Blockchain-Based Distributed Network Security Architecture with Smart Contract Vulnerability Detection Using Improved Tree CNN

Xiaoyan Huo
Information Construction and Management Center, Jiaozuo University, Jiaozuo 454003, China
E-mail: huoxiaoyan2004@163.com

Keywords: blockchain, distributed network architecture, privacy protection, smart contract, data security

Received: January 15, 2025

Abstract: In the era of big data, information security and privacy protection have become important issues facing today's society. This study proposes a distributed network security architecture based on blockchain to enhance the security of information privacy protection. The proposed architecture consists of three primary levels: equipment layer, network service layer, and application layer. It also integrates smart contracts. In addition, this study also proposes a vulnerability detection method based on improved tree convolutional neural networks. The incorporation of a "continuous binary tree" approach effectively addresses the limitation inherent to conventional tree convolution, wherein the number of nodes is fixed. This refinement enables a more effective capture of the hierarchical structure and semantic nuances inherent to smart contract code. The experiment used multiple datasets, each containing multiple IoT attack types and smart contract vulnerability code snippets. These datasets were evaluated based on a set of criteria, including but not limited to accuracy, recall, F1 scores, gas costs, and execution delays. Experiments have shown that the proposed method performs well in accuracy, precision, recall, and F1 scores compared to existing state-of-the-art methods, with an accuracy range of 89.62% to 98.36%, significantly better than Oyente (about 75%) and Securify (about 85%). Specifically, the proposed method achieved 96.14% accuracy in detecting reentrant attacks, compared to 78% for Oyente and 82% for Securify. The findings indicate that the architectural design exerts a substantial influence on enhancing network security performance, thereby ensuring the stability of the system by effectively mitigating the variability in response time.

Povzetek: Predlagana je distribuirana varnostna arhitektura omrežij na osnovi tehnologije veriženja blokov z izboljšano metodo drevesnih CNN za zaznavanje ranljivosti pametnih pogodb.

1 Introduction

With the rapid development of information technology, the internet has become an indispensable infrastructure in modern society, widely used in various fields such as finance, healthcare, education, and government affairs. However, cybersecurity issues are becoming increasingly prominent, with frequent incidents such as hacker attacks, data breaches, and online fraud, posing serious threats to personal privacy, corporate interests, and even national security [1, 2]. The traditional network security architecture mainly relies on centralized protection mechanisms, such as firewalls, intrusion detection systems, etc. These systems gradually show many shortcomings when facing complex network environments and increasing data volumes. Blockchain technology provides a new solution for network security with its decentralized, tamper-proof, and traceable features [3]. In a Distributed Network Security Architecture (DNSA), blockchain can serve as the underlying infrastructure for data storage and transmission. The data in the network are dispersed and stored on multiple nodes, with each node holding a complete copy of the data, thereby eliminating the risk of a single point of failure. At the same time, the immutability of blockchain ensures the authenticity and integrity of data, and any unauthorized tampering behavior will be quickly detected and prevented, effectively preventing the risk of malicious modification of data [4, 5]. However, the current processing speed and storage capacity of blockchain cannot fully meet the needs of large-scale network applications. Secondly, when facing complex network attack methods, blockchain may still have certain security risks. Therefore, this study proposes a DNSA based on blockchain.

Blockchain technology is widely used in various fields such as finance, supply chain, healthcare, etc. due to its decentralized, tamper-proof, and transparent characteristics. To detect and mitigate malicious attacks in software-defined networks, Sharmila et al. used unsupervised and supervised learning methods to perform mitigation operations in software defined networks using dynamic access control lists and implemented them through Mininet. This technology effectively reduced malicious attacks [6]. To enhance the security and privacy protection of blockchain storage systems, Haque et al. developed a security system that integrates the Ethereum blockchain, IPFS, and deep Convolutional Neural Networks (CNN) to achieve distributed storage and privacy protection of data. The system had high accuracy, sensitivity, and specificity, supporting efficient operation

for users [7]. To enhance the security of IoT devices, Rangappa et al. developed a new key generation stage by introducing lightweight blockchain technology and Blowfish symmetric encryption algorithm and utilized the immutability of blockchain to record transactions. This scheme outperformed traditional algorithms in terms of encryption time and memory overhead [8]. Kumar designed an intrusion detection system based on a variational autoencoder and attention-gated recurrent unit for the security of zero contact networks. It achieved secure data sharing through an authentication protocol that combines blockchain, smart contracts, elliptic curve cryptography, and authoritative proofs [9]. Hua et al. designed a secure cloud storage service data deduplication scheme to address the shortcomings of existing solutions in key leakage and dynamic change support. By grouping key servers and using threshold encryption, combined with blockchain technology, secure key updates and management have been achieved [10].

DNSA is the key to ensuring the secure operation of distributed systems in complex network environments. To enhance secure communication in wireless networks with multi-user pairs and un-trusted amplification and forwarding relays, Xie et al. proposed an adaptive optimal channel access strategy by formulating channel access problems, maximizing the throughput of the secure system, and utilizing optimal sequential planning decision theory. The numerical results have verified its effectiveness and high efficiency [11]. Liu et al. proposed a distributed multi-task security estimation algorithm based on local outlier factors and inter-task correlations to address multiple attacks in multi-task networks. A new distributed time-varying fusion strategy has been introduced, which allocates node weights through data density balance to resist attacks. This method could effectively resist various attacks [12]. To address the threat posed by malicious nodes, Luo et al. proposed a new paradigm inspired by distributed systems that ensures the network's identity is unforgeable, non-repudiation, and globally consistent. This scheme significantly reduced key consumption compared to traditional pre-shared key schemes [13]. To improve the real-time and security of Internet of Vehicles communication, Thangam et al. subdivided the roadside unit area and deployed edge computing resources to reduce communication delay, and introduced a consensus mechanism to ensure security. This method had a success rate of over 95%, significantly reduced consensus time, and effectively met the security requirements of vehicle networking communication [14]. To enhance the ability of IoT networks to combat Distributed Denial of Service (DDoS) attacks, Mahdi et al. constructed a detection model by combining k-nearest neighbors, logistic regression, and stochastic gradient descent classifiers, and integrated optimized parameters through machine learning. The model achieved accuracies of 99.965% and 99.968% on two datasets, surpassing existing methods [15]. A summary of the comparison between this method and existing literature is shown in Table 1.

Table 1: Comparison of the proposed method with existing literature

Research purpose	Method	Result	Shortcomings	Reference
Secure data transmission in IoT using remote sensing data	Theil-Sen Regressive Miyaguchi–Preneel-based Cryptographic Hash Blockchain	Enhanced secure data transmission	Complexity in implementation	Sharmila et al. [6]
Privacy-preserving deep learning for blockchain secure storage	Privacy-preserving deep learning framework	High authentication and secure storage	High computational cost	Haque et al. [7]
Secure data communication in IoT healthcare systems	Lightweight Blockchain	Improved data security in healthcare IoT	Limited scalability	Rangappa et al. [8]
Secure zero touch networks using blockchain	Deep-learning-based blockchain	Enhanced network security	Dependence on deep learning accuracy	Kumar et al. [9]
Secure deduplication for large-scale cloud storage	Blockchain-assisted secure deduplication	Efficient and secure cloud storage	Potential latency issues	Hua et al. [10]
Secure channel access in distributed cooperative networks	Optimal secure channel access	Improved security in untrusted relay networks	Complexity in channel management	Xie et al. [11]
Secure distributed estimation over multitask networks	Secure distributed estimation	Robust against multiple attacks	High computational overhead	Liu et al. [12]
Secure quantum key distribution networks	Distributed information-theoretical secure protocols	Enhanced security against malicious nodes	Requires quantum infrastructure	Luo et al. [13]
Secure V2X communication	Edge-enabled DAG-based Distributed Ledger System	Improved V2X communication security	Limited to edge-enabled regions	Thangam et al. [14]
Detection of DDoS attacks on IoT networks	Machine learning algorithms	Real-time DDoS attack detection	Dependence on training data quality	Mahdi et al. [15]
/	Practical Byzantine Fault Tolerance (Baseline approach)	/	Communication overhead and computational complexity increase	/
Improve security and processing speed of distributed network architectures	Smart Contract Vulnerability Detection Using Improved Tree CNN	/	/	This study

In summary, the application of blockchain technology and DNSA has significant advantages. However, current blockchain technology has limitations in processing speed, especially when dealing with a large number of concurrent transactions, which may result in delays. With the continuous growth of blockchain data, nodes need to store a large amount of data, which puts high demands on the storage capacity of nodes. In response to the above issues, this study innovatively designs a modular DNSA. This architecture uses edge computing to improve system security and flexibility. In response to Denial of Service (DoS) attacks, this study proposes a blockchain-based protection mechanism that identifies attacks through data flow detection and filtering models and utilizes blockchain and smart contracts to ensure the security and consistency of policy management.

