain meets VANET: an architecture for identity and location privacy protection in VANET," *Peer-to-Peer Networking and Applications*, vol. 12, no. 5, pp. 1178–1193, 2019.
- [11] Z. J. Lu, W. C. Liu, Q. Wang, G. Qu, and Z. L. Liu, "A privacy-preserving trust model based on blockchain for VANETs," *IEEE Access*, vol. 6, 2018.
- [12] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, 2019.
- [13] X. Liu, H. Huang, F. Xiao, and Z. Ma, "A blockchain-based trust management with conditional privacy-preserving announcement scheme for VANETs," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4101–4112, 2020.
- [14] E. M. Cho and M. N. S. Perera, "Efficient certificate management in blockchain based internet of vehicles," in *Proceedings of the 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pp. 794–797, Melbourne, Victoria, Australia, May 2020.
- [15] C. Lin, D. He, X. Huang, N. Kumar, and K.-K. R. Choo, "BCPPA: a blockchain-based conditional privacy-preserving authentication protocol for vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2020.

- [16] L. Zhang, M. Luo, J. Li et al., "Blockchain based secure data sharing system for Internet of vehicles: a position paper," *Vehicular Communications*, vol. 16, pp. 85–93, 2019.
- [17] X. Zhang and X. Chen, "Data security sharing and storage based on a consortium blockchain in a vehicular ad-hoc network," *IEEE Access*, vol. 7, pp. 58241–58254, 2019.
- [18] S. K. Dwivedi, R. Amin, S. Vollala, and R. Chaudhry, "Blockchain-based secured event-information sharing protocol in internet of vehicles for smart cities," *Computers & Electrical Engineering*, vol. 86, 2020.
- [19] IOTA: An Open, Feeless Data and Value Transfer Protocol, <https://www.iota.org/>.
- [20] A. Churyumov, "Byteball: a decentralized system for storage and transfer of value," <https://obyte.org/Byteball.pdf>.
- [21] "InterValue: connect, transfer and exchange all digital assets over the world," March 2018, https://www.inve.one/file/InterValue_whitepaper_cn.pdf.
- [22] C. LeMahieu, "Nano: a feeless distributed cryptocurrency network," https://content.nano.org/whitepaper/Nano_Whitepaper_en.pdf.
- [23] JURA: An Ultrafast, Feeless Self-regulated Decentralized Network, <https://docsend.com/view/p9f87gp>.
- [24] B. Xu, T. Agbele, and R. Jiang, "Biometric blockchain: a secure solution for intelligent vehicle data sharing," in *Deep Biometrics*, pp. 245–256, Springer, Cham, Switzerland, 2020.
- [25] A. S. Kulathunge and H. R. O. E. Dayarathna, "Communication framework for vehicular ad-hoc networks using Blockchain: case study of metro manila electric shuttle automation project," in *Proceedings of the 2019 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, pp. 85–90, Colombo, Sri Lanka, March 2019.
- [26] Y. Yuan and F. Y. Wang, "Towards blockchain-based intelligent transportation systems," in *Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2663–2668, Rio de Janeiro, Brazil, November 2016.
- [27] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, "Blockchain: a distributed solution to automotive security and privacy," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 119–125, 2017.
- [28] M. Singh and S. Kim, "Introduce reward-based intelligent vehicles communication using blockchain," in *Proceedings of the 2017 International SoC Design Conference (ISOCC)*, pp. 15–16, Seoul, Republic of Korea, November 2017.
- [29] B. Leiding, P. Memarmoshrefi, and D. Hogrefe, "Self-managed and blockchain-based vehicular ad-hoc networks," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (UbiComp '16)*, pp. 137–140, New York, NY, USA, September 2016.
- [30] T. Jiang, H. Fang, and H. Wang, "Blockchain-based internet of vehicles: distributed network architecture and performance analysis," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4640–4649, 2019.
- [31] T. Zhou, X. Li, and H. Zhao, "DLattice: a permission-less blockchain based on DPoS-BA-DAG consensus for data tokenization," *IEEE Access*, vol. 7, pp. 39273–39287, 2019.
- [32] F. Ayaz, Z. Sheng, D. Tian, G. Y. Liang, and V. Leung, "A voting blockchain based message dissemination in vehicular ad-hoc networks (VANETs)," in *Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, Dublin, Ireland, June 2020.
- [33] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pp. 120–130, New York, NY, USA, October 1999.
- [34] Y. Gilad, R. Hemo, S. M. Micali, G. Vlachos, and N. Zeldovich, "Algorand: scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*, pp. 51–68, New York, NY, USA, October 2017.
- [35] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Advances in Cryptology – EUROCRYPT 2018*, vol. 10821, pp. 66–98, Springer, Cham, Switzerland, 2018.
- [36] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security*, vol. 9, no. 1, pp. 1–30, 2006.

Research Article

Stochastic Adaptive Forwarding Strategy Based on Deep Reinforcement Learning for Secure Mobile Video Communications in NDN

Bowei Hao ¹, Guoyong Wang ², Mingchuan Zhang ¹, Junlong Zhu ¹, Ling Xing ¹, and Qingtao Wu ¹

¹School of Information Engineering, Henan University of Science and Technology, Luoyang 471023, China

²School of Computer and Information Engineering, Luoyang Institute of Science and Technology, Luoyang 471023, China

Correspondence should be addressed to Mingchuan Zhang; zhang_mch@haust.edu.cn

Received 3 January 2021; Revised 16 March 2021; Accepted 31 March 2021; Published 26 April 2021

Academic Editor: Yuanlong Cao

Copyright © 2021 Bowei Hao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Named Data Networking (NDN) can effectively deal with the rapid development of mobile video services. For NDN, selecting a suitable forwarding interface according to the current network status can improve the efficiency of mobile video communication and can also avoid attacks to improve communication security. For this reason, we propose a stochastic adaptive forwarding strategy based on deep reinforcement learning (SAF-DRL) for secure mobile video communications in NDN. For each available forwarding interface, we introduce the twin delayed deep deterministic policy gradient algorithm to obtain a more robust forwarding strategy. Moreover, we conduct various numerical experiments to validate the performance of SAF-DRL. Compared with BR, RFA, SAF, and AFSndn forwarding strategies, the results show that SAF-DRL can reduce the delivery time and the average number of lost packets to improve the performance of NDN.

1. Introduction

Recently, with the development of technology and the increase of mobile devices, the proportion of mobile video services in network communications has increased rapidly. At the same time, users pay more attention to the acquired video content information and no longer pay attention to its storage location. This brings about huge challenges to the current host-based TCP/IP network architecture [1–3]. Although there are many research studies to improve current network performance [4–6], they have not been reformed in essence. To deal with these challenges, Named Data Networking (NDN) [7] is proposed as one of the most potential candidate network architectures in the future.

Because the routing nodes in NDN have the crucial feature of being cached in network, the efficiency of video communication can be effectively improved in NDN [8]. In addition, by decoupling information content and

location, NDN can well support the mobility of video communications [9]. Delay is one of the main factors affecting the efficiency of video transmission services and Quality of Experience (QoE) of video services. To reduce the delay of the video transmission service, a high-performance adaptive forwarding strategy is necessary for NDN [10]. Although the NDN architecture provides a certain degree of security for network communications [7], an effective adaptive forwarding strategy can prevent video communication from being attacked and find trusted video content [11]. Thus, adaptive forwarding is a key issue for secure mobile video communications. When users retrieve video content, they will send the Interest packet to network. After the router receives the Interest packet, it may find many available forwarding interfaces after querying the Forwarding Information Base (FIB). Therefore, according to the current interface network conditions, adaptively selecting a suitable interface for forwarding can greatly improve the efficiency of video

communications. In addition, Yi et al. [12, 13] proved that the forwarding plane in NDN is stateful, where routing only provides a guiding role for adaptive forwarding. Therefore, designing an effective adaptive forwarding strategy for NDN can greatly improve efficiency of secure mobile video communications.

Adaptive forwarding in NDN is a dynamic and complex process. When the router receives the Interest packet, it can select one or more certain available interfaces for forwarding [14–17]. Due to the selection of certain available interface forwarding, these strategies lack the exploration of unknown links and cannot find better links in time, which may cause network load imbalance. Thus, some researchers propose probability-based adaptive forwarding strategies [18–21]. In these strategies, the Interest packet will stochastically select the interface for forwarding according to the forwarding probability of the available interface. However, these strategies are less robust to emergencies in the network, such as network congestion or link failure. Therefore, assigning a suitable and robust forwarding probability to each available forwarding interface is the main research content of designing stochastic adaptive forwarding strategy in NDN.

