frontiers | Frontiers in Blockchain

TYPE Original Research
PUBLISHED 01 September 2025
DOI 10.3389/fbloc.2025.1619708

OPEN ACCESS

EDITED BY
Avishek Nag,
University College Dublin, Ireland

REVIEWED BY
Grit Ngowtanasuwan,
Mahasarakham University, Thailand
Ahmad Ali Eyadat,
Irbid National University, Jordan
Lei Yang,
Shenyang University of Technology, China

*CORRESPONDENCE
Fuyang Chen,
fyangchen@126.com

RECEIVED 28 April 2025
ACCEPTED 15 August 2025
PUBLISHED 01 September 2025

CITATION
Chen F, Yao J and Xu Z (2025) Information model operation and maintenance data network security construction based on partitioned blockchain.
Front. Blockchain 8:1619708.
doi: 10.3389/fbloc.2025.1619708

Information model operation and maintenance data network security construction based on partitioned blockchain

Fuyang Chen[1]*, Jie Yao[1] and Zihan Xu[2]

[1]School of Design and Art, Yancheng Institute of Technology, Yancheng, Jiangsu, China, [2]Artificial Intelligence Program, Suzhou Institute of Technology, Suzhou, Jiangsu, China

In response to the problems of single point of failure risk and poor scalability in current network security construction, this article applies an information model operation and maintenance data network security construction method based on partitioned blockchain. The effectiveness and advantages of the partitioned blockchain method used is evaluated in a comparison with traditional models. The research results show that the overall recall rate of the adopted method for daily network attacks is very high. Compared with traditional models, the CPU and memory usage are lower, ensuring the scalability and efficiency of blockchain while achieving a more reliable information operation and maintenance.

KEYWORDS

network security, construction, partitioned blockchain, information model operation, single point of failure

1 Introduction

With the rapid development of the Internet and the deepening of digital transformation, network security issues have become increasingly prominent. In the field of information operation and maintenance (Zhang, 2021), single point of failure risk (Yang et al., 2022) and insufficient scalability have become the main obstacles restricting system security (Xiaoyi et al., 2020) and stability. The current methods for building network security (Ding et al., 2021), such as centralized firewalls and intrusion detection systems (Shijie et al., 2020), although effective to some extent, their vulnerability gradually becomes apparent when facing complex and ever-changing network environments. For example, if a critical node in a centralized system related to a construction project is attacked, the entire system may face paralysis. The field of information operation and maintenance requires the intervention of new technologies to enhance the security and reliability of information model operation and maintenance and ensure the integrity and availability of data.

Partitioned blockchain is a technology that improves system concurrency and fault tolerance by dividing data into multiple relatively independent parts (Hu et al., 2022). In information model operation and maintenance, this technology can effectively reduce the risk of single point of failure and ensure the integrity and security of data. Through decentralization (Chaoyi et al., 2024), partitioned blockchain (Xiaoqing et al., 2021) can achieve precise control over data access and enhance protection against potential threats. The network security framework constructed based on this technology can improve the scalability and reliability of the system, providing solid support for information operation and maintenance.

Partitioned blockchain is more capable of solving single point of failure and scalability challenges than existing methods because of the following reasons.

Firstly, partitioned blockchain divides the network into multiple parallel processing sub networks through sharding technology, with each sub network running independently to improve performance, thus having advantages in solving single point of failure and scalability challenges.

Secondly, Sharding technology divides the blockchain network into multiple sub networks, with each sub network independently processing transactions, significantly improving overall throughput. Sharding technology reduces the pressure on the main chain by allowing parallel processing of data.

Thirdly, Sharding technology itself disperses the risk of single point of failure. Each sub network operates independently, and even if some nodes or sub networks fail, other sub networks can still function normally. In addition, distributed storage and fault-tolerant mechanisms further ensure the robustness of the system.

The purpose of this study is to construct an information model for operation and maintenance data network security based on partitioned blockchain. By designing a framework that combines the decentralization and partitioning features of blockchain, a network security model is gradually built, and the experimental verification is conducted. Firstly, the security requirements of existing information models in the operation and maintenance process are thoroughly analyzed to identify potential vulnerabilities; secondly, a security architecture based on partitioned blockchain is designed to ensure the integrity and confidentiality of data during transmission and storage, and prevent data leakage and tampering; finally, the performance of the model in different environments is evaluated through simulation experiments, and its feasibility and effectiveness in practical applications are analyzed.

The research objectives are to significantly improve the security of information model operation and maintenance, to reduce the risk of single point of failure; to enhance the overall scalability of the system and to provide theoretical support and practical guidance for future network security construction.

2 Related work

Many researchers have proposed different solutions to address the issue of network security construction. For example, scholars such as Shao (2024) and Yu Mingzhou (2024) reduced the risk of single point of failure by applying multiple backup mechanisms, while scholars such as Muhtadi et al. (2021) and Wenwu et al. (2022) focused on improving the scalability of the system and adopting a distributed architecture to achieve efficient resource utilization. In addition, with the development of artificial intelligence technology, scholars such as Saheed et al. (2022) and Hidayat et al. (2023) adopted machine learning-based intrusion detection systems to improve response speed and accuracy to potential threats. These methods often face high complexity, high cost, and poor data privacy protection in practical applications. Multiple backups may lead to data redundancy and storage overhead, while machine learning models require a large amount of labeled data for training (Liu et al., 2021), which is not always feasible in real-world environments. Therefore, existing research has not yet solved some problems and urgently needs to find more effective technical solutions.

Through a review of relevant literature, it can be found that scholars such as Gao (2023) and Cai et al. (2021) attempted to use blockchain technology to enhance network security. The decentralized nature of blockchain provides high security in data storage and transmission processes. Scholars such as Yao (2023) and Wu et al. (2024) proposed the use of decentralization to prevent data tampering and ensure data immutability and transparency. In addition, scholars such as Peng et al. (2021) and Lang (2021) achieved automated execution of security protocols through smart contracts, further reducing the risk of human intervention. However, there is still insufficient exploration of the application of partitioned blockchain in existing research. Partitioned blockchain can play an important role in improving the system's concurrent processing capability and fault tolerance by dividing data into multiple relatively independent parts, while reducing the risk of single point failures. This article adopts a partitioned blockchain and combines the data dynamic update mechanism of information model operation and maintenance to systematically solve the single point of failure and scalability problems in information model operation and maintenance, providing an innovative solution for network security construction.

Compared with the existing literature, the new features of blockchain framework are mainly reflected in the following aspects.

2.1 Distributed storage and consensus mechanism

Blockchain adopts distributed storage technology, where data is stored in multiple nodes to avoid the risk of single point of failure. Consensus mechanisms, such as proof of work and proof of stake, ensure data consistency while reducing power concentration issues in centralized organizations.

2.2 Smart contracts and decentralized applications

Smart contracts are automatically executed through script code, supporting new application scenarios. Decentralized applications (DAPP) run in a distributed network, where data storage and operations are jointly maintained by nodes, enhancing security and reliability.

2.3 Non tampering and traceability

Blockchain ensures data immutability through hash algorithms and chain structures, while adding timestamps to achieve data traceability. The verification mechanism of adjacent blocks further strengthens the authenticity and security of data on the chain.

2.4 Cross chain and privacy protection

Support cross chain protocols to achieve value transfer and data sharing between different blockchains, while ensuring transaction

security and privacy protection through encryption algorithms such as public and private keys.

Whether the partitioned blockchain framework is based on attribute access control depends on the specific framework design. For example, Hyperledger Fabric supports attribute-based access control (ABAC), which implements permission control through login certificates (ECert) containing attribute names and values.

2.5 Integration of attribute access control and blockchain

Blockchain technology can achieve more refined permission management through smart contracts and attribute verification mechanisms. For example, the ADAC framework manages device attributes and control policies through subject contracts, object contracts, and policy contracts to achieve distributed trusted access control.

2.6 Differences in framework design

Some blockchain frameworks, such as Ethereum, use role-based access control (RBAC) to manage user access through role permissions. However, frameworks such as Hyperledger Fabric tend to lean towards attribute driven access control models.

3 Partition data management

Partition data management, as an important component of system architecture, aims to alleviate the risk of Single Point of Failure (SPOF) and enhance the system's fault tolerance, security, and scalability through data partitioning and isolation. This article adopts a partition-based storage and management strategy (Shiyu and Zhang, 2021), which divides data into multiple autonomous subsets to ensure that the failure of a single partition does not affect the entire system. This section covers multiple dimensions such as partition strategy, data mapping, data consistency, and security assurance, and provides a detailed introduction to the application of relevant technical methods and their practical effects in network security.