2 Methods and materials

To improve data privacy protection and network security, and effectively detect and defend against network attacks, especially vulnerabilities in smart contracts, the research proposes a DNSA design based on blockchain, which integrates smart contracts into the architecture. The improved Tree-based Convolutional Neural Network (TCNN) is used to improve the accuracy and efficiency of vulnerability detection, and a protection mechanism based on blockchain technology is proposed.

2.1 DNSA design based on blockchain

Traditional service systems are typically based on centralized architectures, simplifying deployment processes, reducing operational complexity, and ensuring data consistency. However, it poses security risks when resisting specific network attacks, such as DoS attacks in LoRa networks. The utilization of LoRa terminals is susceptible to security breaches due to their cost-effectiveness and lightweight design, which are inadequate in terms of security protection capabilities. These terminals are prone to intrusion and control by attackers, resulting in the generation of substantial attack data. This, in turn, leads to a single point of failure, server resource depletion, and service denial in centralized systems. Reference [16] proposes a distributed IoT network architecture based on blockchain, which ensures data security and immutability through blockchain technology and improves communication efficiency. Reference [17] demonstrates the resource management capabilities of blockchain in distributed systems. It proves that a distributed service architecture has significant advantages in providing flexibility, security, reliability, and autonomy. To ensure the stability and security of the system in the face of attacks, this study proposes a blockchain-based DNSA, as shown in Figure 1.

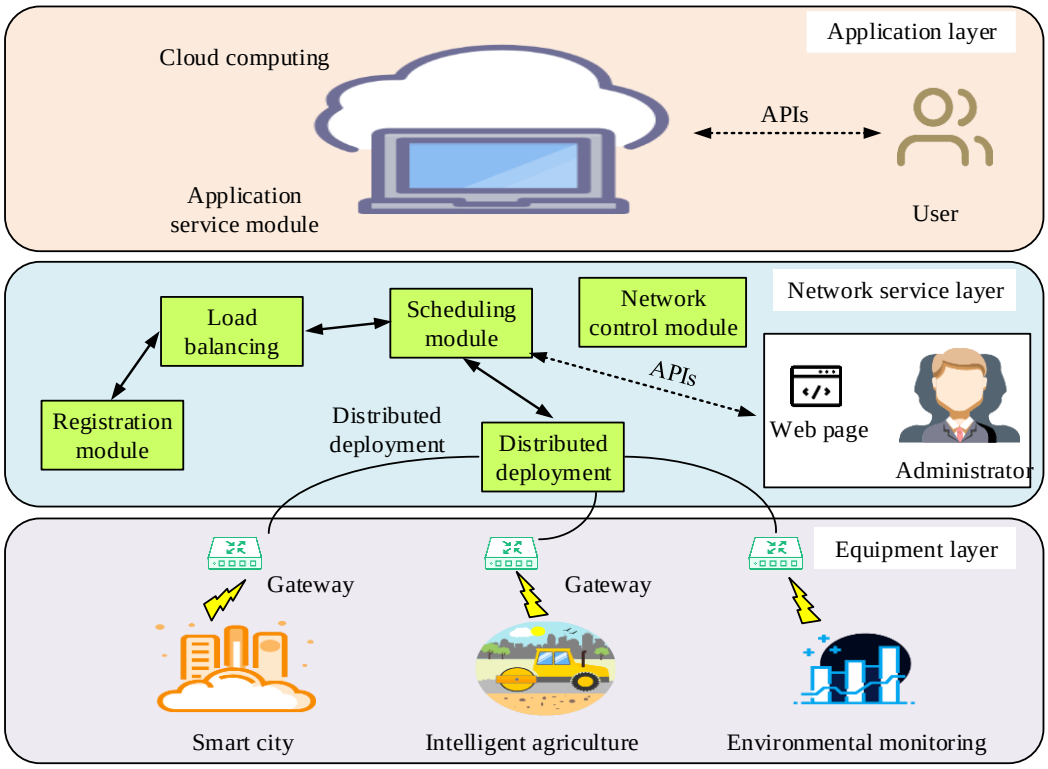


Figure 1: Schematic diagram of DNSA based on blockchain

This architecture is typically divided into three main levels and integrates various network components including terminal devices, gateways, and cloud servers, covering the entire process from data collection to processing, storage, and management. The Equipment layer is the foundation of the entire system and is responsible for direct interaction with physical devices and sensors. It not only ensures the accuracy and security

of the data but also provides a solid foundation for the upper-layer network services. Through the collaboration of the device layer and the network service layer, intelligent application scenarios can be comprehensively monitored and managed, thereby improving the efficiency and effect of the entire system. The network service layer focuses on providing core services such as access management, protocol analysis, and information transmission processing for terminal devices. The data collected at the device layer are transmitted to the network service layer through the gateway to provide basic information for decision-making and control at the upper layer. The network service layer can send commands to the device layer through the gateway to control the operation of the device and realize remote management and control. As for the application layer, it mainly undertakes the responsibilities of application data and user management, while providing interface services and user interfaces, as well as conducting in-depth analysis and mining of data. A layered system architecture design can ensure efficient operation and scalability of the network while providing flexible and reliable network services for various application scenarios. In response to security challenges in data protection and management processes, this study integrates smart contracts into a distributed service model, as shown in Figure 2.

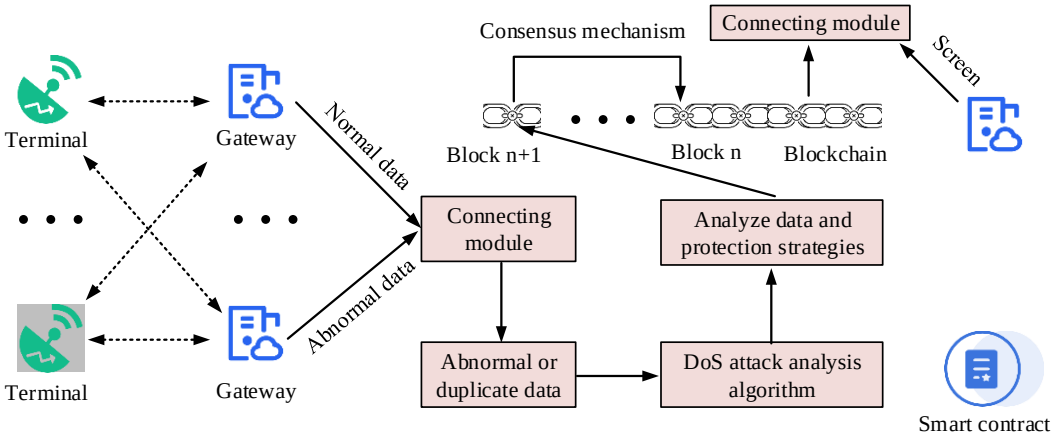


Figure 2: Schematic diagram of smart contract implementation in DNSA

Essentially, smart contracts are automated execution programs running on the blockchain, ensuring that all parties to the contract automatically comply with the agreement. Blockchain technology offers a decentralized, secure, and immutable execution platform for smart contracts, enabling any node in the network to participate in execution without the need for third-party intervention. This reduces the likelihood of default or fraudulent behavior. However, attackers can consume network resources through a large number of invalid transactions, causing contracts to malfunction. To solve this problem, a smart contract vulnerability analysis technique based on improved TCNN is proposed.