In recent years, the rise and development of reinforcement learning theory has brought about new ideas for the design of NDN adaptive forwarding strategies [22]. Reinforcement learning aims to obtain the optimal strategy through independent learning through interaction with the environment [23]. In this paper, we use the advantages of reinforcement learning to design a more suitable and robust stochastic adaptive forwarding strategy in NDN. When a user requests mobile video data in NDN, the user cannot obtain all the mobile video data by sending one Interest packet to the server. It is necessary to continuously send multiple Interest packets with a common prefix to request all the mobile video data. We introduce the twin delayed deep deterministic policy gradient (TD3) [24] algorithm to solve this continuous control problem. Then, we take the throughput, delay, and error rate of each interface as the state of the algorithm, the total utility of the node as the reward function of the algorithm, and the forwarding probability of each interface as the action of the algorithm. Through continuous iterative training, an adaptive forwarding strategy with maximum network utility can be finally obtained.

We summarize our main contributions as follows:

We propose an adaptive forwarding framework based on deep reinforcement learning in NDN. This framework can assign a suitable and robust forwarding probability for each available interface.

We propose a stochastic adaptive forwarding strategy for secure mobile video communications based on deep reinforcement learning (SAF-DRL) which introduces the TD3 algorithm into NDN.

We conduct numerical experiments on the SAF-DRL algorithm under different topologies. By comparison with four other adaptive forwarding strategies, the experimental results show that the SAF-DRL can achieve higher network performance.

The rest of the paper is arranged as follows: Section 2 introduces related work on adaptive forwarding strictly in NDN. Section 3 discusses the system model of adaptive forwarding in NDN. Section 4 presents and describes the SAF-DRL algorithm. Section 5 evaluates the performance of the SAF-DRL. Section 6 summarizes this paper and proposes future work.

2. Related Work

In recent years, NDN has received more attention and there are a large number of researchers studying this field. Many of them are making great efforts to study adaptive forwarding strategies. In this section, we review some typical works.

Yi et al. [15] proposed a framework for adaptive forwarding in NDN networks and also proposed an adaptive forwarding strategy based on the color (GREEN, YELLOW, and RED) of the forwarding interface. After this, adaptive forwarding in NDN is mainly divided into two categories. One is the early use of mathematical optimization methods, the most important of which is the adaptive forwarding strategy based on probability, and the other is the use of reinforcement learning method in recent years through continuous iterative training to get the optimal adaptive forwarding strategy.

Probability-based adaptive forwarding has a large number of documents in the early stage. Qian et al. [18] proposed an adaptive forwarding strategy based on probability. This strategy mainly assigns forwarding probability for each available interface and minimizes the delay of Interest packet transmission on each interface as an optimization goal. The objective function is optimized through the ant colony optimization algorithm to finally achieve the optimal adaptive forwarding strategy. Lei et al. [19] proposed a maximizing deviation based probabilistic forwarding strategy. This forwarding strategy comprehensively considers multiple related attributes of the node and by using the maximizing deviation method assigns the optimal weight to each attribute. On this basis, the comprehensive score of each available forwarding interface can be obtained, and this is taken as the forwarding probability of the forwarding interface. Lei et al. [20] proposed an entropy-based probabilistic forwarding strategy. This forwarding strategy uses the entropy weight theory to assign weights among multiple attributes. The node combines its own performance and the assigned weight to calculate the availability of each available interface and then uses this availability as the forwarding probability of the available interface. Posch et al. [21] proposed stochastic adaptive forwarding (SAF). SAF imitates the water pipe system in reality. It can guide and

distribute Interest packets intelligently in network nodes to avoid link congestion. SAF adopts the overpressure valve, takes the throughput of the link as an important measure, divides the Interest packets into satisfied and unsatisfied, and allocates the forwarding probability of each interface, so that the congested nodes can reduce the pressure independently.

As the advantages of reinforcement learning gradually manifest, some researchers use reinforcement learning to find the optimal adaptive forwarding strategy. Yao et al. [25] proposed an adaptive forwarding strategy called SMDPF. This strategy regards the request forwarding in the network as a Semi-Markov Decision Problem (SMDP). Then, based on SMDP theory and considering the randomness of network requests, an optimal adaptive forwarding strategy is designed to deal with the request forwarding by combining Q-learning with artificial neural network. Akinwande [26] proposed an adaptive forwarding strategy based on reinforcement learning and random neural network. Based on the dynamic self-awareness strategy layer, the strategy can reply to the request content quickly through local Content Store (CS). At the same time, it uses *probe* Interest packet and combines it with local routing information to actively seek new available delivery path under controllable degree. Zhang et al. [27] proposed an intelligent forwarding strategy using reinforcement learning. The strategy does not rely on the model programmed in advance but trains a neural network model to select the interface by collecting information from nodes. At the same time, it only learns new decisions by observing the results of past decisions. Zhang et al. [28] proposed an adaptive forwarding strategy using improved Q-learning. The strategy is mainly divided into two phases, exploration and exploitation. In the exploration phase, the information in the network is collected, and then the information is used as the basis to guide forwarding of Interest packets in the exploitation phase.

Probability-based adaptive forwarding can greatly reduce the resource waste caused by deterministically selecting one or more interfaces in mobile video communications. At the same time, it can avoid attacks due to the selection of certain interfaces, thereby improving security. However, adaptive forwarding based on reinforcement learning has higher robustness, especially in the case of link failures. We use the advantages of both and use reinforcement learning to train the forwarding probability assigned to each available forwarding interface, finally obtaining the adaptive forwarding strategy with high robustness in NDN.

3. System Model

In this section, we introduce the system model in detail. We summarize the major variables and expressions, which are depicted in Table 1.

We use a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ to depict the network model. The directed graph consists of two parts, the set of nodes \mathcal{V} and the set of links \mathcal{E} , where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. For each node $i \in \mathcal{V}$, it may maintain several physical interfaces $F_{(i,j)}$, where j is the neighbor node of node i and the tuple $(i, j) \in \mathcal{E}$. We define \mathcal{F}_i as the set of all physical interfaces with node i , $\mathcal{F}_i = \cup_{(i,j) \in \mathcal{E}} F_{(i,j)}$, and N_i is the number of the

interfaces for node i , $N_i = |\mathcal{F}_i|$. For node i , we define $F_{i_+} \in \mathcal{F}_i$ as the set of in-interfaces which receive the Interest packet and $F_{i_-} \in \mathcal{F}_i \setminus F_{i_+}$ as the set of out-interfaces which return the requested data packet or forward the Interest packet to next hop node searched in the FIB. $F_L \notin \mathcal{F}_i$ is the lose-interface, where the Interest packet needs to be lost. As above, we want to learn an adaptive forwarding (AF) strategy for each node. Since the algorithm proposed in this paper only needs local information to train the AF strategy, the algorithm is installed on each node, and no communication is required between the nodes. We only focus on a single node, so we will omit the subscript for the node in the next discussion.

The content catalogue of this system can be labeled as a set \mathcal{K} . For $k \in \mathcal{K}$ represents the content with the common prefix requested by the user, we define $p_k(F_f)$ as the forwarding probability for the interface F_f with the common prefix k and $p_k(F_L)$ as the packet loss rate with prefix k when network congestion or link fails. In this paper, we mainly focus on a common prefix, so we want to omit subscripts for prefix in the remainder of the paper, and we will consider different prefixes in our future work. We show the system model in Figure 1. There are many interfaces for Interest packet to AF for a node in this system. For example, the mobile device (User1 and/or User2) wants to obtain the video /pre1/pre2/n1.mp4, stored in server (Server1 or/and Server2). The mobile device sends Interest packet to router R1. The router R1 has two interfaces (F_1 and F_2) to forward the Interest packet by probability of 2/3 for F_1 and 1/3 for F_2 (confirm forwarding). Then forwarding continues until the Interest packet encounters the requested content. Finally, the data packet with video is returned to the mobile device via the reverse path.