A multidimensional data partitioning strategy is designed based on business characteristics and security requirements. The core goal of data partitioning is to divide operational data into several independent partitions based on their sensitivity, access frequency, correlation between data, and different business scenarios. This article applies a hybrid strategy based on Hierarchical Clustering Algorithm (Zheng et al., 2022) and dynamic weighted data segmentation model (Zhang and Chen, 2021), ensuring that the coupling and distribution characteristics of data can be fully considered during data partitioning. As shown in Figure 1, by defining a distance metric function for data, the operation and maintenance data is hierarchically clustered and segmented. Each object is treated as a class, and the minimum distance between each pair is calculated. The two classes with the smallest distance are merged into a new class, and the distance between the new class and all classes is recalculated. The first two processes are repeated until all classes are finally merged into one class.

The distance measurement function adopts Euclidean Distance or Cosine Similarity, and the specific formulas are as follows:

The formula for calculating the Euclidean distance between two data objects $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$ is indicated by the Equation 1:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

If cosine similarity is used, its formula is indicated by the Equation 2:

$$\mathrm{Sim}(x, y) = \frac{x \cdot y}{\| x \| \| y \|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2}$$

Data partitioning is optimized by minimizing the distance within partitions and maximizing the distance between partitions.

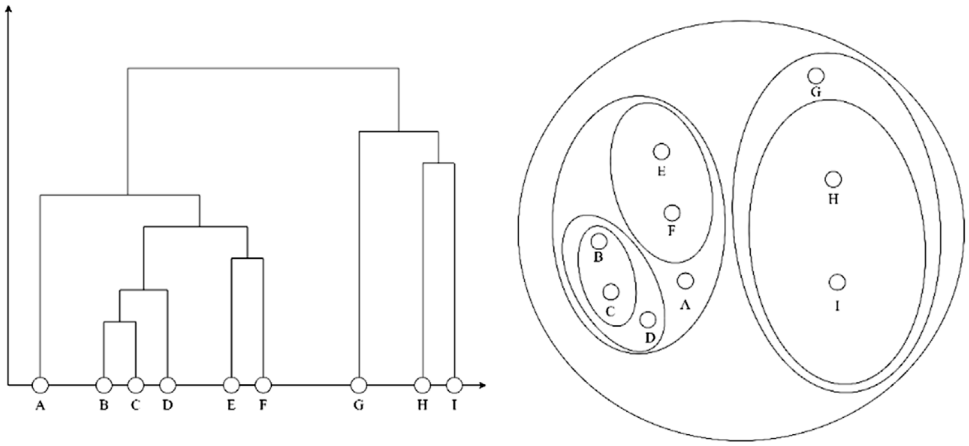

FIGURE 1
Hierarchical clustering algorithm.

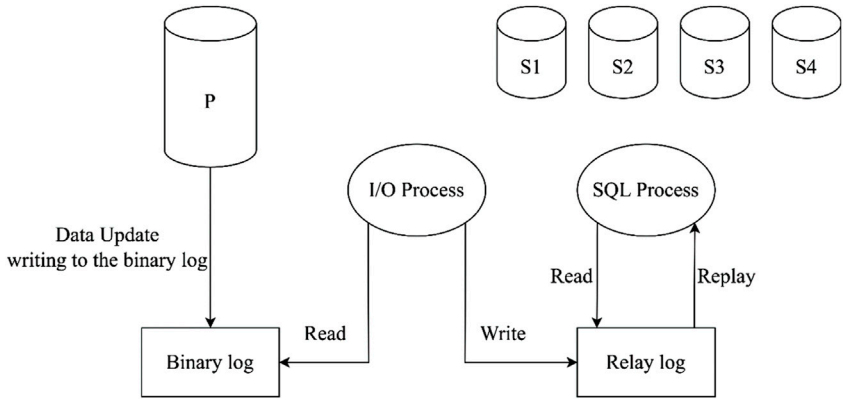

FIGURE 2
Basic process of master-slave replication.

The system objective is to minimize the total distance W within the partition, and its objective function is indicated by the Equation 3:

$$W = \sum_{k=1}^{K} \sum_{i,j \in C_k} d\left(x_i, x_j\right) \tag{3}$$

K is the total number of partitions; C_k is the dataset in the k-th partition; $d(x_i, x_j)$ is the distance between any two data points x_i and x_j.

This model uses a Multi-Criteria Optimization (MCO) mechanism to select the optimal partitioning scheme, ensuring efficient system load balancing and data distribution. To further optimize resource scheduling and access control after data partitioning (Das et al., 2020), this article adopts Sharding technology (Hellings and Sadoghi, 2021). In the Sharding architecture, the system divides data into different Shards based on business requirements, each with independent storage and computing resources. Through Sharding, it ensures that each partition can independently scale and handle its related transaction load in high concurrency scenarios, greatly improving the scalability of the system.

To ensure the efficiency of distributed storage and retrieval of partitioned data, this study applies Distributed Hash Table (DHT) as the core data storage and mapping mechanism. The hash function of DHT (Yanzhe et al., 2021) maps data blocks to unique storage nodes. Through this decentralized mapping method, the storage location of data does not depend on a single node, thereby avoiding data loss caused by single point failures. To improve the efficiency of data retrieval, the storage and query time complexity of DHT is controlled at $O(\log N)$ (N is the number of nodes), which can effectively reduce retrieval latency in large-scale distributed networks.

In the data mapping process of a DHT, the hash function h maps each data object x to a node N in the storage node set, and the mapping formula is as follows indicated by the Equation 4:

$$h(x) = x \bmod N \tag{4}$$

To ensure load balancing and even distribution of data, the hash function h selects a consistent hash algorithm with good distribution characteristics as indicated by the Equation 5. Assuming m is the total number of nodes, the hash function h maps the data x to the hash space of $[0, 2^m - 1]$:

$$h(x) = \text{SHA-256}(x) \bmod 2^m \tag{5}$$

A replica management strategy is employed to address the reliability and consistency issues of data across different partitions. For high-sensitivity data and high-frequency access data, the system maintains synchronous replicas among multiple nodes and adopts a Master-Slave replication mechanism (YANG Jun, 2022) to ensure that the master node can update the slave node replicas in real-time every time data is written. As shown in Figure 2, a master-slave replication mechanism is used for data replica storage, where each master node P maintains a replica set of R slave nodes $S_1, S_2, ..., S_R$.

The write operation first updates the master node, and then propagates to the slave nodes through the following formula as indicated by the Equation 6:

$$S_i = P \text{for} i \in [1, R] \tag{6}$$

For low-frequency access data, a decentralized redundant storage strategy is adopted to store asynchronous replicas at different regional nodes. This not only improves data persistence, but also reduces storage load without affecting overall performance.

In a multi-partition system, data consistency and reliability are crucial. Therefore, this article chooses the Practical Byzantine Fault Tolerance (PBFT) algorithm (Bin and Zhang, 2024) to ensure that consensus can be reached on most nodes during data updates.

As shown in Figure 3, the consensus reached by PBFT is based on a core three-stage communication protocol: pre-preparation stage, preparation stage, and commit stage.

Pre-preparation stage: in this stage, the master node is responsible for broadcasting a message to all participating child nodes. According to the rules, an honest master node does not send two messages with the same sequence number but different content. Therefore, if a child node receives two messages with the same node number but carrying different content, they reject these requests to prevent potential data inconsistencies.