The code of smart contracts usually has a hierarchical structure, which can be represented by an Abstract Syntax Tree (AST). TCNN can directly process the tree structure data and capture the hierarchical and semantic information of the code. While Graph Neural Networks (GNNs) and Recurrent Neural Networks (RNNs) are both capable of processing graph-structured data. The code structure of smart contracts is better represented by a tree structure. TCNNs demonstrate superior proficiency in capturing semantic information within this hierarchical structure. In contrast, RNNs and GNNs have the potential to disregard these hierarchical characteristics, thereby introducing unnecessary complexity when handling tree structures.

TCNN mainly uses AST as input to capture the hierarchical and semantic information of smart contract code through tree structure, so as to improve the accuracy and efficiency of vulnerability detection. In a smart contract, the set of all subtrees is defined as $X \in R^{F\times N\times K}$, where F represents the number of subtrees. H is the maximum number of nodes in the subtree. K is the dimension of the node vector. In a fixed depth convolutional kernel, if there are n nodes corresponding to word embeddings $[x_1, x_2, \ldots, x_n]$, the output of the convolutional layer can be expressed as equation (1).

$$\mathbf{y}_{\text{conv}} = \sigma\left(\sum_{i=1}^{n}\sum_{j=1}^{3}\mathbf{W}_{\text{conv},j,i}\cdot\mathbf{x}_i + \mathbf{b}_{\text{conv}}\right) \quad (1)$$

In equation (1), $\mathbf{W}_{\text{conv},j,i}$ is the weight matrix. j represents three different types of weights. $\mathbf{b}_{\text{conv}}$ is the bias term. The activation function σ is selected as a nonlinear tanh function in the study. However, in equation (1), a defect of tree convolution is that the number of nodes n selected into the sliding window each time is not fixed, which makes it a great difficulty to set how many $W_{\text{conv},i}$ variable matrices. To solve this problem, a method called "continuous binary tree" is proposed, which can treat each sub-tree of AST as a binary tree, regardless of its shape and size. Figure 3 shows some definitions for applying the continuous binary tree method.

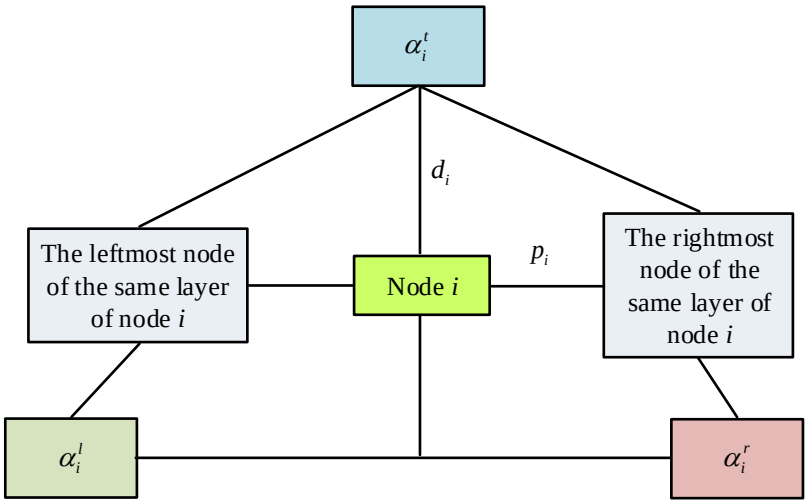


Figure 3: Schematic diagram of the continuous binary tree method

As mentioned above, for each sliding window, three variables $W_{\text{conv}}^t, W_{\text{conv}}^l, W_{\text{conv}}^r$ need to be set, and the corresponding coefficient $\alpha_i^t, \alpha_i^l, \alpha_i^r$ are respectively defined as up, left, and right. Therefore, for each node in the sliding window, its corresponding $W_{\text{conv},i}$ is a linear combination of these three variables, and the corresponding coefficient is determined by the relative position of the node in the sliding window sub-tree.

Even in complex situations where a node has multiple child nodes, the network can still perform convolution operations through three weight matrices. The calculation method of the weight matrix is shown in equation (2).

$$W_{\text{conv},i} = \alpha_i^t W_{\text{conv}}^t + \alpha_i^l W_{\text{conv}}^l + \alpha_i^r W_{\text{conv}}^r \tag{2}$$

In equation (2), W_{conv}^t is the coefficient of the weight matrix β_i^t, and its calculation is shown in equation (3).

$$\beta_i^t = \frac{d_{i-1}}{d_{\max-1}} \tag{3}$$

In equation (3), d_i is the depth of node i within the sliding window, and $d_{\max}$ is the maximum depth that the sliding window can cover. The coefficients of the other two weight matrices can be defined as equation (4).

$$\begin{cases} \alpha_i^l = (1-\alpha_i^t)\dfrac{n-p_i}{n-1} \\ \alpha_i^r = (1-\alpha_i^t)\dfrac{p_{i-1}}{n-1} \end{cases} \tag{4}$$

In equation (4), p_i is the sequential position of node i among its peers. Through this formula, it can understand how the specific position of nodes in the tree structure affects the relative importance of the weight matrix. Especially, nodes located on the left side of the tree will have a weight matrix W_{conv}^l that accounts for more than W_{conv}^r when performing convolution operations, while nodes located on the right side will have the opposite situation. After the convolution operation, the pooling layer extracts key features from the intermediate expressions of the network and summarizes these features in the form of high-dimensional vectors.

This study adopts the maximum pooling strategy, which selects the most prominent elements from the feature matrix generated by the convolutional layer. Compared to average pooling, which calculates the mean value of the pooling window, maximum pooling focuses on the most significant features, making it more suitable for capturing critical vulnerabilities in smart contract code. This is of particular importance in the realm of security applications, where the ability to detect rare but severe vulnerabilities, such as re-entrancy attacks, is of the essence. Additionally, maximum pooling reduces the dimensionality of the feature map while preserving the most relevant information, which enhances the efficiency of the subsequent layers. For a given pooling window of size $k \times k$, the maximum pooling operation can be defined.

$$\mathbf{V}_{\text{pooling}} = \max_{m,n\in[0,k-1]} x_{i\times s+m,\, j\times s+n} \tag{5}$$

In equation (5), x is the input feature map. $\mathbf{V}_{\text{pooling}}$ is the output feature map. s is the stride. This operation extracts the maximum value within each pooling window, effectively reducing the dimensionality of the feature map while preserving the most significant features.

After the pooling step is completed, the extracted feature vectors are passed to the hidden layer, which processes the features and feeds them into the classifier. The classifier then generates the final output for the prediction model. The hidden layer's role is to transform the pooled features into a higher-level representation that the classifier can use to make accurate predictions. Similar to convolutional layers, hidden layers also use tanh

activation functions. The output of the hidden layer can be expressed as equation (6).

$$\mathbf{y}_{\text{hide}} = \tanh\left(\mathbf{W}_{\text{hide}} \cdot \mathbf{V}_{\text{pooling}}^{\text{T}} + \mathbf{b}_{\text{hide}}\right) \tag{6}$$

Subsequently, the output of the hidden layer is fed into the classifier. This study uses a softmax classifier to predict the probability of various types of vulnerabilities occurring, expressed as equation (7).

$$\mathbf{y}_{\text{output}} = \text{softmax}(\mathbf{y}_{\text{hide}}) \tag{7}$$

Figure 4 shows the architecture of a vulnerability detection method based on an improved TCNN. Smart contract vulnerabilities are the premise for attackers to launch attacks, and attackers use the vulnerabilities in smart contracts to achieve their malicious purposes. The research identifies potential vulnerabilities in smart contracts, such as reentrant attacks, integer overflow, etc., through improved TCNN technology. Based on the type of vulnerability and its potential impact range, the risk level of each vulnerability was evaluated to help developers prioritize high-risk vulnerabilities. When an attack is detected, the defense mechanism can respond in real time, for example, by restricting access to malicious nodes or adjusting Gas charges to stop the attack. The vulnerability information identified by the vulnerability detection method can provide the basis for the defense mechanism and help the system better cope with potential attacks. The defense mechanism can prevent attackers from using identified vulnerabilities to launch attacks, thereby enhancing the actual effect of vulnerability detection.