According to the α -fairness [29, 30] model widely used for Network Utility Maximization (NUM), the utility function is defined as

$$U_\alpha(x) = \begin{cases} \log x, & \text{if } \alpha = 1, \\ \frac{x^{1-\alpha}}{1-\alpha}, & \text{otherwise,} \end{cases} \quad (1)$$

where α is positive numbers and fairness tuning parameter. For $\alpha > 0$, the function is strictly nondecreasing. If $\alpha = 1$, the function is proportional fairness and is widely used in NUM. Similar to [30], we define the utility function of interface $f \in \mathcal{F}$ as

$$U(x_f, y_f, z_f) = U_{\alpha_1}(x_f) - \beta U_{\alpha_2}(y_f) - \gamma U_{\alpha_3}(z_f), \quad (2)$$

where x_f, y_f , and z_f are represented as the throughput, delay, and error rate of the f th interface, respectively; β and γ represent the relative importance of the throughput versus delay versus error rate and $\beta, \gamma \in [0, 1]$. Especially if Interest packet has to be discarded, the corresponding utility is defined as 0; that is, $U(x_L, y_L, z_L) = 0$, where x_L, y_L, z_L , respectively, represent the throughput, delay, and error rate of the lose-interface F_L . For the utility function proportional fairness, we set $\alpha_1 = \alpha_2 = \alpha_3 = 1$. The utility function becomes

TABLE 1: Variables and expressions.

| Variable | Expression |
|--------------------|--|
| N | Number of physical interfaces. |
| F_+, F_- | The set of in-interfaces and out-interfaces. |
| F_L | The lose-interface, where the Interest packet needs to be lost. |
| $p(F_f)$ | Probability of forwarding for interface F_f . |
| P_L | Interest packet loss rate. |
| x_f, y_f, z_f | Throughput, delay, and error rate of the interface $f \in \mathcal{F}$. |
| $U(x_f, y_f, z_f)$ | The network utility of interface $f \in \mathcal{F}$. |
| x_L, y_L, z_L | Throughput, delay, and error rate of the lose-interface F_L . |
| $U(x_L, y_L, z_L)$ | The network utility of lose-interface F_L . |
| U_{all} | The total utility of each node. |

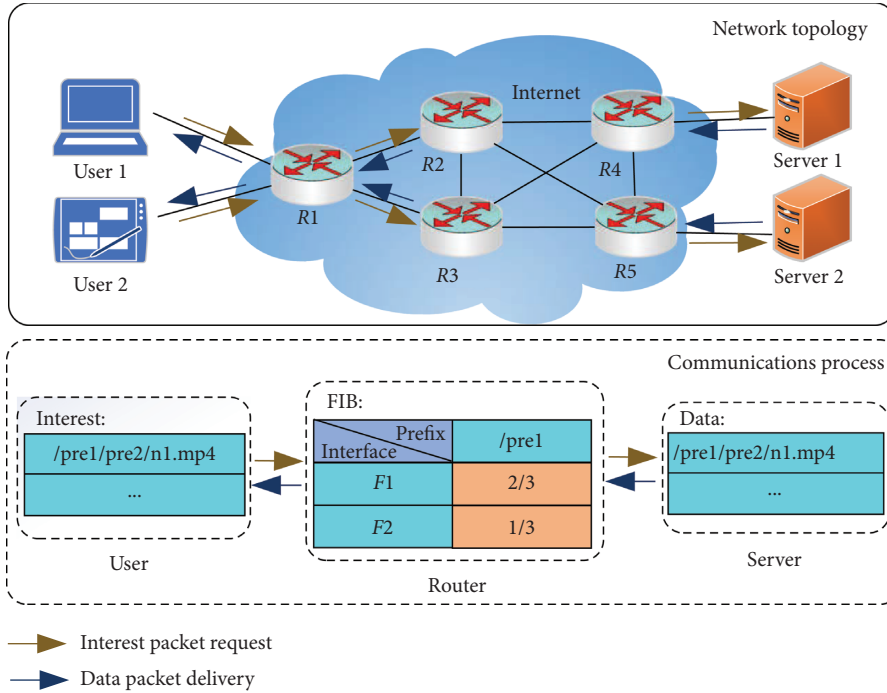


FIGURE 1: The system model.

$$U(x_f, y_f, z_f) = \log x_f - \beta \log y_f - \gamma \log z_f. \quad (3)$$

According to [31], the total utility of each node in the network can be expressed as

$$U_{\text{all}} = \sum_{f=1}^N p(F_f)U(x_f, y_f, z_f) + p(F_L)U(x_L, y_L, z_L). \quad (4)$$

Therefore, the objective of optimizing the AF strategy is to maximize the total network utility U_{all} .

4. Stochastic Adaptive Forwarding Based on Deep Reinforcement Learning

In this section, we first introduce the background knowledge about the TD3 algorithm and then propose the SAF-DRL algorithm and describe the algorithm in detail.

4.1. Background of TD3. In this subsection, in order to better understand the SAF-DRL algorithm, we will introduce some necessary background knowledge about the TD3 algorithm.

For a standard reinforcement learning (RL), the training process is to interact with the environment during each decision epoch and finally obtain the optimal strategy gradually. The specific process is that, at epoch t , the agent observes a state \mathbf{s}_t and selects an action \mathbf{a}_t according to this state. After execution, the environment state will convert from \mathbf{s}_t to \mathbf{s}_{t+1} , and, at the same time, the single-epoch reward $r(\mathbf{s}_t, \mathbf{a}_t)$ given by the environment will be obtained. The goal of reinforcement learning is to find an optimal policy π^* to maximize the discounted future reward $R_t = \sum_{k=t}^T \gamma^{k-t} r(\mathbf{s}_k, \mathbf{a}_k)$, where $\gamma \in [0, 1]$ is a discounted factor. At epoch k , for RL, the goal is to require an optimal policy π_ω under parameterization ω to maximize the expected reward,'

$$J(\omega) = \mathbb{E}_{\mathbf{s}_k \sim p, \pi, \mathbf{a}_k \sim \pi_\omega} (R_t), \quad (5)$$

where p_π is the sampling space of the state.

In order to learn the problem of continuous decision-making, Sutton et al. [32] proposed the policy gradient (PG) method. This method uses a probability distribution function $\mathbb{P}(\mathbf{a}|\mathbf{s}; \omega)$ to represent the optimal policy for each epoch and performs action sampling according to the policy distribution at each epoch to obtain the best action value: $\pi_\omega(\mathbf{a}|\mathbf{s}) = \mathbb{P}(\mathbf{a}|\mathbf{s}; \omega)$. We can use the gradient of reward $\nabla J(\omega)$ to update the parameterized strategy π_ω . Therefore, we can get

$$\nabla_\omega J = \mathbb{E}_{\mathbf{s} \sim p_\pi, \mathbf{a} \sim \pi_\omega} [\nabla_\omega \log \pi_\omega(\mathbf{a}|\mathbf{s}) Q^\pi(\mathbf{s}, \mathbf{a})], \quad (6)$$

where, according to the expected return given by equation (5), $Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}_k \sim p_\pi, \mathbf{a}_k \sim \pi_\omega} (R_t | \mathbf{s}, \mathbf{a})$. Since the process of generating actions is a stochastic process, the last strategy learned is also a stochastic policy. However, since the action space is usually a high-dimensional vector, frequent sampling in the high-dimensional action space is an extremely computationally consuming task. Therefore, Silver et al. [33] proposed deterministic policy gradient (DPG). For each epoch of the action, the determined value is directly obtained through the function μ , $\mathbf{a} = \mu_\omega(\mathbf{s})$. Therefore, we can get

$$\nabla_\omega J = \mathbb{E}_{\mathbf{s} \sim p_\pi} [\nabla_\omega \mu_\omega(\mathbf{s}) \nabla Q^\mu(\mathbf{s}, \mathbf{a}) |_{\mathbf{a}=\mu_\omega(\mathbf{s})}]. \quad (7)$$

In order to deal with high-dimensional input problems, DeepMind introduced deep learning into Q-learning and proposed Deep Q-Network (DQN) [34]. At the epoch k , DQN uses a greedy strategy $\pi(\mathbf{s}_k) = \operatorname{argmax}_{\mathbf{a}_k} Q(\mathbf{s}_k, \mathbf{a}_k)$ and then trains by minimizing loss,

$$\operatorname{Loss}(\omega^Q) = \mathbb{E} [\operatorname{Tar}_k - Q(\mathbf{s}_k, \mathbf{a}_k | \omega^Q)], \quad (8)$$

where Tar_k is the target Q value, which can be expressed as

$$\operatorname{Tar}_k = r(\mathbf{s}_k, \mathbf{a}_k) + \gamma Q(\mathbf{s}_{k+1}, \pi(\mathbf{s}_{k+1} | \omega^\pi) | \omega^Q). \quad (9)$$

For continuous control problems, the actor-critic method [35] is often used to solve the problem. This method usually uses the policy gradient method to find the optimal policy. DeepMind combined DQN and DPG and proposed a new actor-critic method, deep deterministic policy gradient (DDPG) [36]. The training of critic network in DDPG is based on the DQN method, and the training of actor network is based on the DRL method. The specific update formula is

$$\nabla_{\omega^\mu} J = \mathbb{E} \left[\nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a} | \omega^Q) \Big|_{\mathbf{s}=\mathbf{s}_k, \mathbf{a}=\mu(\mathbf{s}_k)} \cdot \nabla_{\omega^\mu} \mu(\mathbf{s} | \omega^\mu) \Big|_{\mathbf{s}=\mathbf{s}_k} \right]. \quad (10)$$