Preparation stage: after the preparation stage is completed, each child node sends a preparation message to all other child nodes,

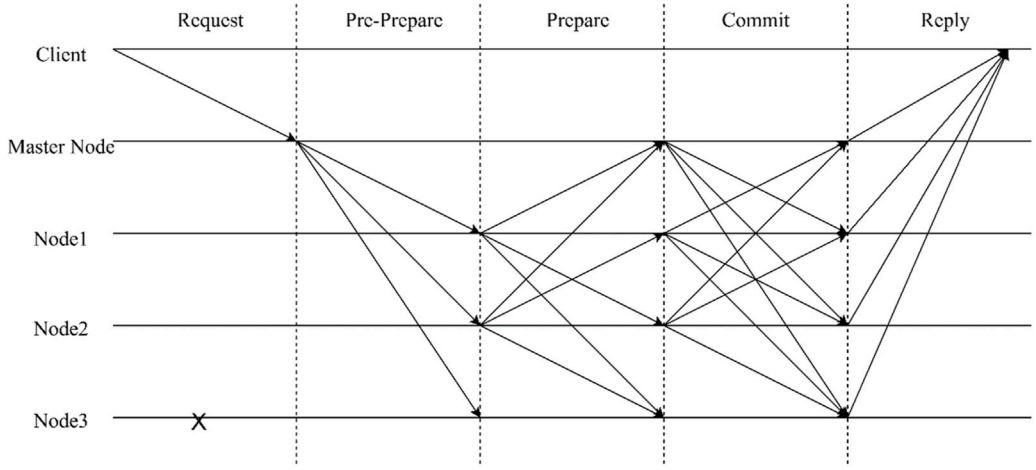

FIGURE 3
Practical Byzantine fault tolerant algorithm.

indicating that it has accepted the message from the preparation stage and is ready to move on to the next stage. In this process, if a sub node receives preparation messages from at least 2f+1 different nodes (f is the maximum number of tolerable faulty nodes), it can be considered that the preparation stage has been successfully completed, indicating that most nodes have agreed to the content of the preparation stage.

Commit stage: like the preparation stage, after receiving enough preparation messages, the child nodes begin sending commit messages. Once a node collects 2f+1 submission messages from different nodes (including the one sent by itself), it can be determined that most nodes have confirmed and are ready to perform the operation. This node can safely execute client requests and update its status or database records.

The state change function calculated for each node i in the v-th vote in each stage is indicated by the Equation 7:

$$v_i = \sum_{j=1}^{n} w_j \cdot m_j \tag{7}$$

m_j represents the voting value of the j-th node, and w_j is the weight of the corresponding node. The prerequisite for achieving consensus in the entire network is $f \leq \frac{n-1}{3}$ faulty nodes (that is, Byzantine nodes). Among them, f is the number of faulty nodes, and n is the total number of nodes.

The overall time complexity of PBFT is indicated by the Equation 8:

$$T(n) = O(n^2) \tag{8}$$

This is mainly because in the three stages of communication at the core, the communication volume between each node and the other nodes increases at the square level.

PBFT uses a voting mechanism and a multi-round communication protocol to prevent data tampering and forgery even if malicious nodes exist in the system. As shown in Figure 4, this article combines the Merkle tree (Zheng and Wang, 2022) structure to generate hash values from the data changes in each partition and organize them into a tree structure. This not only verifies the consistency of the data, but also effectively reduces the communication overhead of PBFT algorithm and improves its feasibility in large-scale systems.

There are significant differences between mainstream blockchain security solutions and non-blockchain methods in terms of security, decentralization, data integrity, and efficiency. Here is a specific comparison.

3.1 Safety

Blockchain achieves data immutability through decentralized structures and cryptographic safeguards such as public/private key encryption and hash functions, while traditional systems rely on a single central authority to manage data and are vulnerable to attacks.

3.2 Data integrity

Blockchain ensures transaction consistency through consensus mechanisms such as PoW/PoS, and any tampering requires more than 51% computing power, far beyond the feasibility of reality; Traditional systems may have inconsistent data due to a single point of failure.

3.3 Privacy protection

Blockchain uses technologies such as zero knowledge proofs (ZK SNARKs) to verify privacy transactions, while traditional methods (such as encrypted transmission) only protect the transmission process and cannot verify the authenticity of data.

3.4 Efficiency and cost

Smart contracts on blockchain can achieve automated execution and reduce human intervention; Traditional methods rely on manual review, which is inefficient and prone to errors.

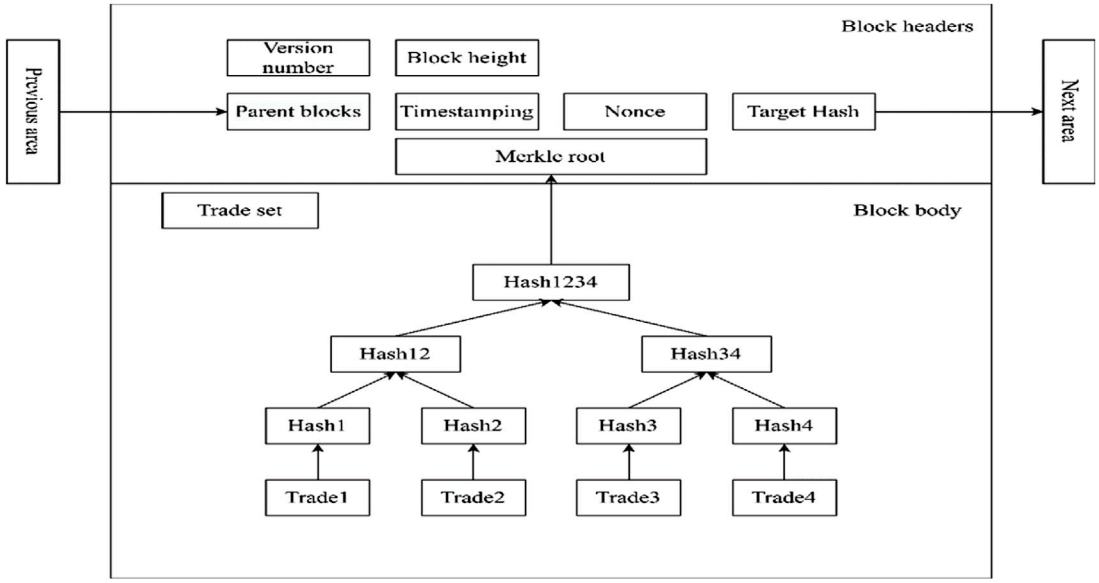

FIGURE 4
Merkle tree structure model.

3.5 Anti-attack capability

Blockchain significantly enhances its ability to resist quantum computing attacks through distributed storage and fault-tolerant mechanisms, such as PoW's power competition; Traditional methods, such as single key management, are easily cracked.

The improvement of partitioning strategy compared to traditional sharding or PBFT mechanism is mainly reflected in the following aspects.

There are significant differences in data structure, consensus mechanism, and performance, and the following are the main improvement points.

3.6 Data structure

Traditional blockchain stores data in blocks, with each block containing multiple transaction records, forming a chain structure through timestamps and encrypted links. The partition strategy is directly based on a single transaction, forming a network through reference relationships, and each node needs to verify other transactions before initiating new transactions.

3.7 Consensus mechanism

Traditional blockchain adopts the longest chain consensus mechanism, where all nodes synchronously verify the legitimacy of new blocks. The partition strategy uses multi chain mutual authentication consensus, where each node needs to verify other transactions before initiating new transactions and distributes the verification responsibility to each user in the network.

3.8 Performance

The partition strategy supports asynchronous concurrent write transactions, allowing for slight differences in node data and ultimately synchronization. Traditional blockchain requires synchronization of the entire block data, resulting in lower processing efficiency. Partition strategy improves processing speed through multi-core and multi-threaded mode, suitable for high-frequency trading scenarios.

3.9 Safety

The partition strategy enhances security through a distributed verification mechanism, where each node participates in transaction verification and reduces the risk of single point of failure. Traditional blockchain relies on centralized miners to maintain the network, which leads to trust dependency issues.

4 Execution of smart contracts

In the information model operation and data network security construction of partitioned blockchain, smart contract execution is an important mechanism for achieving network security automation. Through smart contracts, automated network security policy deployment, dynamic adjustment, and event-based response can be achieved.

The core of smart contracts is that the code logic is enforced on the blockchain to ensure that each operation follows pre-set security rules. To build a network security system for operation and maintenance data, the smart contract in this article is divided into three main modules:

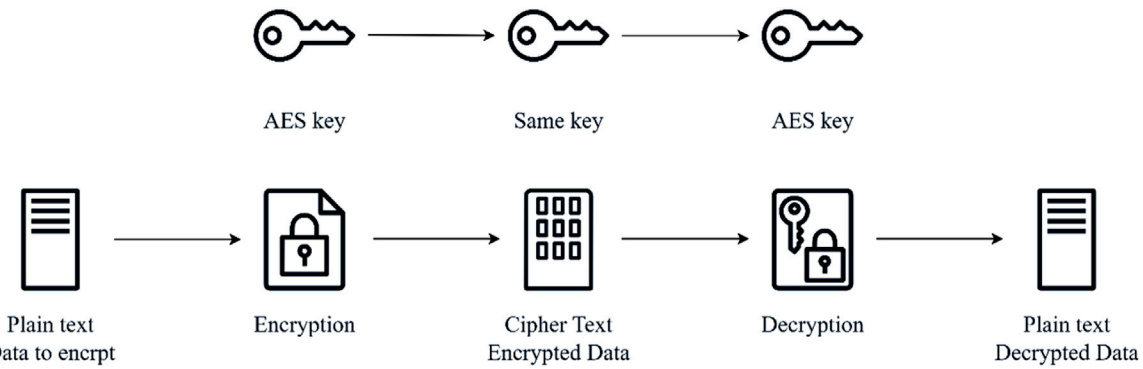

FIGURE 5
AES-256 encryption model.