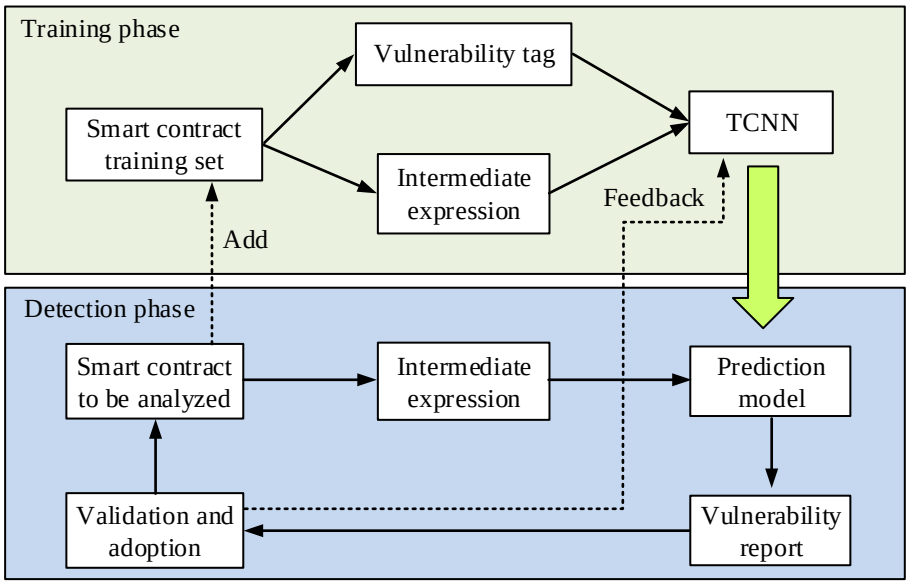


Figure 4: Vulnerability detection method architecture based on improved TCNN

2.2 Protection mechanism based on blockchain technology

DNSA faces complex system security issues in its application process, therefore it is necessary to establish effective security barriers. This study classifies and analyzes existing attack methods, proposes corresponding defense strategies, and constructs a relatively complete blockchain security defense system to combat DoS and DDoS attacks. The operation process is detailed in Figure 5. DoS attacks and DDoS attacks can launch a large number of invalid requests, consume network resources, and make services unavailable. An attacker can launch a coordinated attack by taking control of multiple nodes or devices. The attack may be continuous (continuous DoS attack) or intermittent (pulsed DoS attack), affecting the response time and stability of the system. When the detected abnormal traffic exceeds the preset security threshold, the connection module will activate the smart contract in the blockchain platform to record and respond to attack behavior. Smart contracts will determine corresponding punishment measures and restriction periods based on the frequency and duration of attack behavior. Subsequently, the processing results of the smart contract will be encapsulated into a new block and recorded on the blockchain through the network consensus mechanism.

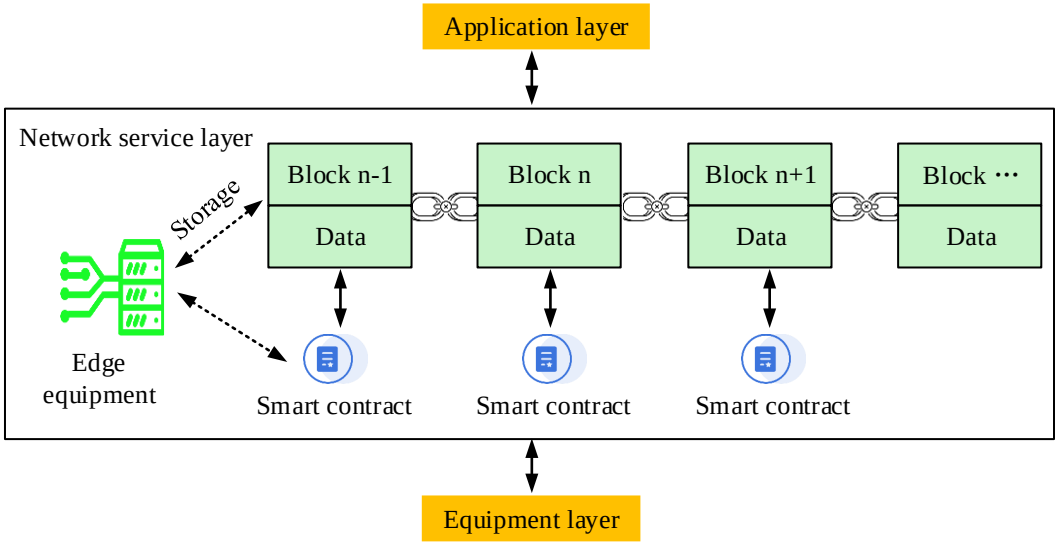


Figure 5: Blockchain-based DoS attack protection mechanism

According to Figure 5, to get rid of the disadvantages of a centralized management model and make full use of edge device resources, smart contracts are usually executed and verified by edge devices. However, limitations such as wide-area deployment, complex environment, and hardware performance of edge devices make them easier targets for attackers. Compared with other cyber attacks, the anonymity of blockchain technology also gives the attacker a natural camouflage, making the attack behavior with low risk, high profit, and easy-to-realize characteristics [18, 19]. Therefore, there is an urgent need for appropriate and efficient detection methods to conduct comprehensive security analysis for IoT smart contracts before deployment.

Edge computing is a paradigm that facilitates the execution of data processing and analytics at the periphery of the network, in closer proximity to the data sources and users. This approach has the potential to markedly reduce the time required for data transfer to remote data centers, thereby decreasing latency. With the increase of IoT devices, the amount of data has risen dramatically. By processing data locally and sending only critical data to the cloud, edge computing effectively reduces the pressure on network bandwidth and reduces data transmission costs. To reduce the response time of the system and enhance the efficiency of data transmission, this study also introduces edge computing and integrates multiple service modules to form the edge layer, as shown in Figure 6.

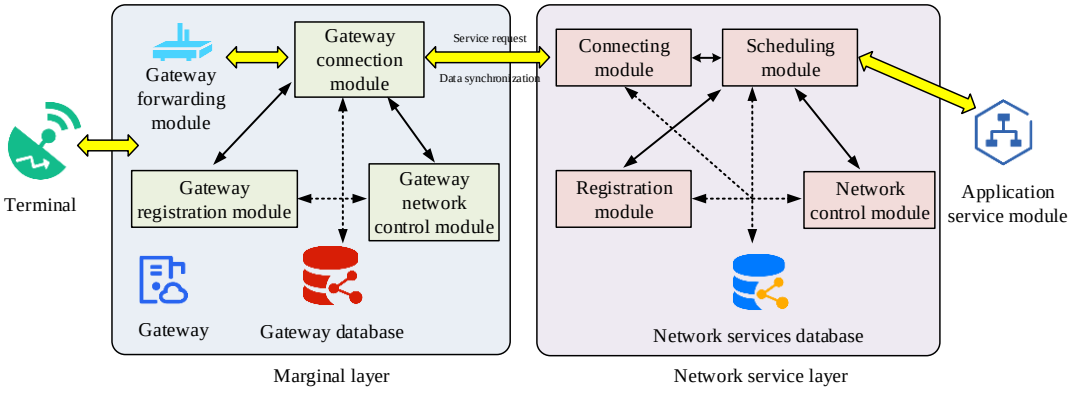


Figure 6: Network service architecture based on edge computing

A comparison with Figure 6 reveals that the original gateway, which is responsible for data forwarding, deployed a variety of service modules and is divided into edge layers. The diagram shows the gateway deploying all optional functional modules, including the connection module, registration module, and network control module, as well as the lightweight database. It should be noted that the gateway forwarding module is the basic function of the original gateway and is mainly responsible for receiving and packaging the physical layer data of the terminal. The network service layer retains all modules in the original system and provides complete service functions to ensure that all types of gateways can be accessed.