Although DDPG has achieved certain success, there are still problems such as overestimation of Q value and excessive variance. Therefore, Fujimoto et al. [24] proposed an improved DDPG algorithm TD3. The first point of improvement is to eliminate the problem of overfitting. TD3 introduces the idea of Double DQN [37] and uses two critic networks. In the practice of actor-critic algorithm, due to the high similarity between the current network and the target network, independent evaluation cannot be made. In order to solve this problem, Double DQN uses this update method:

$$\begin{aligned} y_1 &= r + \eta \widehat{Q}(s', \pi(s' | \omega^{\phi_1}) | \widehat{\omega}_1^q), \\ y_2 &= r + \eta \widehat{Q}(s', \pi(s' | \omega^{\phi_2}) | \widehat{\omega}_2^q), \end{aligned} \quad (11)$$

where $\pi(\cdot | \omega^{\phi_1})$ and $\pi(\cdot | \omega^{\phi_2})$ are optimized with $\widehat{Q}(\cdot | \widehat{\omega}_1^q)$ and $\widehat{Q}(\cdot | \widehat{\omega}_2^q)$, respectively. After practice, it is found that the final effects of the two actor networks are relatively similar, so one actor network is selected. The update goals of both critic networks are the same, $y_2 = y_1$. Because actor network will choose high evaluations, the overestimated ones will gradually accumulate, so when choosing a smaller evaluation, the final goal can be expressed as

$$y_1 = r + \eta \min_{i=1,2} \widehat{Q}(s', \pi(s' | \omega^{\phi_i}) | \widehat{\omega}_i^q). \quad (12)$$

In order to solve the problem of excessive variance, TD3 introduced the idea of delaying strategy updates. This is to adjust the update frequency of critic network to be a bit higher than that of actor network. Solve the problem that the DQN is constantly updated, which may cause blind iteration of actor network. When calculating the Q function, a certain action is randomly selected within a certain range to achieve smooth strategy, so as to get rid of the influence of false peaks. In this way, the problem of incorrect strategy caused by inaccurate Q function in DDPG can be solved.

4.2. SAF-DRL Algorithm. In this subsection, we present the stochastic adaptive forwarding strategy based on DRL (SAF-DRL) for secure mobile video communications in NDN. For all DRL approaches, the state space, the action space, and the reward function are important components:

STATE: the state space mainly consists of three parts: throughput, delay, and error rate of each interface in the network. Therefore, the state at epoch t is $\mathbf{s}_t = [(x_t^1, y_t^1, z_t^1), (x_t^2, y_t^2, z_t^2), \dots, (x_t^N, y_t^N, z_t^N)]$.

ACTION: the action is defined as the adaptive forwarding probability of each interface and the Interest packet loss rate p_L for specific content. Therefore, the action at epoch t is $\mathbf{a}_t = [(p_t(F_1)), (p_t(F_2)), \dots, (p_t(F_N)), (p_t(F_L))]$, where $p_t \in [0, 1]$ and $\sum_{j=1}^N p_t(F_j) + p_t(F_L) = 1$.

REWARD: the reward function is the objective of AF strategy, which is the total utility of each node in the network. Formally, $r_t = \sum_{f=1}^N p_t(F_f) U_t(x_f, y_f, z_f) + p_t(F_L) U_t(x_L, y_L, z_L)$.

Please note that the design of the state space, action space, and reward function will seriously affect the performance of the SAF-DRL algorithm. Our design is based on the full consideration of mobile video communications and can be closer to the real situation. Moreover, the probability selection interface can improve the security of mobile video communications, especially when encountering link failures or congestion; and reinforcement learning shows higher robustness than mathematical calculations, thus ensuring the performance of mobile video communications.

Since each node receives a large number of Interest packets to request mobile video content, we assume that the

Interest packets received are continuous. Nodes need to continuously make forwarding decisions for these Interest packets. At the same time, in order to make the forwarding probability assigned to each available interface in mobile video communication have high robustness, we use the TD3 algorithm in the actor-critic algorithm. In addition, as the TD3 algorithm is also a deep reinforcement learning [34] algorithm, it can deal with the high-dimensional problems caused by a large number of entries in FIB in reality.

The situation faced by all interfaces of the router constitutes the training environment of DRL Agent. We collect the information of throughput, delay, and error rate for each interface, which constitutes the Agent's state space. This information is mainly composed of local storage in the forwarding process and does not communicate with other nodes in the network. The Interest packet forwarding probability of each interface in the node constitutes the action space of the Agent. Then actions are performed to update the forwarding probability of the interface, and finally a more suitable and robust forwarding probability is got by training. The suitable and robust forwarding probability on the interface can effectively improve the forwarding efficiency, thereby improving the overall performance of the node. As for the reward, it has been discussed in Section 3. We show the detailed framework of SAF-DRL in Figure 2. For the available forwarding interface of the node, we collect the local information of throughput, delay, and loss rate as the states. Then the Agent uses the TD3 method to train the collected information. Through training, we get an AF strategy that maximizes rewards.

We propose the SAF-DRL algorithm as Algorithm 1. First the algorithm initializes two critic networks $Q(\mathbf{s}, \mathbf{a}|\omega_1^q)$ and $Q(\mathbf{s}, \mathbf{a}|\omega_2^q)$ and one actor network $\mu(\mathbf{s}|\omega^u)$ (line 1), and the parameters ω_1^q , ω_2^q , and ω^u are the weights of two critic networks and the actor network, respectively. The corresponding target networks are initialized as $Q(\mathbf{s}, \mathbf{a}|\omega_1^q)$, $Q(\mathbf{s}, \mathbf{a}|\omega_2^q)$, and $\mu(\mathbf{s}|\omega^u)$ (line 2). The parameters $\hat{\omega}_1^q$, $\hat{\omega}_2^q$, and $\hat{\omega}^u$ are the weights of the target network, and their values are defined as ω_1^q , ω_2^q , and ω^u . In order to enable the target network to be updated slowly, a delayed update method is used. On the basis of soft update, the target networks are updated with the update rate φ after every d critic update (line 16). We use the uniform distribution $U(0, |F_-|)$ to initialize forwarding probability of each available forwarding interface, $p(F_-) = (1/|F_-|)$, where $|F_-|$ is the number of the out-interfaces, and we initialize the Interest packet loss rate to 0, $p(F_L) = 0$ (line 4).

We use replay buffer B to store the transition samples and we initialize it in line 3. We first store the experience gained through interaction with the environment in B (lines 6–9) and then we train the actor network and critic network according to sample M random transition samples from B (lines 10–16). The function of the clip($\mathcal{N}(0, \hat{\sigma}), -k, k$) is that the value range of $\mathcal{N}(0, \hat{\sigma})$ is limited between $-k$ and k . When the value is less than $-k$, the value is equal to $-k$, and when the value is greater than k , the value is equal to k (line 11). Calculate the minimum Q value of the two critic networks (line 12) and compute the critic update by minimizing commonly used mean square error loss (line 13). The computation of the actor network update uses the DPG approach [33] (line 15). According to the SAF-DRL

algorithm, we can get an optimal Interest packet forwarding probability for each available interface in the node. Finally, we can get an adaptive forwarding strategy that maximizes network utility.

5. Numerical Experiment

We conducted numerical experiments on SAF-DRL method in the NDN environment. The node has certain computing capabilities and can adaptively forward requests for users. At the same time, the node can move to a certain extent, and the main content requested is the video service. Thus, we use one node in the network as an agent to study. Then, a comparison is performed with the other four adaptive forwarding strategies in terms of delivery time, the average number of lost packets, load balancing factor, and hop count.

We use the Python language to generate three topologies based on Erdős-Rényi (ER) [38] model, Barabási-Albert (BA) [39] model, and random model, as shown in Figure 3. Each topology is composed of 100 nodes. The possibility of creating links between two nodes in ER topology is set to 0.08, one edge is added between two nodes in BA topology at a time, and the distance threshold between two nodes in the random topology is set to 0.15. For each link of the three topologies, we allocate bandwidth of 1 Mbps to 5 Mbps, and the delay of each link is within [10 ms, 30 ms]. We randomly selected 10 nodes in the network as the users, 10 nodes as servers, and the rest as routers. In order to better study the adaptive forwarding strategy, we set the CS capacity on each node to 0. We set $\beta = \gamma = 1$ to balance the importance of throughput, delay, and error rate. The experiment has gone through 1500 time-slots' iterative training every cycle, and finally we take the average value through multiple experiments. We run and train SAF-DRL algorithm on Windows 10 operating system with Intel (R) Core (TM) i5 2.4 Ghz CPU with 8 GB memory.