Access control module: it is responsible for permission management and user authentication, ensuring that only authorized users can access and operate specific data.

Data storage and verification module: it is responsible for the storage and verification of data in blockchain and distributed storage systems, ensuring data consistency and security.

Security policy execution module: it is responsible for automating the execution of security policies and dynamically adjusting them based on real-time monitoring data.

As shown in Figure 5, the encryption algorithm is automatically called to encrypt the data before transmission. In this article, the symmetric encryption algorithm AES-256 (Wu and Rong, 2023) is used to encrypt the data and generate ciphertext $C = E_K(D)$, where K is a dynamically generated encryption key. The key is distributed to authorized users through asymmetric encryption algorithms, and the smart contract automatically manages and distributes the key to ensure that only authorized users can decrypt and access the data.

Integrity checks are performed on transmitted data through hash functions to ensure that the data has not been tampered with during transmission. Whenever data is transmitted, the contract automatically calculates the hash value $H(D)$ of the data and compares it with the receiver as indicated by the Equation 9. If the hash value calculated by the receiver is consistent with the hash value before transmission, the integrity of the data is verified to ensure that the data has not been tampered with.

$$H(D) = SHA - 256(D) \tag{9}$$

When the hash value of the data matches, the data transmission is automatically approved; if there is no match, the contract rejects transmission and issues an alert.

Smart contracts are deployed on various blockchain nodes, and the distributed nature of blockchain is utilized to ensure that even if some nodes fail or engage in malicious behavior, they can still operate normally on the remaining nodes, maintaining the overall security of the system. In partitioned blockchain, data transmission involves multiple nodes. To ensure that data is not tampered with or intercepted during transmission, a series of encryption and verification operations are automatically performed. Multiple network security control operations can be automatically executed through preset security policies.

In Figure 6, when the system detects intrusion behavior, the contract can respond immediately. For example, when a node issues an abnormal request or engages in unauthorized data modification behavior, the smart contract automatically triggers the security policy (G et al., 2020), isolates the node, and notifies the administrator for further processing.

The intrusion detection response mechanism can be described as indicated by the Equation 10:

$$R(T) = \begin{cases} \text{Isolate Node if} I(T) = 1 \\ \text{Allow Access if} I(T) = 0 \end{cases} \tag{10}$$

$R(T)$ is the response operation; $I(T)$ is the intrusion detection result; 1 represents intrusion; 0 represents normal.

To ensure that network security policies can be dynamically adjusted, smart contracts support version management and compatibility checks. When new network security policies need to be deployed, the system can achieve automated upgrades by updating smart contracts. Smart contracts use the version control mechanism of blockchain to automatically detect compatibility between old and new versions, ensuring that network security vulnerabilities are not triggered during contract updates. Each smart contract has a unique version identifier V, and when implementing network security policies, the system first checks the version information of the contract. If a new contract version is detected, the system decides whether to upgrade to the latest version through a consensus mechanism as indicated by the Equation 11:

$$V_{\text{new}} = \max(V_1, V_2, \ldots, V_n) \tag{11}$$

Through this approach, smart contracts ensure the latest security policies and system compatibility.

5 Access control mechanism

In the context of information security, it is crucial to ensure that only authorized users can access specific data. A hierarchical structure of roles is established by clearly defining the roles of system users. Each role represents a group of users with similar permissions. For example, the system can define the following roles:

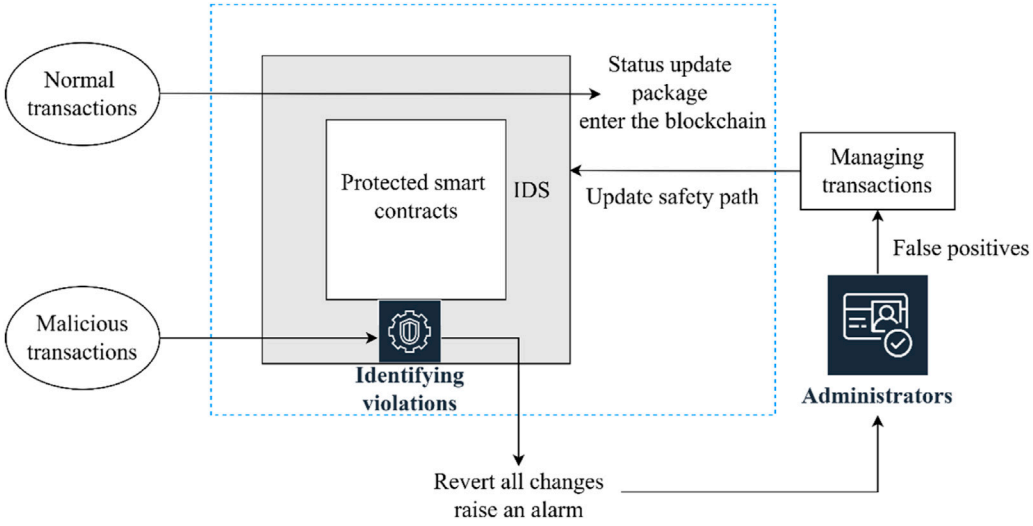

FIGURE 6
Smart contract intrusion detection system.

administrator, auditor, operator, and visitor. Each role is assigned corresponding permissions based on their functions in the business process, such as read, write, modify, and delete. To enhance the access control and security of partitioned data, this article adopts the Attribute-Based Access Control (ABAC) mechanism (Gupta et al., 2020). Unlike traditional Role-Based Access Control (RBAC), ABAC dynamically decides whether to grant access to specific data partitions based on user attributes, request context, data sensitivity, and other factors by defining finer grained access policies.

To achieve ABAC, this article defines the authorization rules for access control as follows indicated by the Equation 12:

$$A(u,d) = \begin{cases} 1, \text{if}\, \phi(u) \geq \gamma(d) \\ 0, \text{otherwise} \end{cases} \tag{12}$$

$A(u,d)$ represents the access permission of user u to data d; $\phi(u)$ represents the attribute vector of the user; $\gamma(d)$ represents the sensitivity level of the data. When the user attributes meet the security requirements for data access, access is authorized.

In the process of data encryption, the system adopts a hybrid encryption mechanism. Assuming the symmetric encryption algorithm is AES, the encryption process is as follows indicated by the Equation 13:

$$C = E_{K_s}(M) \tag{13}$$

M is plaintext; C is ciphertext; K_s is symmetric encryption key; E is AES encryption function. Asymmetric encryption is used for key exchange. Assuming that the key pair for RSA encryption is $(K_{\text{pub}}, K_{\text{priv}})$, the encryption formula is indicated by the Equation 14:

$$C_K = E_{K_{\text{pub}}}(K_s) \tag{14}$$

C_K is the encrypted symmetric key K_s, ensuring the security of key transmission under public key encryption.

As shown in Figure 7, AA (Attribute Authority) is responsible for creating, managing, and querying entity attributes; PAP (Policy Administration Point) is responsible for creating, managing, and querying access control policies; PEP (Policy Enforcement Point) queries AA for attributes based on access requests, generates attribute based access requests, and sends them to PDP (Policy Decision Point) to access resources based on the judgment results; PDP receives attribute-based access requests from PEP, receives policy sets from PAP, and then judges the access requests based on the policies, returning the judgment results to PEP. This mechanism enables the system to flexibly respond to complex business scenarios, ensuring that only authorized users can operate on specific data in a data partitioning environment.

In the process of role definition and permission allocation, the specific tasks of different roles in the system are analyzed and the required permissions are determined. According to the principle of minimum privilege, only the roles are given the minimum privilege required to execute tasks, to avoid excessive permission expansion. If the required permission for role R_i is $P_i \subseteq P$, the design is based on the principle of minimum permissions, that is, $P_i = \{p_j \mid j \in \mathcal{T}_i\}$, where $\mathcal{T}_i$ is the set of tasks for role R_i. The role permissions are updated in a timely manner when role responsibilities change, or business requirements are adjusted to ensure that security policies are consistent with actual needs.