Considering that DoS attacks may occur in various forms, for example, some terminals may be frequently attacked in a short period, while other terminals may be intermittently attacked for a longer period [20, 21]. Therefore, the analysis method must simultaneously consider the frequency and time interval of attacks in order to update corresponding defense strategies. This study proposes a calculation method for the penalty parameter $\Delta\alpha$ based on the time interval of abnormal behavior

occurrence. When calculating the penalty parameter $\Delta\alpha$, the system takes into account the time interval between the latest attack t and the last attack T. The penalty parameter exhibits a proportional relationship with the increase in the number of attacks, thereby ensuring that the terminal with the shorter attack interval is subjected to a more severe penalty. The formula is defined as equation (8).

$$\Delta\alpha = \alpha_{prev} + a \cdot (t - T) \tag{8}$$

In equation (8), α_{prev} is the previous penalty parameter, a is a positive weight factor that determines the sensitivity of the penalty adjustment, and $t-T$ is the time interval between attacks. The weight factor a is set to 0.1 based on empirical testing, ensuring that the penalty increases gradually but significantly for frequent attackers. This approach aligns with the system's goal of dynamically adjusting restrictions based on attack behavior. Using penalty parameters, smart contracts are responsible for determining and adjusting the restriction period for terminals exhibiting aggressive behavior. This strategy is achieved by setting a limit period d for the terminal, as shown in equation (9).

$$d = d^* + \beta \cdot p + \gamma \tag{9}$$

In equation (9), d^* is the previously set limit period. β is a positive adjustment factor used to adjust the restriction period based on the penalty parameter p. γ is a random variable that is uniformly distributed within a predetermined range. The introduction of randomness is to alleviate the instantaneous peak pressure that DoS attacks may cause and to impose varying degrees of restrictions on malicious terminals launching attacks simultaneously [22]. As terminal abnormal behavior accumulates, the restriction period will gradually be extended. Finally, based on the restriction period t_e, the specific time point for implementing the restriction measures is determined, as shown in equation (10).

$$t_e = \begin{cases} t + d, & if: d \le \eta \\ t_\infty, & if: d > \eta \end{cases} \tag{10}$$

In equation (14), t_∞ is the maximum possible value of the time stamp. η serves as a critical value for evaluating the behavior of the terminal. The system employs a two-tiered approach to handle DoS attacks: temporary restrictions for initial or intermittent attacks, and permanent restrictions for persistent offenders. When a terminal exhibits repeated attack behavior, the system dynamically adjusts the restriction period based on the frequency and severity of the attacks. If the terminal continues to attack after multiple temporary restrictions and its accumulated restriction time exceeds a critical threshold η, the system will impose permanent measures to prohibit further access. This ensures a balanced approach between mitigating immediate threats and preventing long-term abuse.

3 Results

This study first conducts performance testing on the newly proposed DNSA and conducts in-depth comparisons of the performance of various methods to reveal their differences.

3.1 DNSA performance testing based on blockchain

The testing environment for this study is Intel(R) Core(TM) i5-2430M, CPU@2.40GHz, and 64 bit Windows 7 system. The IoTID20 dataset used in this study is a dataset specially created for the research of intrusion detection systems in the IoT environment, which contains a variety of IoT attack types, such as DDoS, DoS, Mirai, and ARP spoofing, etc. These datasets can provide rich data for the training and evaluation of intrusion detection models. Both the datasets CWE-119-SET and CWE-399-SET are based on the U.S. National Vulnerability Database and the National Institute of Standards and Technology's Software Assurance Reference dataset. The dataset CWE-119-SET contains 10,440 code gadgets related to buffer errors, which are commonly used for training and testing deep learning models. The dataset CWE-399-SET covers Resource Management Errors, such as resource leakage and post-release use. It also contains 7,285 code snippets related to resource management errors: used to train and test vulnerability detection models to identify resource management related vulnerabilities. To ensure that the model has enough data for training and the reliability of verification and testing process, the dataset is divided into training set, test set, and verification set according to the ratio of 6:2:2. Batch size is set to 32 to improve training efficiency and stability. The number of training rounds is set to 100 to ensure full convergence of the model. Table 2 shows the definition of each hyperparameter and the optional values.

Table 2: Definitions and optional values of hyperparameters

Hyper parameter	Definition	Optional value
conv_filter	The number of filters in the convolution layer	[32, 256] (step=32)
conv_kernel	Convolution window size	[3, 7]
pool_size	Maximum pool window size	[2, 4]
lstm_units	Number of units in the LSTM layer	[32, 512] (step=32)

dropout_rate	Random dropout rate of dropout layer 1	[0.2,0.5]
dense_units	The number of units in the fully connected layer	[32, 512] (step=32)
dense_dropout_rate	Random dropout rate of dropout layer 2	[0.2, 0.5]
optimizer	Adaptive optimizer	Adam,RMSProp,SGD
learning_rate	Training learning rate	[0.01, 0.005, 0.001, 0.0005, 0.0001]

```
# Training
batch_size = 32
epochs = 100

early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_
weights=True)

history = model.fit(
    X_train, y_train,
    batch_size=batch_size,
    epochs=epochs,
    validation_data=(X_val, y_val),
    callbacks=[early_stopping]
)
```

Figure 7: Exact code for model training

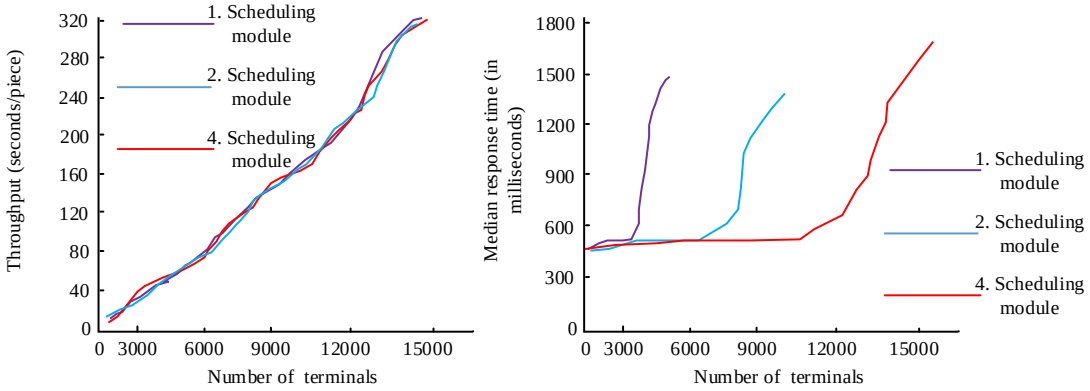
The exact code for model training is shown in Figure 7. To prevent overfitting, the optimized experiment collocation is adjusted using the early stop method and the learning rate scheduler. In the experiment, the main evaluation index of the early stop method is loss. The early stop strategy is that when the performance of the model on the verification set does not improve for ten consecutive epochs, it is considered that the model begins to over-fit and stop training. Table 3 shows all hyperparameter settings for the 10 Bayesian optimization experiments, with batch_size defaulting to 32. Finally, it is determined that the hyperparameters of the eighth experiment are the optimal settings of TCNN.