To explore the reward convergence of different agents under different network topologies, we adopted three topologies of ER topology, BA topology, and random topology and performed experiments on five agents (1, 10, 30, 50, and 70) on each topology. In Figure 4, we can see that, under the same topology, although all agents converge at different speeds, they eventually converge to approximately the same stable value. Because there are certain differences among the three topologies (such as the degree of connectivity), the final stable convergence values obtained by the different topologies are not completely equal, but the difference among the convergence values is very small. This proves that our scheme is convergent and can be used in different topologies. Therefore, in the follow-up experiments, we only explore the comparative experiments under ER topology, and the trend is the same in other topologies.

In order to evaluate the performance of our algorithm in many ways, we compare our SAF-DRL algorithm with four other adaptive forwarding strategies:

BestRoute (BR) [14]: interest packets are forwarded through the available interface with the lowest cost (e.g., hop count).


```

1 Randomly initialize critic network  $Q(\mathbf{s}, \mathbf{a}|\omega_1^q)$ ,  $Q(\mathbf{s}, \mathbf{a}|\omega_2^q)$  and actor network  $\mu(\mathbf{s}|\omega^u)$  with random parameters  $\omega_1^q$ ,  $\omega_2^q$  and  $\omega^u$  respectively;
2 Initialize target critic network  $\tilde{Q}(\mathbf{s}, \mathbf{a}|\tilde{\omega}_1^q)$ ,  $\tilde{Q}(\mathbf{s}, \mathbf{a}|\tilde{\omega}_2^q)$  and target actor network  $\tilde{\mu}(\mathbf{s}|\tilde{\omega}^u)$  with parameters  $\tilde{\omega}_1^q = \omega_1^q$ ,  $\tilde{\omega}_2^q = \omega_2^q$  and  $\tilde{\omega}^u = \omega^u$  respectively;
3 Initialize replay buffer  $B$ ;
4 Initialize the forwarding probability of the out-interface  $F_{-}$ , and  $p(F_L) = 0$ ;
5 Receive the observed initial state  $\mathbf{s}_1$ ; /** Decision Epoch ** /
6 for  $t = 1$  to  $T$  do
7 Obtain the action  $\mathbf{a}_t = \mu(\mathbf{s}_t|\omega^u) + \mathcal{N}_t$  by the current policy  $\mu(\mathbf{s}_t|\omega^u)$  and exploration noise  $\mathcal{N}_t \sim \mathcal{N}(0, \sigma)$ ;
8 Execute action  $\mathbf{a}$ , receive reward  $r$  and observe next state  $\mathbf{s}$ ;
9 Store transition sample  $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$  in  $B$ ; /** Training Transition Sampling ** /
10 Sample a mini-batch of  $M$  transition  $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$  from  $B$ ;
11 Execute action  $\hat{\mathbf{a}} = \tilde{\mu}(\mathbf{s}|\tilde{\omega}^u) + \mathcal{N}$ , where the  $\mathcal{N} \sim \text{clip}(\mathcal{N}(0, \hat{\sigma}), -k, k)$ ;
12 Compute the Q value of critic network:  $y = \begin{cases} r, & \text{if } j = M \\ r + \eta \min_{i=1,2} \tilde{Q}(\mathbf{s}', \hat{\mathbf{a}}|\tilde{\omega}_i^q), & \text{otherwise} \end{cases}$ 
13 Update the critic  $\omega_i^q$  by minimizing the loss:  $\text{Loss} = \min_{\omega_i^q} (1/M) \sum (y - Q(\mathbf{s}, \mathbf{a}|\omega_i^q))^2$ 
14 if  $t \bmod d$  then
15 Compute the actor update by policy gradient:
 $\nabla_{\omega^u} J = (1/M) \sum \nabla_{\omega^u} \mu(\mathbf{s}) \cdot \nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}|\omega_i^q)|_{\mathbf{a}=\mu(\mathbf{s})}$ ;
/** Target Network Update ** /
16 Update the parameters of the target networks:
 $\tilde{\omega}_i^q \leftarrow \varphi \omega_i^q + (1 - \varphi) \tilde{\omega}_i^q$ ;
 $\tilde{\omega}^u \leftarrow \varphi \omega^u + (1 - \varphi) \tilde{\omega}^u$ ;
17 end if
18 end for

```

ALGORITHM 1: SAF-DRL for secure mobile video communications in NDN.

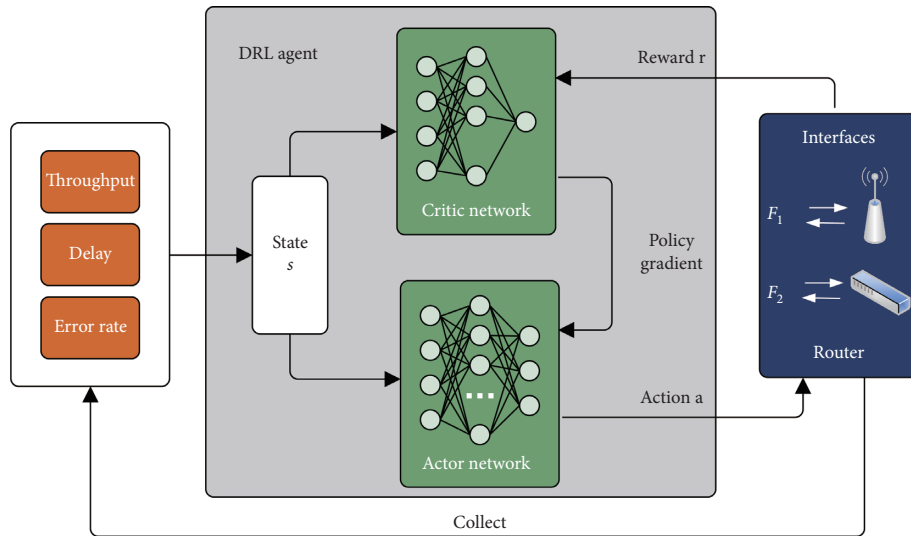


FIGURE 2: The framework of SAF-DRL.

Stochastic Adaptive Forwarding (SAF) [21]: interest packets are forwarded through the interface with the largest throughput-based measurement.

Adaptive Forwarding Strategy in NDN (AFSndn) [28]: interest packets are forwarded through the interface with the lowest delay.

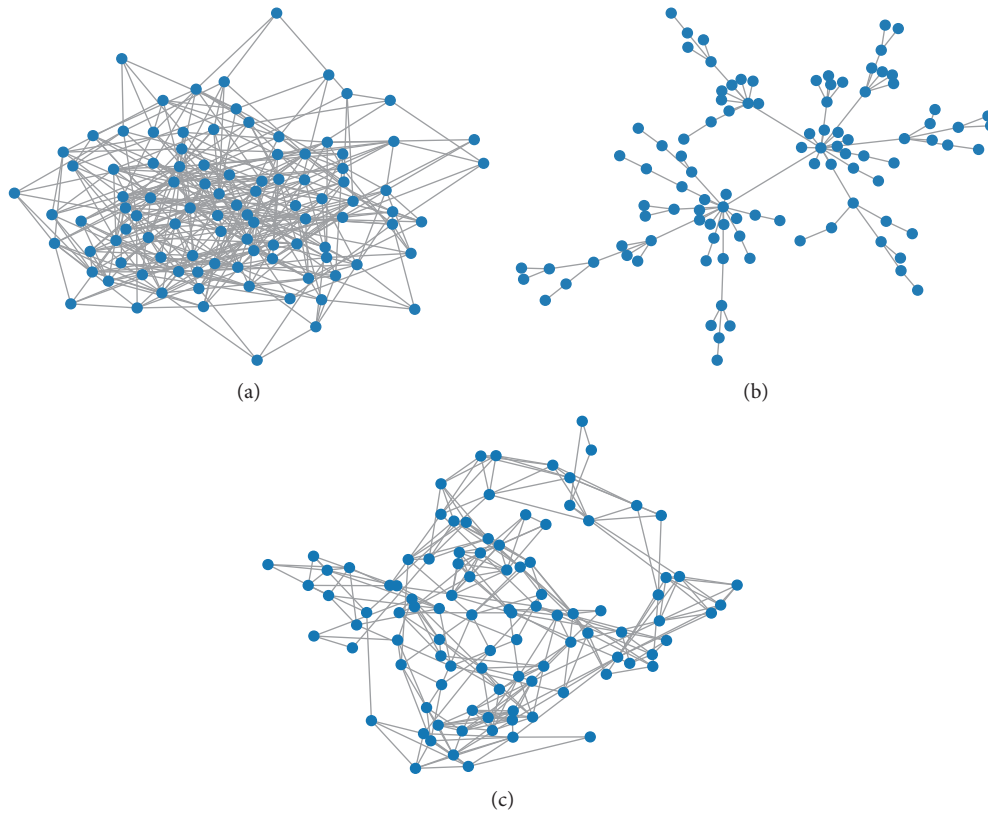


FIGURE 3: The three different network topologies. (a) The ER topology. (b) The BA topology. (c) The random topology.