After the allocation of roles and permissions is completed, the system implements specific access control to data through an Access Control List (ACL), which can be represented as $ACL_d = \{(R_i, P_i) | R_i \in R\}$, and d is the data object. When user U attempts to access data object d, the system performs identity authentication, calculates user role R_U, and checks $(R_U, P) \in ACL_d$. Each data object maintains an access control list, which lists the roles and corresponding permissions that can access the object. When a user attempts to access specific data, the system first verifies their identity. As shown in Figure 8, this article adopts the TOTP (Time-Based One-Time Password) password algorithm in Multi-Factor Authentication (MFA) technology. It is a time-based one-time password generation mechanism that combines passwords and dynamic verification codes to improve

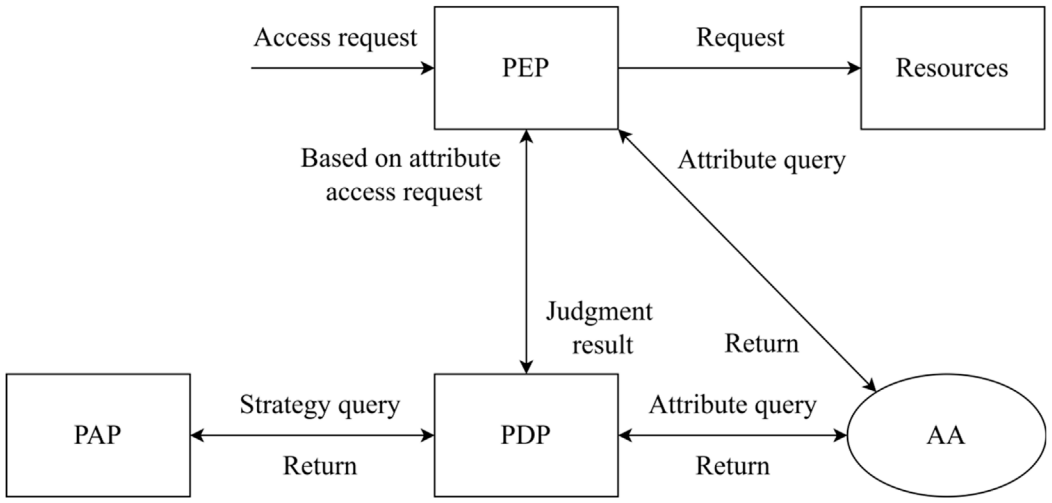

FIGURE 7
Attribute-based access control model.

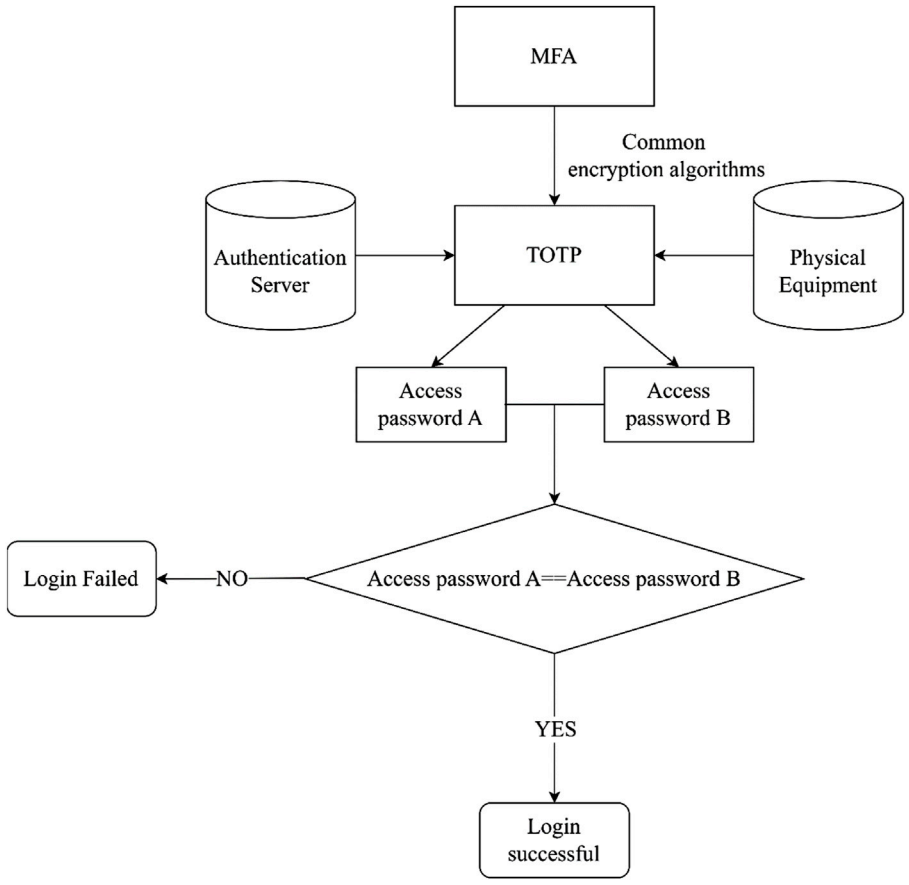

FIGURE 8
Multi-factor authentication model.

the security of authentication. After confirming the user's identity, the system queries the access control list based on their corresponding role to verify whether they have permission to access the data object. If the user does not have the corresponding permissions, the access request is denied and detailed information about unauthorized access attempts is

recorded for subsequent auditing and analysis. The processing result of the access request is fed back to the user in real-time. If the request is rejected, the system provides the reason for the rejection, enhancing the transparency of the user experience.

To further enhance security, the system keeps detailed records of all access requests, including user ID, request time, request data, operation type, etc. All access requests are $A = \{(U, d, t, o)\}$; U is the user ID; d is the requested data; t is time; o is the type of operation that needs to be recorded in log L to ensure immutability. Anomaly detection model $D(x)$ is utilized to perform real-time analysis on logs, where x represents access patterns. If $D(x)$ detects an abnormality, it triggers alarm A_{alert}. The log recording adopts an immutable method to ensure the integrity and credibility of the data. By establishing a rule-based anomaly detection model, real-time analysis of access logs can be conducted. If the system detects abnormal access patterns, such as multiple unauthorized accesses within a short period of time, the system automatically triggers an alert to remind the security administrator to investigate. The system sets up a regular audit mechanism, and security administrators regularly review access logs to analyze user access behavior and permission usage, ensuring the effectiveness and rationality of access control policies. Audit reports are generated regularly and used to evaluate and optimize access control policies.

To ensure the effectiveness of the access control mechanism, the system also implements user training and security awareness enhancement measures, providing regular security training to all users, covering access control policies, potential security threats, and countermeasures, to enhance users' security awareness. A security manual is published to clarify the responsibilities and permissions of each role, and guide users on how to use the system correctly and securely. User feedback channels are established to encourage users to report issues and suggestions during the access control process, to continuously optimize system security.

6 Construction of real-time montonotoring system

To achieve dynamic monitoring and rapid response to information model operation and maintenance data network security, the real-time monitoring system adopts a layered architecture, mainly including a data collection layer and a data processing layer.

The data collection layer collects real-time information such as network traffic, user behavior, system status, access logs, and security events through sensors and log proxies. All data is preprocessed during the collection process, filtering out irrelevant information to ensure the efficiency of subsequent analysis.

The data processing layer adopts the real-time streaming data processing framework (Flink) (Cheng et al., 2020) to perform real-time analysis on the collected data, supporting both stream processing and batch processing with the same runtime mechanism. As shown in Figure 9, Flink can calculate based on the actual occurrence time of events, which enables it to correctly handle data that arrives out of order. It also provides powerful state management functions for state management, making it easy to save and access state information in operators. Flink provides end-to-end consistency and fault tolerance through a checkpointing mechanism, ensuring that even in the event of a failure, it can recover to a consistent state and handle large amounts of data with low processing latency. It offers different levels of abstraction, including SQL, Table API, DataStream API, and DataSet API, to meet different levels of development needs.

At the data processing layer, by modeling the behavior data of normal operations, a baseline of user and system behavior is constructed. Clustering analysis methods are used to classify user behavior and determine the scope of normal activities. Machine learning algorithms are applied for anomaly detection, and real-time user and system behavior data is analyzed. Compared with behavior baselines, activities that deviate from normal patterns are identified. Using Association Rules Mining (Santoso, 2021), potential relationships between security events are identified. For example, if a user frequently attempts unauthorized access, their relationship with other security events is included in the analysis scope to determine possible attack patterns.