Table 3: Hyperparameter settings of 10 Bayesian optimization tests

Trial_id	0	1	2	3	4	5	6	7	8	9
conv_filter	192	32	224	96	256	224	160	32	256	256
conv_kernel	7	6	4	6	4	5	7	7	7	7
pool_size	2	4	3	4	2	2	3	2	2	2
Ismm_units	480	128	320	384	288	384	448	512	512	512
dropout_rate	0.41153	0.4175	0.32132	0.47735	0.40378	0.40965	0.46895	0.5	0.5	0.5
dense_units	96	352	128	256	32	192	192	32	32	416
dense_dropout_rate	0.30759	0.49836	0.29813	0.31899	0.41386	0.41325	0.32615	0.21152	0.42292	0.31177
optimizer	RMSProp	SGD	SGD	Adam	RMSProp	RMSProp	Adam	Adam	Adam	Adam
leamning_rate	0.005	0.01	0.005	0.005	0.0001	0.0001	0.001	0.005	0.001	0.01

Figure 8 shows the trend of throughput changes in the network service layer. The median response time refers to the value in the middle of a set of response time data arranged in order from smallest to largest. In a computer system or network system, the median response time can be used to evaluate the performance of the system. For example, for a website's response time data, the median can reflect the response time when most users visit the website. In Figure 8 (a), when the number of LoRa devices connected to the network is small, the network service layer can effectively manage all received data streams, resulting in a direct proportional relationship between throughput and the number of devices. When the number of terminals is small (for example, less than 2000), the throughput difference between the three scheduling modules is small. With the increase of the number of terminals, the impact of the number of scheduling modules on throughput gradually appears, and the advantage of using four scheduling modules is more obvious. In Figure 8 (b), even with a small number of terminals, the system's response time exceeds 200 milliseconds. This phenomenon is attributed to the DoS attack defense

mechanism implemented in the network service system. However, as long as the number of terminals does not reach the maximum carrying capacity of the system, the median response time will remain stable. However, once the server faces the limit of its processing capacity, the median response time will significantly increase, mainly due to the server hardware performance reaching its limit, resulting in a significant extension of data processing time.

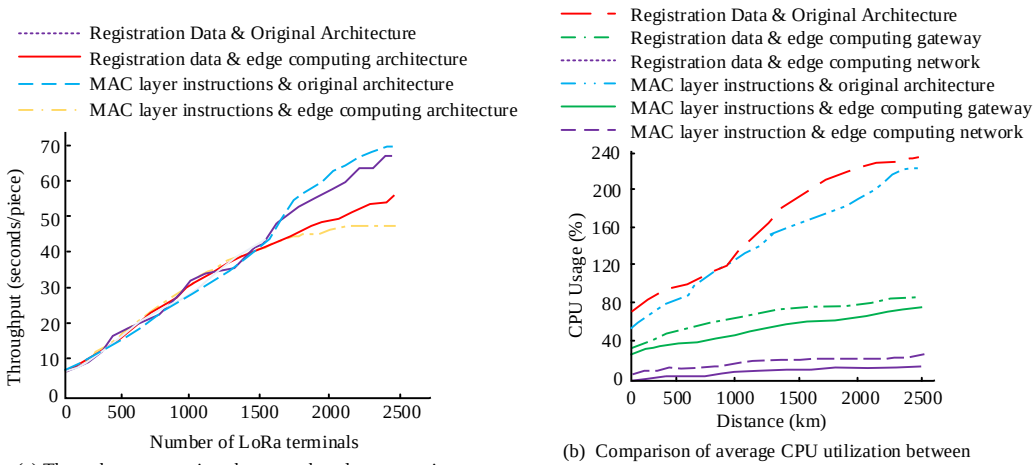


(a) Throughput changes of parallel functional modules in distributed deployment

(b) Median response time for distributed deployment of parallel functional modules

Figure 8: Throughput variation trend of network service layer

To measure the performance of the gateway in the edge computing environment, the experiment compares two types of uplink data: registration information and MAC command messages. According to Figure 9, in the case of a small number of terminal devices, the data processing throughput of both architectures shows a linear increase trend with an increase in the number of devices. However, when the number of terminal devices reaches 1600, the data processing capacity of the gateway reaches its limit, and its maximum throughput remains stable at about 53 data per second. In addition, the performance of the gateway in handling these two types of data is basically the same. Due to the original architecture deploying functional modules directly on the server, it provides strong computing power, ensuring further improvement in throughput. Although the processing power of a single gateway is limited, it can indeed undertake some of the computing work of the backend network system.



(a) Throughput comparison between the edge computing architecture and the original architecture

(b) Comparison of average CPU utilization between the edge computing architecture and the original architecture

Figure 9: Feasibility analysis of network security architecture based on edge computing

Figure 10 depicts the fluctuation of the median response time of conventional data under continuous and pulse DoS attack scenarios. In Figure 10 (a), under sustained DoS attacks, the fixed duration restriction strategy significantly reduces response latency after two applications and showed slight fluctuations in the subsequent time period. The proposed protection strategy can more quickly control the fluctuation of response time and reduce the amplitude of the fluctuation. This is thanks to the introduction of randomized parameters and penalty mechanisms in the strategy, which can dynamically adjust the duration of restrictions and effectively alleviate the continuous impact of attack traffic. In Figure 10 (b), the fixed duration restriction strategy cannot adjust the restriction duration on malicious nodes, resulting in a significant increase in response time for each attack, which may temporarily render the service unavailable. In contrast, the research strategy gradually increases the

punishment after identifying the attack behavior multiple times, reducing the impact of the attack on the system's processing capability and ensuring the stability of response time.

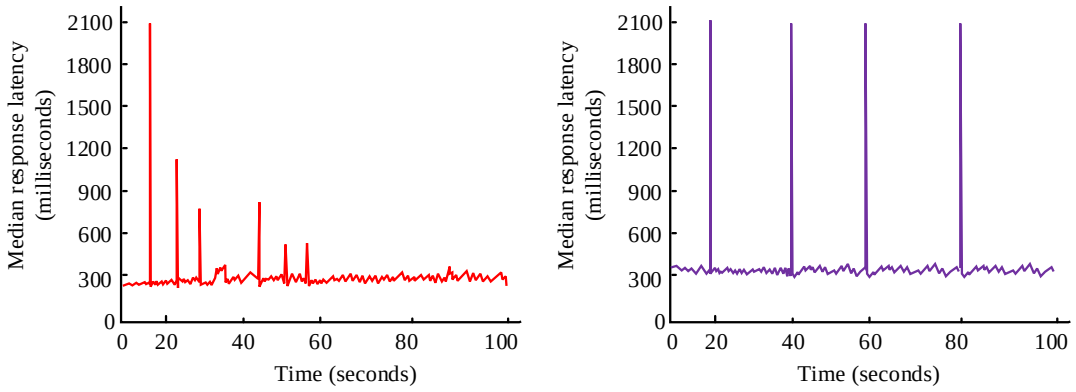


Figure 10: Data response time under different DoS attack scenarios

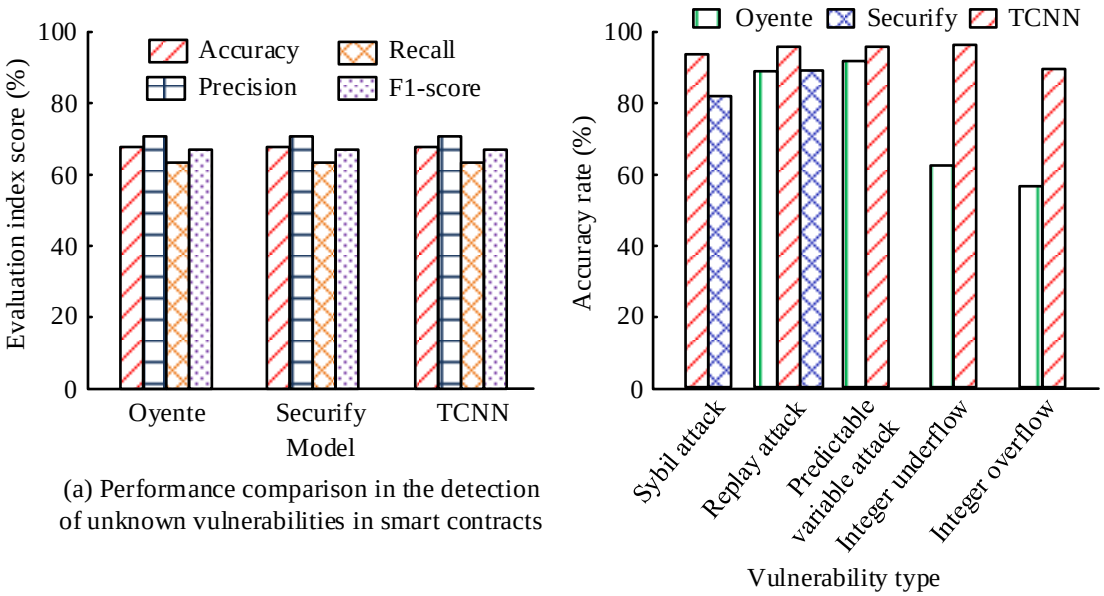
3.2 Performance analysis of TCNN vulnerability detection

To verify the effectiveness of the TCNN vulnerability detection method, this study analyzes accuracy, precision, recall, and F1 score as indicators, as shown in Table 4. When TCNN identifies vulnerabilities of five different categories, the accuracy of the test dataset ranges from 89.62% to 98.36%. This proves that the predictive model performs well in identifying potential vulnerabilities in the training samples. In addition, the accuracy and recall of this technology also meet high-performance requirements, effectively reducing the occurrence of misidentification and missed problems.