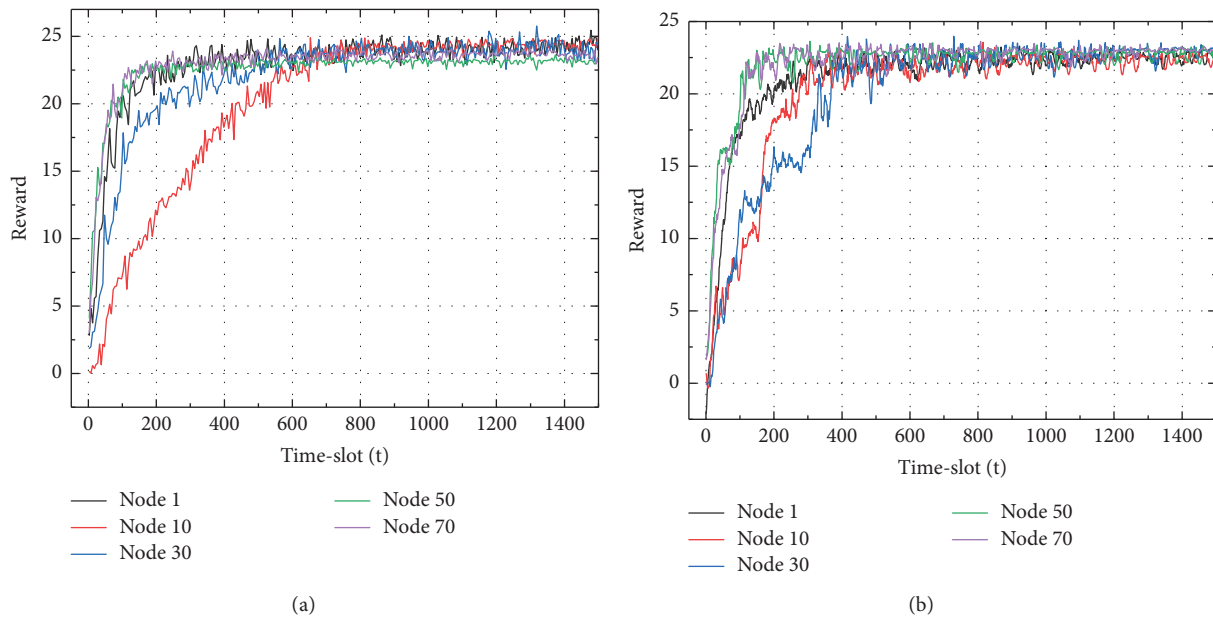


FIGURE 4: Continued.

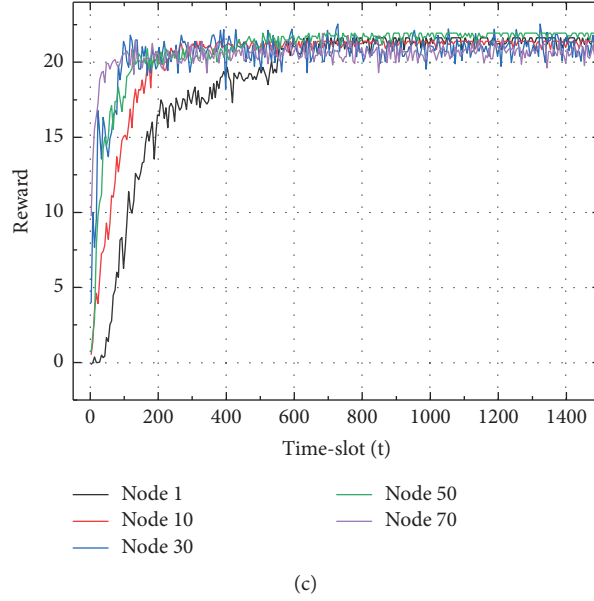


FIGURE 4: Main theoretical results under different topologies. (a) The local reward of different agents (1, 10, 30, 50, and 70) under the ER topology. (b) The local reward of different agents under the BA topology. (c) The local reward of different agents under the random topology.

Request Forwarding Algorithm (RFA) [40]: interest packets are forwarded through the interface with the least count of pending Interest packets.

We mainly compare our algorithm with other algorithms in four aspects.

5.1. Delivery Time. The delivery time is mainly the average time it takes for the Interest packet to find the specific content it requests. The delivery time can be specifically defined as

$$D = \frac{\sum_{i=1}^K (\text{get}_i - \text{send}_i)}{K}. \quad (13)$$

Here, send_i represents the moment when the i -th Interest packet is sent, get_i represents the moment when the target node receives the i -th Interest packet, and K represents the total number of Interest packets requested.

5.2. The Average Number of Lost Packets. The average number of lost packets indicates the average number of Interest packets lost due to other reasons (e.g., not finding the target node or network congestion) during all episodes. The average number of lost packets can be specifically defined as

$$\text{Lost} = \frac{\sum_{i=1}^T \text{lost}_i}{T}. \quad (14)$$

Here, lost_i represents the number of Interest packets lost in the i -th episode and T represents the number of episodes.

5.3. Load Balancing Factor. The load balancing factor represents the degree of dispersion of the number of Interest packets forwarded by each node in the network. We use

coefficient of variation for calculation, so the load balancing factor can be specifically defined as

$$Cv(\text{NI}) = \frac{\text{stdev}[\text{NI}(v)]}{\mathbb{E}[\text{NI}(v)]}. \quad (15)$$

Here, $\text{NI}(v)$ represents the number of Interest packets forwarded by the v node and $\text{stdev}[\text{NI}(v)]$ represents the standard deviation of $\text{NI}(v)$.

5.4. Hop Count. The hop count represents the average number of hops experienced by all Interest packets when they find their target node. The hop count can be specifically defined as

$$H_c = \frac{\sum_{i=1}^K h_i}{K}. \quad (16)$$

Here, h_i represents the number of hops taken by the i -th Interest packet before finding the target node.

We describe in detail the performance of each aspect as follows.

5.4.1. Delivery Time. From Figure 5, we can see the delivery time of the five algorithms under different bandwidth (1 Mbps, 3 Mbps, and 5 Mbps). The delivery time of SAF-DRL is lower than the delivery times of the other four algorithms. This is because the SAF-DRL algorithm uses delay as one type of the link status information, and delay is also one of the indicators of the reward function, which can minimize the delay. Therefore, the delay of SAF-DRL is lower than those of SAF, RFA, and BR. Although the AFSndn algorithm also considers the delay information as the indicator of the reward function, the AFSndn algorithm needs to spend a certain amount of time in the early stage of forwarding for exploration. At the same time, because the

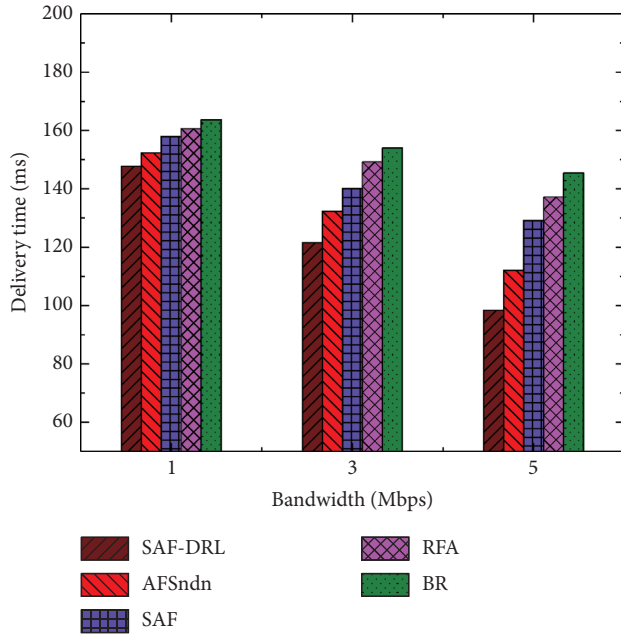


FIGURE 5: Comparison of delivery time with four other algorithms in different bandwidths (1 Mbps, 3 Mbps, and 5 Mbps).