7 Evluation indicator design

Once a potential security threat is detected, the real-time monitoring system immediately triggers a response mechanism to monitor abnormal behavior in real-time through preset thresholds. Once an anomaly is detected, the system automatically sends an alert, including the type of event, scope of impact, and preliminary analysis results, and promptly notifies the security team. The system automatically takes measures based on the defined response strategy. For example, if malicious login behavior is detected, the system temporarily locks the account and conducts a detailed review. The execution of this strategy is managed by smart contracts to ensure automation and accuracy of responses. All security incidents and response measures are recorded in the event log for subsequent auditing and analysis. These logs are not only used for post investigation, but also provide data support for improving the monitoring algorithm.

As shown in Figure 10, two key monitoring indicators are set to evaluate the effectiveness of the real-time monitoring system:

Abnormal event detection: Figure 10A shows the number of abnormal events and security events detected by the real-time monitoring system during four time periods.

Abnormal event response: Figure 10B measures the time from event occurrence to system response, with an average response time of around 203 m.

When evaluating network security, the first focus is on network security indicators, which not only reflect the system's ability to resist various security threats, but also reflect the effectiveness of the deployed security policies.

Definition of relevant parameters: As indicated by the Equation 15, TP is the number of samples correctly predicted as positive examples; TN is the number of samples correctly predicted as negative examples; FP is the number of samples that are incorrectly predicted as positive; FN is the number of samples that are incorrectly predicted as negative examples. The

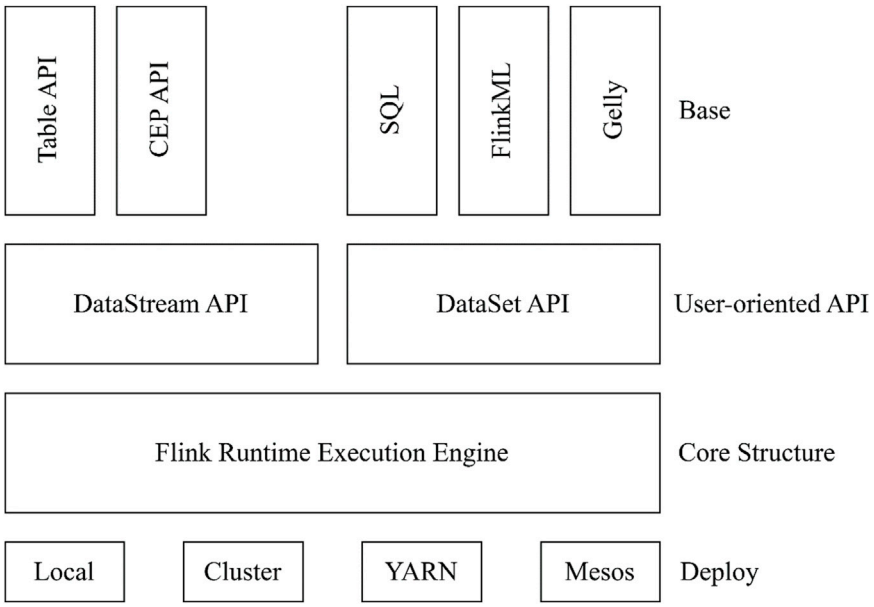

FIGURE 9
Real-time streaming data processing framework Flink.

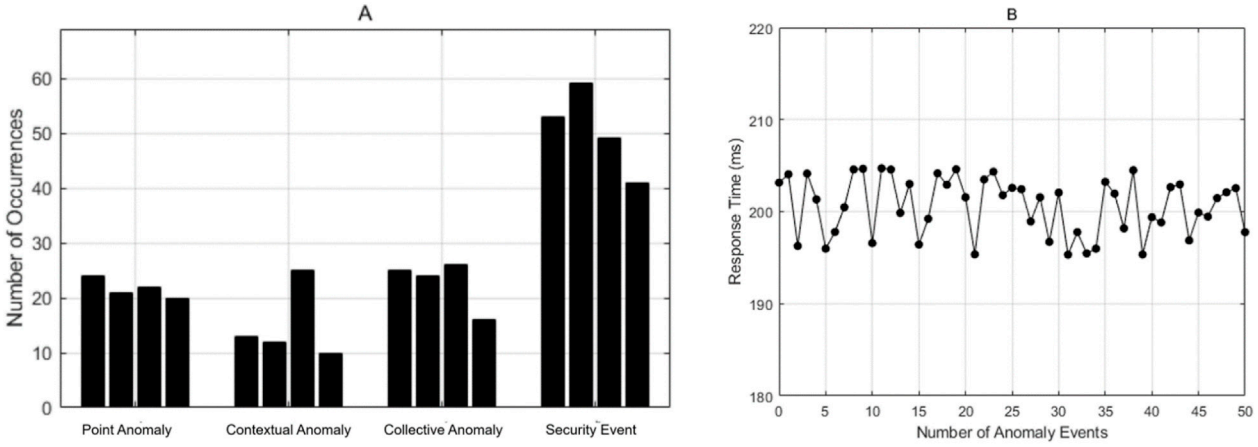

FIGURE 10
Real-time monitoring system detection. **(A)**: Statistics of abnormal event types; Figure 10 **(B)**: Response time and number of abnormal events.

harmonic average of precision and recall are F1 as indicated by Equations 16-18.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{15}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{16}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{17}$$

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{18}$$

To quantify this indicator, a series of pre-set security test cases are used, including but not limited to common attack methods such as Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and Distributed Denial of Service (DDoS) (Chen and Chen, 2020). The Metasploit penetration testing tool is used to simulate these attacks and execute attack attempts in specific scenarios through custom scripts. In a controlled environment, a zombie network simulator is used to generate abnormal traffic to simulate real DDoS attacks, to monitor the system's responsiveness.

As shown in Table 1, the results of all attacks in the experiment are recorded, and the success rate of each type of attack is calculated based on this data, with different weights given according to its potential impact. In this experiment, 90,221 samples are added for different types of network attacks, and 46,521 security data are mixed in. From Table 1, the model can still maintain a recall rate of

TABLE 1 Simulation of various real attack data statistics.

Network flow type	Number of samples collected	Accuracy	Precision	Recall	F1
XSS	11,256	96.54%	96.25%	96.54%	96.39%
CSRF	10,489	95.89%	95.65%	95.89%	95.76%
SYN Flood	11,755	96.67%	96.43%	96.67%	96.54%
Ping Flood	12,459	98.32%	98.59%	98.32%	98.45%
UDP Flood	10,976	97.73%	97.57%	97.73%	97.64%
Fragmentation Bombs	10,395	98.51%	98.30%	98.51%	98.40%
LAND	11,028	97.24%	97.01%	97.24%	97.12%
Smurf	11,863	97.83%	97.62%	97.83%	97.72%
Normal network flow	46,521	99.34%	99.54%	99.34%	99.43%
Overall	136,742	97.56%	97.44%	97.56%	97.49%

TABLE 2 Data leakage testing.

Field type	Field input quantity	Number of field detection	Missing fields	Leakage rate
Basic fields	8,564	7,867	697	8.86%
Business Field	6,925	6,458	467	7.23%
Confidential Field	5,673	5,324	349	6.56%
Overall	21,162	19,649	1,513	7.70%

up to 97.56% and an F1 value of 97.49% under different network attacks and can also maintain a precision rate of 97.44% in monitoring.

Another key security indicator is data leakage, which directly affects the protection status of sensitive information in enterprises. To accurately monitor this indicator, a comprehensive log audit is implemented to ensure that all access requests and user behavior are recorded in detail. SIEM (Security Information and Event Management) is deployed to integrate log information from multiple sources, and big data analysis techniques are used to quickly identify suspicious activities. Finally, machine learning algorithms are used to train an anomaly detection model that automatically labels and alerts for any behavior that deviates from the normal baseline. As shown in Table 2, 21,162 pieces of data are provided internally, and all data are defined as three different fields: basic field, business field, and confidential field. Various virus plugins are used for data leakage testing. From Table 2, the average data leakage rate of the virus for each field is only 7.70%.

The response time of the model is the time delay between the client initiating a request and the server returning the result, which directly affects the quality of user experience. To evaluate this, JMeter is used as the stress testing tool. As shown in Figure 11, the same task is repeatedly executed in an unloaded state until a stable average time is obtained. During the experiment, the number of concurrent connections is gradually increased until the hardware resource limit is reached.