Table 4: Performance results of TCNN detecting different types of vulnerabilities

Vulnerability type	Data set	Accuracy (%)	Precision (%)	Recall rate (%)	F1 score (%)
Re-entrant attack	Training set	98.36	97.96	98.28	98.12
	Test set	96.14	96.38	95.23	95.80
Integer overflow	Training set	93.79	96.31	86.43	91.10
	Test set	89.62	90.22	80.23	84.93
Integer underflow	Training set	96.64	89.53	72.43	80.08
	Test set	96.66	90.84	71.64	80.11
Predictable variable attack	Training set	97.06	99.56	93.09	96.22
	Test set	95.94	98.66	91.16	95.55
Abnormal state	Training set	93.84	86.28	77.04	81.40
	Test set	93.14	83.33	74.38	78.60

Figure 11 (a) shows the best performance results of Oyente, Securify, and TCNN models in the unknown vulnerability detection of smart contracts. The TCNN model has the best effect, achieving the highest value in the four evaluation indicators, which verifies the significant advantages of TCNN in the unknown vulnerability detection of smart contracts. In Figure 11 (b), TCNN has the highest accuracy in all attack types, and the detection accuracy of Sybil attack and Replay attack both exceeds 90%. Securify performs second on most attack types but outperforms Oyente on Integer underflow and Integer overflow. Oyente has a relatively low accuracy rate across all attack types, particularly on Integer underflow and Integer overflow.



(a) Performance comparison in the detection of unknown vulnerabilities in smart contracts

(b) Comparison of the accuracy of detecting different types of leakage holes

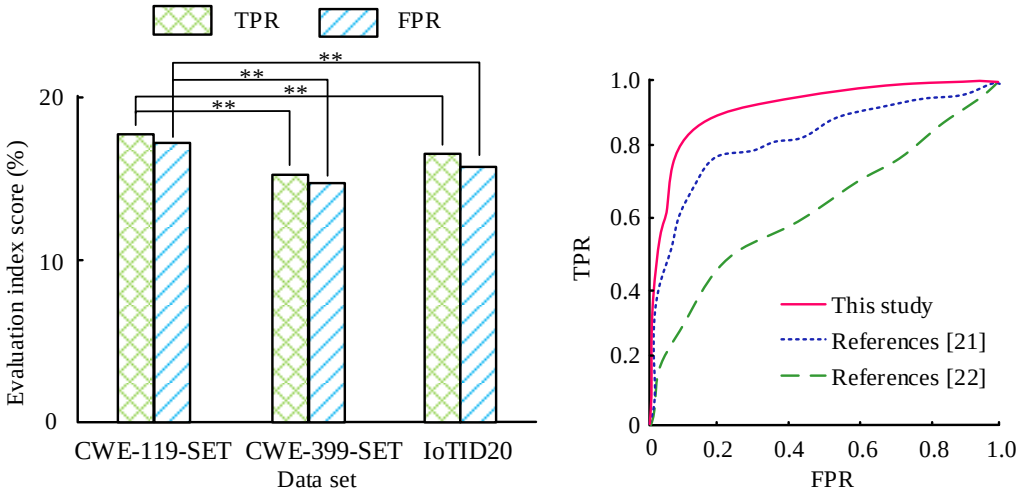
Figure 11: Performance assessment of multiple types of vulnerabilities

To evaluate the computational cost of deploying smart contracts, the Gas fees and execution delays for different settings (Table 5) are measured. TCNN achieves the lowest Gas fees (38,000 Gwei) and execution delay (90 ms), significantly outperforming Oyente and Securify. This result demonstrates the efficiency of the proposed architecture in real-time applications.

Table 5: Gas fees and execution delay analysis

Setting	Gas Fee (Average, in Gwei)	Execution Delay (Average, in ms)
Reference [21]	50,000	120
Reference [22]	45,000	110
Oyente	55,000	150
Securify	52,000	140
TCNN (Proposed)	38,000	90

Figure 12 shows the results of the comparison between different scenarios and different approaches. In Figure 12 (a), the False Negative Rate (FNR) and False Positive Rate (FPR) are not significantly different in all data sets, indicating that the model has more balanced types of misjudgment on these data sets. To verify the statistical significance of the performance differences between the Settings, a paired T-test is performed for the FNR and FPR values and it is found that both CWE-399-SET and IoTID20 are significantly better than CWE-119-SET ($P<0.05$). The effect deviation of the overall vulnerability detection is within the acceptable range, which also indicates that the method in this paper is generally applicable to different vulnerability types. Figure 12 (b) shows the Receiver Operating Characteristic curve (ROC) graph between the proposed model and references [21] and [22]. From the area represented in the figure, the ROC curve of the proposed method is closer to the upper left corner, indicating that the proposed method has better performance.



(a) Comparison of false positive rate and false negative rate in different data sets (Note: "**" means P< 0.05)

(b) ROC curves of different detection methods

Figure 12: Comparison results between different scenarios and different methods

4 Discussion

The proposed DNSA architecture has significant advantages over existing methods, mainly due to its deep integration of smart contracts and high efficiency of improved TCN. First, the integration of smart contracts enables the system to automate the execution of security policies and attack responses, reducing the need for human intervention and improving the real-time and reliability of the system. Although the scheme proposed by Yu et al. enhances the security of data transmission, its implementation is complicated and relies on manual intervention, which is difficult to cope with large-scale network attacks [23]. The privacy-protecting deep learning framework of Miglani et al., despite its excellent performance in storage security and authentication performance, relies on complex models, resulting in a substantial increase in computational overhead [24]. By automatically recording and responding to attack behaviors through smart contracts, this architecture significantly improves the automation level of the system. In particular, when dealing with DoS and DDoS attacks, it can dynamically adjust the restriction policy (as shown in Figure 5), effectively alleviating the impact of attacks on the system. Second, the improved TCNN excelled in vulnerability detection, with a significantly higher accuracy (89.62% to 98.36%) than existing methods such as Oyente (around 75%) and Securify (around 85%). TCNN solved the problem that the number of nodes in traditional tree convolution was not fixed by "continuous binary tree" method. TCNN could better capture the hierarchical structure and semantic information of smart contract code, thereby improving the accuracy and efficiency of vulnerability detection. TCNN also outperformed Oyente (55,000 Gwei/150 ms) and Securify (52,000 Gwei/140 ms) in terms of Gas costs (38,000 Gwei) and execution latency (90 ms), making it more suitable for real-time application scenarios.

In summary, this architecture significantly improves the security, real-time performance, and scalability of the system through the comprehensive application of automatic execution of smart contracts, efficient vulnerability detection of TCNN, and dynamic defense mechanisms. It is superior to existing methods in dealing with complex network attacks and large-scale data. However, the training of the TCNN model relies on large-scale labeled data sets, and its training process consumes a lot of computing resources. Although the overfitting problem is alleviated by Bayesian optimization and early stop strategy, the model training cost may still become the deployment bottleneck. In addition, although edge computing reduces network layer latency, the gateway's throughput ceiling may limit its scalability in high-concurrency scenarios. Future research could optimize blockchain storage efficiency by introducing sharding technology, and explore lightweight TCNN models to balance detection accuracy with computational overhead.