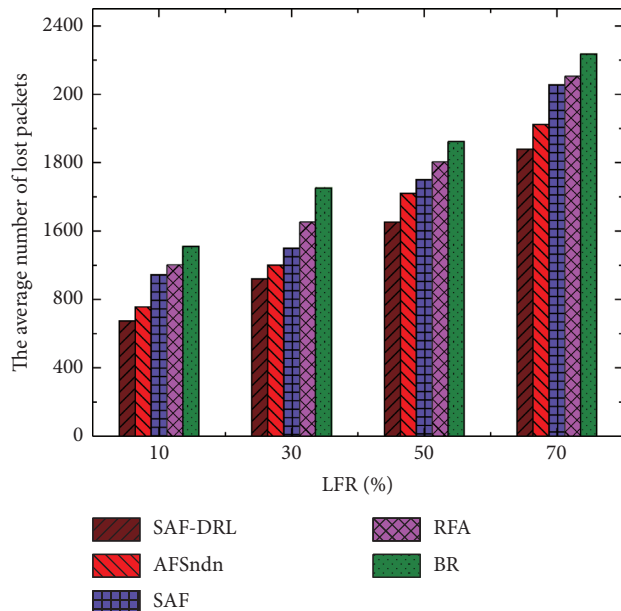


FIGURE 6: Comparison of the average number of lost packets with four other algorithms in different LFR (10%, 30%, 50%, and 70%).

AFSndn algorithm uses Q-learning in reinforcement learning and Q-learning has certain limitations in processing high-dimensional data, it will increase the delivery time to be higher than that of our SAF-DRL algorithm as the number of entries in the FIB increases. Because the BR algorithm selects a link for forwarding, causing network congestion to exceed other algorithms, its delivery time becomes the longest. The RFA algorithm can avoid link congestion through load balancing, which can reduce the delivery time to a certain extent.

5.4.2. The Average Number of Lost Packets. Figure 6 shows the average number of lost packets of the five algorithms under different Link Failure Rate (LFR) (10%, 30%, 50%, and 70%). As can be seen from the figure, with the gradual increase of LFR, the average number of lost packets of the five algorithms is increasing. But SAF-DRL algorithm has always had a lower average number of lost packets, among which the BR algorithm has the largest average number of lost packets. This is because the BR algorithm uses the least hop count as the forwarding basis, which may cause network congestion and eventually may discard a large number of Interest packets. The RFA algorithm only uses the count of pending Interest packets as the basis for forwarding probability. Although network congestion can be avoided as much as possible, interfaces with a small number of pending Interest packets may have poor link status, so the number of lost Interest packets is only lower than that in the BR algorithm. The SAF algorithm considers information such as link throughput and can select an effective interface for forwarding, thereby reducing the number of lost packets. The AFSndn algorithm is more robust than the SAF algorithm through reinforcement learning training. However, due to the large number of Interest packets sent in the previous exploration phase, only a few are effective, and a large number of unused Interest packets are discarded. The SAF-DRL algorithm has relatively the lowest average number of lost packets, because it considers multiple different types of link state information, and training through reinforcement learning has high robustness. At the same time, each interface can be assigned a higher efficiency forwarding probability, which reduces the average number of lost Interest packets.

5.4.3. Load Balancing Factor. Figure 7 shows the results of the load balancing factor under different LFR. It can be seen from the figure that, in the case of four link failures, the SAF-DRL algorithm has a lower load balancing factor. When the user retrieves the content, the BR algorithm only considers the shortest path for forwarding. With the increase of LFR, the congestion on this link intensifies, and then the resources on other links are idle, making the network load unbalanced and the load balancing ability is poor. The SAF algorithm selects links for forwarding. After the Interest packet cannot be satisfied, the SAF algorithm will distribute the traffic on the failed link to other links according to throughput-based measure, which can appropriately improve the load capacity of the network. AFSndn is based on the information in the early exploration phase, and when guiding the forwarding of Interest packets, it tries to avoid network congestion, ensuring the load capacity of the network. Compared with the SAF algorithm and AFSndn algorithm, the SAF-DRL algorithm is more robust due to the reinforcement learning training, which makes the forwarding probability distribution allocated on each available forwarding interface more robust, especially when the link fails. The RFA algorithm has the lowest load balancing factor, because RFA algorithm uses the count of pending Interest packets as the reference basis for forwarding. The count of pending Interest packets can

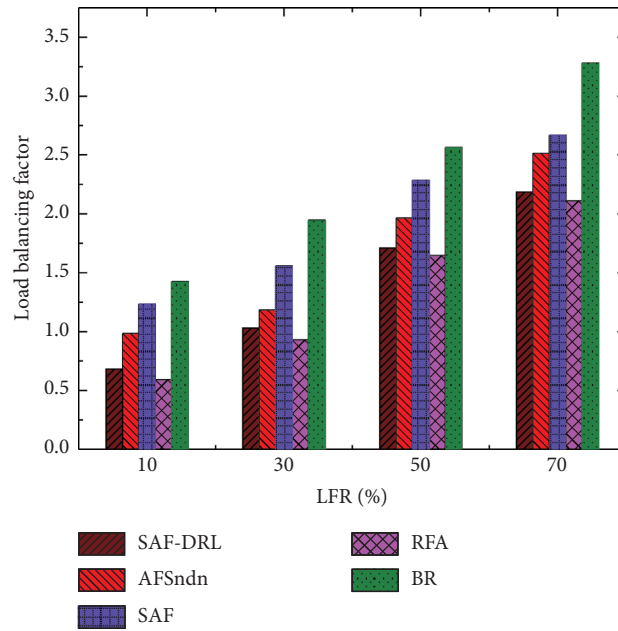


FIGURE 7: Comparison of load balancing factor with four other algorithms in different LFR (10%, 30%, 50%, and 70%).

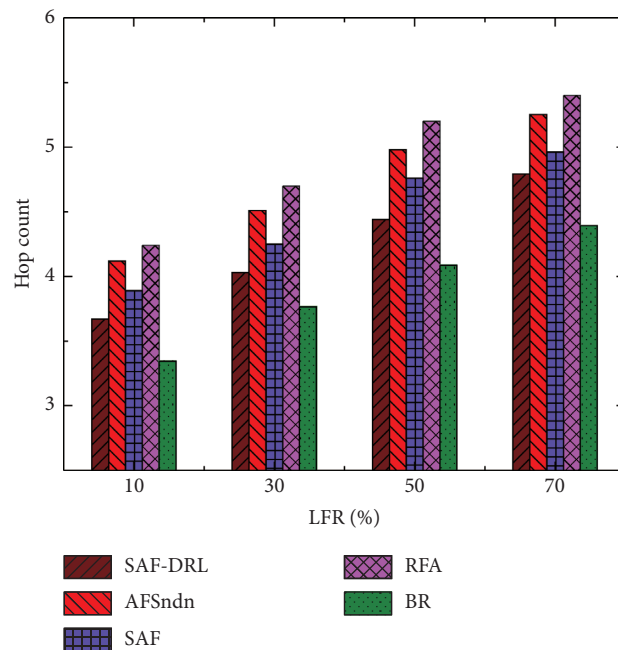


FIGURE 8: Comparison of hop count with four other algorithms in different LFR (10%, 30%, 50%, and 70%).

reflect the load status in a period of time in the future, so RFA algorithm can effectively balance the load with the lowest load balancing factor.

5.4.4. Hop Count. Figure 8 shows the results of the average hop count. It shows that the average hop count of the BR algorithm is the lowest, and the average hop count of the RFA algorithm is the highest. The average hop counts of the SAF algorithm, AFSndn algorithm, and SAF-DRL algorithm

are between the two, and the average hop count of AFSndn algorithm is higher than those of SAF algorithm and SAF-DRL algorithm. This is mainly because the BR algorithm mainly considers forwarding through the least hop count and does not consider other indicators, so the average hop count of the BR algorithm is the lowest. However, the RFA algorithm only considers the count of pending Interest packets and does not care about delaying this information, so its hop count is the highest. As for the AFSndn algorithm, in the early exploration phase, Interest packets will be

forwarded through all available interfaces, and there are a certain number of links with very long paths, which leads to higher average hop count. Since the SAF algorithm and the SAF-DRL algorithm are adaptive forwarding strategies based on probability, the forwarding probability of selecting a link with better performance is greater, but the link with better performance is not necessarily the shortest. The SAF algorithm only uses throughput as the measure. However, the SAF-DRL algorithm takes delay and other information into account, which is equivalent to considering the length of the link to a certain extent, so that it can find the target node faster with fewer hops.