The experimental results are plotted using software to create line graphs of response times for each stage, to visually display the trend of service performance changing with load. The initial value of the stress testing software is set to record data every 4 s, and the number of concurrent links is increased by 100 at a time until the upper limit is reached before stopping sending.

From Figure 12, the system response time fluctuates between 210 m and 230 m, and the average system response time obtained by the software is 221 m.

Table 3 shows the continuous monitoring of server CPU and memory usage during the testing process and comparison with traditional models. When the concurrency is 50, the improved model has a CPU usage rate that is about 5.02% lower and a memory usage rate that is about 4.82% lower compared to the traditional model. It has demonstrated excellent performance in the construction of information model operation and data network security in partitioned blockchain. This model has consistently shown near ideal results in multiple experiments.

Simulations of zero-day attacks are also made and indicates that partition blockchains (such as Ethereum 2.0 sharding) enhance scalability through sharding technology, but also amplify the risk of zero-day attacks.

a. Expanded vulnerability exposure: Sharding architecture increases the complexity of cross shard communication interfaces and consensus mechanisms, which may become a new target for zero-day vulnerabilities; Simulation shows that vulnerabilities in IoT devices, which account for 35% of high-risk vulnerabilities in industrial control systems, can be likened to blockchain edge nodes and are easily exploited to trigger network level attacks.

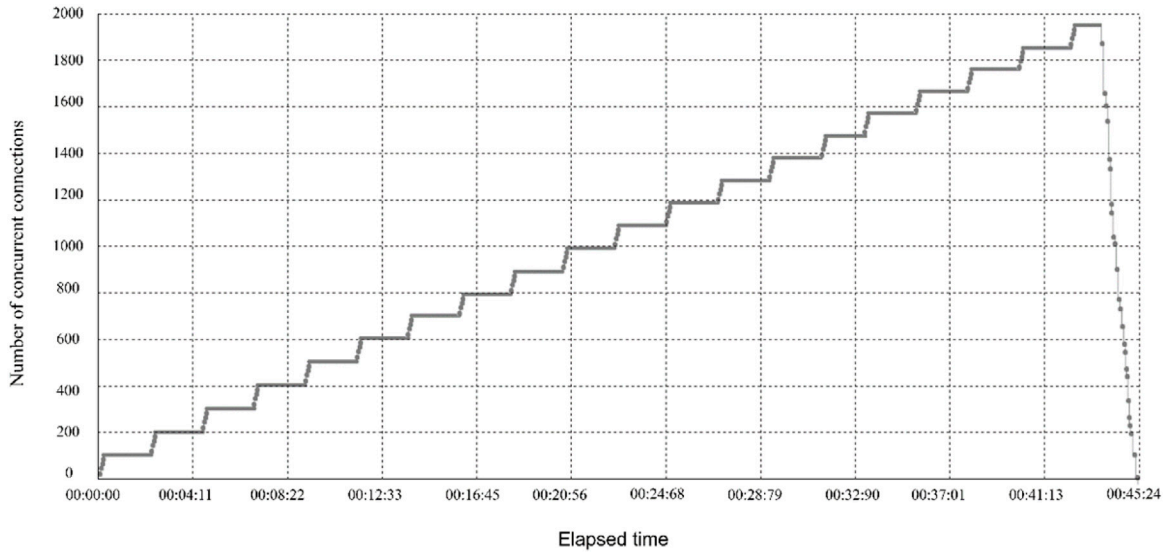

FIGURE 11
JMeter concurrent connection stress test view.

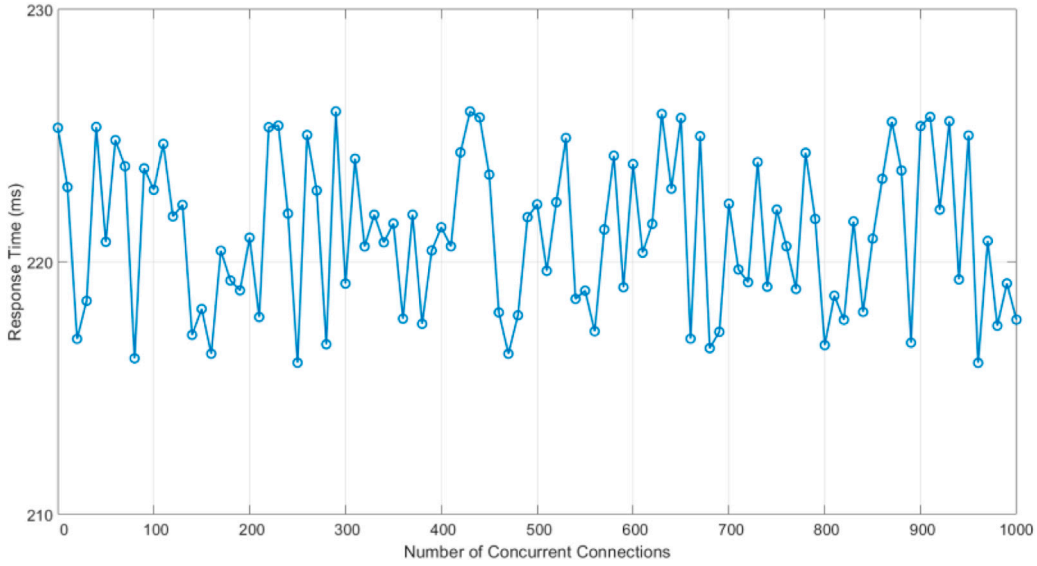

FIGURE 12
JMeter monitoring system response time.

b. Increased challenges in fixing windows: The decentralized nature of blockchain may lead to delayed vulnerability fixes; Reference data shows that the average repair cycle for traditional systems is 98 days, while industrial control systems can last up to 150 days. In sharded blockchain systems, coordinating multi partition repairs or exacerbating the "attack and defense time gap" puts 40% of zero-day vulnerabilities at long-term risk of not being fixed.
c. New attack path: Zero-day vulnerability exploitation is combining AI and big model technology (such as automatic generation of attack code), which may target smart contracts or cross chain bridges on sharded blockchain; In the Google experiment, the large model has successfully identified 0 day vulnerabilities in memory security, indicating that attack tools are rapidly evolving and threatening defense systems.

To cope with the zero-day attack risk of partitioned blockchain, the cutting-edge defense frameworks can be used for reference as follows.

a. Multi-layer protection strategy: Implement full lifecycle security requirements like the EU's. Network Resilience Act,

TABLE 3 Comparison of CPU and memory usage between traditional and improved models.

Concurrency	Improved model		Traditional model	
	CPU usage	Memory usage	CPU usage	Memory usage
50	7.23%	11.52%	12.25%	16.34%
100	7.40%	10.89%	12.43%	16.62%
200	7.56%	11.21%	12.68%	16.87%
300	7.71%	11.46%	12.91%	17.11%

including vulnerability scanning, intrusion detection, and automated response mechanisms to compress attack window.

b. AI driven threat detection: Utilize large models for vulnerability mining (such as the Google case) can enhance the proactive defense capabilities of blockchain systems; The AI security guidelines of the Five Eyes Alliance emphasize multi-layer protection and are suitable for anomaly monitoring in sharded environments.

c. Compliance and Collaboration: Strengthen the transparency of blockchain open source components and establish cross partition emergency collaboration mechanisms to resist industrial attack chains.

In summary, the simulation risk of zero-day attacks in partitioned blockchain is highlighted as a triple upgrade of "speed-scale- complexity", requiring the integration of AI enhanced defense and strict compliance frameworks to reduce threats.

Adversarial sample testing of partitioned blockchain has also been conducted, which shows that the system strengthens its defense against malicious construction traffic through multiple layers of mechanisms, with the core reflected in the following aspects.

7.1 Permission control and identity authentication enhancement

The system strictly separates the access permissions of different roles (such as users and administrators) to prevent unauthorized operations and uses multi factor authentication and key management to resist identity forgery or session hijacking, ensuring that malicious traffic cannot obtain illegal access.

7.2 Smart contract security and consensus mechanism protection

Using static analysis and fuzzy testing to detect logical vulnerabilities in smart contracts (such as re-entry attacks and overflow), while simulating scenarios such as 51% attacks and double spending attacks to verify the robustness of consensus algorithms (such as PoW/PoS) and block malicious traffic from exploiting protocol weaknesses. Five Combining AI driven behavior detection, such as kernel level file write blocking technology, can achieve high precision killing before malicious payloads are executed.