5 Conclusion

This study was based on the design of DNSA using blockchain technology, which addressed the shortcomings of traditional network security architectures in dealing with complex network environments and large-scale data volumes. By combining blockchain technology with DNSA, this study has achieved significant results in data security, privacy protection, and resistance to network attacks. Experiments have shown that this architecture can effectively reduce response latency and maintain high throughput when handling numerous concurrent transactions. For example, when the number of LoRa devices reached 1600, the maximum throughput of the gateway remained stable at about 53 data per second, demonstrating good data processing capabilities. In the case of sustained DoS attacks, research strategies could more quickly control the fluctuation of response time and reduce the amplitude of the fluctuation. This was thanks to the introduction of randomized parameters and penalty

mechanisms in the strategy, which can dynamically adjust the duration of restrictions and effectively alleviate the continuous impact of attack traffic. After identifying the attack behavior multiple times, the punishment was gradually increased, reducing the impact of the attack on the system's processing capability and ensuring the stability of response time. In addition, the smart contract vulnerability detection method based on the improved TCNN performed well in accuracy, precision, recall, and F1 score, with an accuracy ranging from 89.62% to 98.36%, significantly better than other existing methods. The results demonstrated the effectiveness of the proposed architecture in improving network security performance.

References

[1] Abdullah N F, Kairaldeen A R, Abu-Samah A, Nordin R. Machine learning-based transactions anomaly prediction for enhanced IoT Blockchain network security and performance. KSII Transactions on Internet and Information Systems, 2024, 18(7): 1986-2009. DOI:10.3837/tiis.2024.07.014.

[2] Li Y, Liang L, Jia Y, Wen W, Tang C, Chen Z. Blockchain for data sharing at the network edge: trade-off between capability and security. IEEE/ACM Transactions on Networking, 2024, 32(3): 2616-2630. DOI:10.1109/TNET.2024.3364023.

[3] Wang J, Ou W, Wang W, Sherratt R S, Ren Y, Yu X. Data security storage mechanism based on blockchain network. Computers, Materials & Continua, 2023, 74(3): 4933-4950. DOI:10.32604/cmc.2023.034148.

[4] El Ghazouani M, Ikidid A, Zaouiat C A, Aziz L, Lachgar M, Er-Rajy L. A Blockchain-based method ensuring integrity of shared data in a distributed-control intersection network. International Journal of Advanced Computer Science and Applications, 2023, 14(10): 489-497. DOI:10.14569/IJACSA.2023.0141052.

[5] Fkaier S, Khalgui M, Frey G, et al. Secure distributed power trading protocol for networked microgrids based on blockchain and elliptic curve cryptography. IET Smart Grid, 2023, 6(2): 175-189. DOI:10.1049/stg2.12087.

[6] Sharmila B, Kalavathi Devi T. Theil-Sen regressive Miyaguchi-Preneel-based cryptographic hash blockchain for secure data transmission using remote sensing data in IoT. IETE Journal of Research, 2024, 70(1): 116-129. DOI:10.2139/ssrn.4156878.

[7] Haque S M U, Sofi S A, Sholla S. A privacy-preserving deep learning framework for highly authenticated blockchain secure storage system. Multimedia Tools and Applications, 2024, 83(36): 84299-84329. DOI:10.1007/s11042-024-19150-7.

[8] Rangappa J D, Manu A P, Kariyappa S, Chinnababu S K, Lokesh G H, Flammini F. A lightweight Blockchain to secure data communication in IoT network on healthcare system. International Journal of Safety & Security Engineering, 2023, 13(6): 1015-1024. DOI:10.18280/ijsse.130604.

[9] Kumar R, Kumar P, Aloqaily M. Deep-learning-based blockchain for secure zero touch networks. IEEE Communications Magazine, 2022, 61(2): 96-102. DOI:10.1109/MCOM.001.2200294.

[10] Hua Z, Yao Y, Song M, Zheng Y, Zhang Y, Wang C. Blockchain-Assisted secure Deduplication for large-scale cloud storage service. IEEE Transactions on Services Computing, 2024, 17(3): 821-835. DOI:10.1109/TSC.2024.3350086.

[11] Xie J, Zhang Z, Atapattu S, Ye Y, Zhang H. Optimal secure channel access in distributed cooperative networks with untrusted relay. IEEE Wireless Communications Letters, 2023, 12(6): 1091-1095. DOI:10.1109/LWC.2023.3262287.

[12] Liu Q, Ye M, Chen F. Secure distributed estimation over multitasks networks against multiple attacks. IEEE Transactions on Aerospace and Electronic Systems, 2022, 59(3): 2480-2493. DOI:10.1109/TAES.2022.3215948.

[13] Luo Y, Li Q, Mao H K. Distributed information-theoretical secure protocols for quantum key distribution networks against malicious nodes. Journal of Optical Communications and Networking, 2024, 16(10): 956-968. DOI:10.1364/JOCN.530575.

[14] Thangam S, Chakkaravarthy S S. An edge enabled region-oriented DAG-based distributed ledger system for secure V2X communication. KSII Transactions on Internet and Information Systems, 2024, 18(8): 2253-2280. DOI:10.3837/tiis.2024.08.011.

[15] Mahdi Z, Abdalhussien N, Mahmood N, Zaki R. Detection of real-time distributed denial-of-service (DDoS) attacks on internet of things (IoT) Networks using machine learning algorithms. Computers, Materials & Continua, 2024, 80(8): 2139-2159. DOI:10.32604/cmc.2024.053542.

[16] Sharma R K, Pippal R S. Blockchain based efficient and secure peer-to-peer distributed IoT network for non-trusting device-to-device communication. Informatica, 2023, 47(4): 515-522. DOI:10.31449/inf. v47i4.3494.

[17] Lyu Z, Cheng C, Lv H, Song H. Blockchain based intelligent resource management in distributed digital twin's cloud. IEEE Network, 2023, 38(4): 143-150. DOI:10.1109/MNET.2023.3326099.

[18] Wu W, Yu L, Yang L, Zhang Y, Wang P. Efficient digital twin placement for blockchain-empowered wireless computing power network. Computers, Materials & Continua, 2024, 80(7): 587-603. DOI:10.32604/cmc.2024.052655.

[19] Singh U, Sharma S K, Shukla M, Jha P. Blockchain-based BATMAN protocol using mobile ad hoc network (MANET) with an ensemble algorithm. International Journal of Information Security, 2024, 23(3): 1667-1677. DOI:10.1007/s10207-023-00804-w.

[20] Filatovas E, Stripinis L, Orts F, Paulavičius R. Advancing research reproducibility in machine learning through blockchain technology. Informatica, 2024, 35(2): 227-253. DOI:10.15388/24-INFOR553.

[21] Pise R G, Patil S. Pioneering automated vulnerability detection for smart contracts in blockchain using KEVM: Guardian ADRGAN. International Journal of Information Security, 2024, 23(3): 1805-1819. DOI:10.1007/s10207-024-00817-z.

[22] El Haddouti S, Khaldoune M, Ayache M, Ech-Cherif El Kettani M D. Smart contracts auditing and multi-classification using machine learning algorithms: an efficient vulnerability detection in ethereum blockchain. Computing, 2024, 106(9): 2971-3003. DOI:10.1007/s00607-024-01314-w.

[23] Yu D, Chen Y. The analysis of the characteristics and evolution of the collaboration network in blockchain domain. Informatica, 2021, 32(2): 397-424. DOI:10.15388/20-INFOR437.

[24] Miglani A, Kumar N. A blockchain based matching game for content sharing in content-centric vehicle-to-grid network scenarios. IEEE Transactions on Intelligent Transportation Systems, 2023, 25(5): 4032-4048. DOI:10.1109/TITS.2023.3322826.