6. Conclusion and Future Work

In this paper, we have proposed stochastic adaptive forwarding strategy based on deep reinforcement learning (SAF-DRL), a novel adaptive forwarding strategy for secure mobile video communications in NDN. SAF-DRL can forward each Interest packet with a common prefix according to the forwarding probability. To obtain a more robust forwarding probability on each available interface, we have also introduced the twin delayed deep deterministic policy gradient to NDN for adaptive forwarding. Through numerical experiments, the results have shown that SAF-DRL algorithm can achieve final stability under ER topology, BA topology, and random topology. Compared with BR, RFA, SAF, and AFSdn, SAF-DRL has obvious advantages in delivery time and the average number of lost packets. Since we only considered the same video prefix in this paper, in future work, we will consider the priority between different video content prefixes requested by mobile devices; and different applications require different weights for different status information. For example, the transmission of live broadcast service requires lower delay. We will combine the content priority and the weight of the interface status to improve the security and efficiency of mobile video communications.

Data Availability

No data were used to support this study.

Conflicts of Interest

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

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant nos. 61971458 and 61976243, in part by the Leading Talents of Science and Technology in the Central Plain of China under Grant no. 214200510012, in part by the basic research projects in the University of Henan Province under Grant no. 19zx010, and by the Key Project of the Education Department of Henan Province under Grant no. 20A520011.

References

- [1] M. Zhang, Y. Zhou, W. Quan, J. Zhu, R. Zheng, and Q. Wu, "Online learning for IoT optimization: a frank-wolfe adam-based algorithm," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8228–8237, 2020.
- [2] F. Song, Y.-T. Zhou, L. Chang, and H.-K. Zhang, "Modeling space-terrestrial integrated networks with smart collaborative theory," *IEEE Network*, vol. 33, no. 1, pp. 51–57, 2019.
- [3] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, V. C. M. Leung, and MASM, "MASM: a multiple-algorithm service model for energy-delay optimization in edge artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4216–4224, 2019.
- [4] F. Song, Z. Ai, Y. Zhou, I. You, K.-K. R. Choo, and H. Zhang, "Smart collaborative automation for receive buffer control in multipath industrial networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1385–1394, 2020.
- [5] C. Xu, J. Zhu, and D. O. Wu, "Decentralized online learning methods based on weight-balancing over time-varying digraphs," *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–13, 2018.
- [6] F. Song, Y.-T. Zhou, Y. Wang, T.-M. Zhao, I. You, and H.-K. Zhang, "Smart collaborative distribution for privacy enhancement in moving target defense," *Information Sciences*, vol. 479, pp. 593–606, 2019.
- [7] L. Zhang, A. Afanasyev, J. Burke et al., "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.
- [8] M. Zhang, B. Hao, F. Song, M. Yang, J. Zhu, and Q. Wu, "Smart collaborative video caching for energy efficiency in cognitive content centric networks," *Journal of Network and Computer Applications*, vol. 158, p. 102587, 2020.
- [9] M. Meddeb, A. Dhraief, A. Belghith et al., "AFIRM: adaptive forwarding based link recovery for mobility support in NDN/IoT networks," *Future Generation Computer Systems*, vol. 87, pp. 351–363, 2018.
- [10] Y. Ye, B. A. Lee, R. Flynn, N. Murray, and Y. Qiao, "HLAF: heterogeneous-latency adaptive forwarding strategy for peer-assisted video streaming in NDN," in *Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC)*, pp. 657–662, Heraklion, Greece, July 2017.
- [11] Z. Rezaeifar, J. Wang, H. Oh, S.-B. Lee, and J. Hur, "A reliable adaptive forwarding approach in named data networking," *Future Generation Computer Systems*, vol. 96, pp. 538–551, 2019.
- [12] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013.
- [13] C. Yi, J. P. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the role of routing in named data networking," in *Proceeding of the 1st International Conference on Information-Centric Networking (ICN)*, pp. 27–36, Paris, France, August 2014.
- [14] A. Afanasyev, J. Shi, B. Zhang et al., "NFD developer's guide," <https://named-data.net/publications/techreports/nfd-developer-guide/> Technical Report NDN-0021.
- [15] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 3, pp. 62–67, 2012.
- [16] Y. Ren, Z. Li, J. Li et al., "A dynamic multi-path forwarding strategy for information centric networks," in *Proceedings of the 21st IEEE International Conference on High Performance*

- Computing and Communications; (HPCC/SmartCity/DSS)*, pp. 2495–2501, Sydney, Australia, September 2019.
- [17] B. Abdelkader, M. R. Senouci, and B. Merabti, “Parallel multipath forwarding strategy for named data networking,” in *Proceedings of the 13th International Joint Conference on E-Business and Telecommunications (ICETE)*, pp. 36–46, SciTePress, Lisbon, Portugal, July 2016.
- [18] H. Qian, R. Ravindran, G. Wang, and D. Medhi, “Probability-based adaptive forwarding strategy in named data networking,” in *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 1094–1101, Ghent, Belgium, May 2013.
- [19] K. Lei, J. Yuan, and J. Wang, “MDPF: an NDN probabilistic forwarding strategy based on maximizing deviation method,” in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, San Francisco, CA, USA, November 2015.
- [20] K. Lei, J. Wang, and J. Yuan, “An entropy-based probabilistic forwarding strategy in named data networking,” in *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*, pp. 5665–5671, London, UK, June 2015.
- [21] D. Posch, B. Rainer, and H. Hellwagner, “SAF: stochastic adaptive forwarding in named data networking,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1089–1102, 2017.
- [22] N. C. Luong, D. T. Hoang, S. Gong et al., “Applications of deep reinforcement learning in communications and networking: a survey,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.
- [23] S. Çalisir and M. K. Pehlivanoglu, “Model-free reinforcement learning algorithms: A survey,” in *Proceedings of the 27th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, Trabzon, Turkey, April 2019.
- [24] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 1582–1591, Stockholm, Sweden, July 2018.
- [25] J. Yao, B. Yin, and X. Tan, “A SMDP-based forwarding scheme in named data networking,” *Neurocomputing*, vol. 306, pp. 213–225, 2018.
- [26] O. Akinwande, “Interest forwarding in named data networking using reinforcement learning,” *Sensors*, vol. 18, no. 10, pp. 3354–3373, 2018.
- [27] Y. Zhang, B. Bai, K. Xu, and K. Lei, “IFS-RL: An intelligent forwarding strategy based on reinforcement learning in named-data networking,” in *Proceedings of the 2018 Workshop on Network Meets AI & ML*, pp. 54–59, NetAI@SIGCOMM, Budapest, Hungary, August 2018.
- [28] M. Zhang, X. Wang, T. Liu, J. Zhu, and Q. Wu, “AFSndn: a novel adaptive forwarding strategy in named data networking based on Q-learning,” *Peer-to-Peer Networking and Applications*, vol. 13, no. 4, pp. 1176–1184, 2020.
- [29] J. Mo and J. Walrand, “Fair end-to-end window-based congestion control,” *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, 2000.
- [30] K. Winstein and H. Balakrishnan, “TCP ex machina: computer-generated congestion control,” in *Proceedings of the ACM SIGCOMM 2013 Conference (SIGCOMM)*, pp. 123–134, Hong Kong, China, August 2013.
- [31] S. Ramakrishnan and V. Ramaiyan, “Completely uncoupled algorithms for network utility maximization,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 607–620, 2019.
- [32] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Advances in Neural Information Processing Systems*, vol. 12, no. NIPS, pp. 1057–1063, 1999.
- [33] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, “Deterministic policy gradient algorithms,” in *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pp. 387–395, Beijing, China, June 2014.
- [34] V. Mnih, K. Kavukcuoglu, D. Silver et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [35] V. R. Konda and J. N. Tsitsiklis, “Actor-Critic algorithms,” in *Advances in Neural Information Processing Systems*, pp. 1008–1014, MIT Press, Cambridge, MA, USA, 1999.
- [36] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, PR, USA, May 2016.
- [37] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2094–2100, New York, NY, USA, February 2016.
- [38] P. Erdős and A. Rényi, “On random graphs I,” *Publicationes Mathematicae*, vol. 4, pp. 3286–3291, 1959.
- [39] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, no. 5439, pp. 509–512, 1999.
- [40] G. Carofoglio, M. Gallo, and L. Muscariello, “Optimal multipath congestion control and request forwarding in information-centric networks: protocol design and experimentation,” *Computer Networks*, vol. 110, pp. 104–117, 2016.