7.3 Network layer and application layer defense in depth

Integrate DDoS protection mechanisms, such as dynamically responding to sudden traffic surges through elastic bandwidth and load balancing (such as Nginx) and verify the effectiveness of P2P network communication encryption and node admission mechanisms.

For adversarial samples, the system utilizes AI models (such as generative adversarial network GAN defense) to detect high simulation traffic and prevent data leakage caused by malicious sequence injection or role-playing attacks.

7.4 Data privacy and end-to-end security

Protecting data transmission and storage privacy through zero knowledge proof, homomorphic encryption, and other technologies, combined with federated learning architecture to ensure communication confidentiality in distributed environments, and resist traffic theft or tampering.

At the level of computing power networks, deploy intrusion detection and SDN security mechanisms to ensure real-time response capabilities in the "cloud edge collaboration" scenario.

Overall, the system combines technical countermeasures (such as vulnerability detection) with operational countermeasures (such as continuous monitoring) to form a comprehensive defense system against malicious traffic.

There are certainly some differences between the simulated environment and the actual network scales as described below.

7.4.1 Node size and data authenticity

Simulated environments typically contain a small number of nodes (such as within 100 nodes), and the data has been desensitized or fabricated. For example, a student performance management system only contains simulated data for 5 classes. However, the actual network environment usually involves large-scale nodes (such as thousands to tens of thousands of nodes), processing TB level unstructured data in real business scenarios, such as social media platforms requiring real-time analysis of millions of user behavior logs.

7.4.2 Performance differences in consensus mechanisms

Simplified consensus algorithms (such as PBFT) are often used in simulated environments to test basic functions, with an average TPS (transactions per second) of only 1.59, while in actual networks, using more efficient consensus mechanisms (such as Kafka) can

achieve 2.37 TPS. In addition, simulation environments cannot verify complex scenarios such as Byzantine fault tolerance and dynamic node joining/exiting in real networks.

7.4.3 Fault injection and risk-taking

Simulate the environment for testing through preset faults (such as malicious node attacks), but without causing real economic losses. In actual networks, it is necessary to deal with sudden failures, and wrong decisions may lead to the loss of transactions worth hundreds of thousands of dollars per minute. For example, in a case where a certain e-commerce promotion lost 27% of orders due to insufficient bandwidth.

7.4.4 Distributed feature verification

The simulated environment only verifies single point or cluster communication, while the actual network needs to verify multi node data synchronization consistency, cross chain atomicity, and other characteristics. For example, blockchain testing requires verifying the immutability of Merkle tree hashes, while traditional testing only verifies ACID properties.

The adaptability of large-scale distributed networks is a multidimensional and continuously evolving goal. It depends on the combination of a series of distributed algorithms, protocols, architecture patterns and emerging technologies (decentralization, DHT, Gossip, SDN, NFV, ML/AI, edge computing, etc.). Designers need to make wise choices among various trade-offs (consistency, availability, partition tolerance, efficiency, overhead, security) and follow core principles such as decentralization, locality, redundancy, and feedback control. With the continuous growth of network size and complexity, utilizing AI/ML for intelligent perception, prediction, and automated decision-making will become a key direction for building the next-generation of highly adaptive distributed networks.

Since the current partitioning relies on static clustering algorithms without addressing adjustments under dynamic data updates, therefore, it is necessary to supplement the design with dynamic partitioning strategies.

The dynamic partitioning strategy of partitioned blockchain is a data management method that combines the characteristics of blockchain and dynamic partitioning technology to address the adjustment problem under dynamic data updates. It mainly optimizes performance and resource allocation by adjusting data storage or network structure in real-time. Its core strategy includes the following key points.

7.5 Implementation mechanism of dynamic partitioning

7.5.1 Data driven partition adjustment

Dynamically partition data storage areas based on real-time loads such as transaction volume and node pressure. For example, when there is a surge in transactions in a specific partition, the system automatically splits the partition and migrates data to a new node to avoid single point congestion.

7.5.2 Time series partition management

Adopting dynamic partitioning rules like databases (such as hourly/daily partitioning), pre creating future partitions and regularly cleaning up expired data to ensure efficient storage expansion.

7.5.3 Intelligent partition merging and splitting

Based on a preset threshold (such as the upper limit of data volume), trigger partition merging (reducing fragmentation) or splitting (sharing load), combined with algorithms (such as machine learning) to predict hotspot areas.

7.6 Key technical support

7.6.1 Dynamic accumulator and cryptographic guarantee

Use dynamic accumulators (such as improved Merkle trees) to quickly verify cross partition data integrity and prevent tampering.

7.6.2 Adaptation of consensus mechanism

Dynamic partitioning needs to be coordinated with consensus algorithms: during partition adjustment, voting or BFT protocols are used to ensure consistent states between nodes and avoid forking.

7.6.3 Lightweight node synchronization

After the partition boundary changes, only the metadata (such as the new partition header) needs to be synchronized, reducing network transmission overhead.

7.7 Core optimization objectives

7.7.1 Break through the limitations of the "impossible triangle"

Dynamic trade-off between decentralization, security, and high performance: partition refinement improves processing speed (performance) but requires the addition of verification nodes to maintain security; Partition merging enhances decentralization and sacrifices some throughput.

7.7.2 Maximizing resource utilization

Allocate node resources through algorithms such as first adaptation and best adaptation to reduce memory fragmentation (dynamic partition allocation like operating systems).

The following are the core technical points of dynamic partitioning strategy for partitioned blockchain.

7.8 Dynamic shard adjustment mechanism

7.8.1 Load responsive shard quantity control

The system automatically increases or decreases the number of shards based on real-time transaction load to ensure that resource allocation matches network demand.

7.8.2 Automated management driven by smart contracts

Creating, merging, and migrating shards through smart contracts to reduce manual intervention costs. The dynamic sharding algorithm proposed by MIT can automatically perform sharding operations based on preset rules.

TABLE 4 Performance improvement effect comparison between traditional architecture and the dynamic sharding architecture.

Index	Traditional architecture	Dynamic sharding architecture
System Throughput	≤100, 000TPS	Million level TPS
Storage Scalability	Fixed Partitioning	Elastic Expansion and Contraction
Cross Shard Consistency Maintenance	High latency	Smart Contract Protection

7.9 Key technology implementation path

7.9.1 Incremental graph partitioning strategy

This strategy can model network nodes as dynamic graphs, optimize shard structures through real-time analysis of node interaction relationships, and cope with the pressure of massive data and high-frequency trading.

7.9.2 Tiered data storage

The system supports automatic cooling of hot and cold data, and regularly creates/deletes sub tables through dynamic partitioning rules to reduce storage costs.

7.10 Performance improvement effect comparison between the traditional architecture and the dynamic sharding architecture (as indicated by Table 4 below)

The dynamic partitioning strategy solves the scalability bottleneck of blockchain through shard elasticity, automatic data migration, and innovative consistency protocols, providing underlying support for high concurrency scenarios such as the Internet of Things.

When blockchain data is dynamically updated, the stability of partition strategies mainly depends on efficient rebalancing mechanisms and adaptive techniques to prevent data skewing and performance degradation. A fixed number of partition architectures can significantly reduce the amount of data migration during node changes (such as adding or deleting nodes) and maintain load balancing by avoiding the use of hash mod N methods.

The dynamic adjustment strategy combined with deep reinforcement learning models the system as a Markov decision process, optimizing the number and configuration of shards in real-time to adapt to changes in block size, block generation time, network load, etc., improving system throughput and fault tolerance.

Sharding technology uses spatial partitioning (such as hash distribution) and temporal sharding (such as Merkle Tree structure), combined with differential compression and hierarchical management of hot and cold data, to compress storage space, reduce read and write latency, and ensure stability under high concurrency data updates. These mechanisms collectively ensure the data consistency of blockchain in the dynamic changes of the network.

Finally, the Partitioned blockchain technology can also strengthens sensitive data privacy protection through the following core measures.

a. Distributed storage: Data is stored in a decentralized manner among network nodes to avoid single point of failure and centralized leakage risks, ensuring that even if individual nodes are attacked, the overall data remains secure.
b. Advanced encryption technology: using asymmetric encryption (such as public and private key mechanisms), only authorized users can access sensitive information, preventing data from being illegally cracked.
c. Non tampering: Once data is written into the blockchain, it is permanently recorded and cannot be changed. Any tampering behavior will be detected and rejected by the node network, ensuring data integrity.
d. Smart contract control: By automatically executing programs with preset conditions, data access permissions are restricted to ensure that only authorized operations can trigger data flow.
e. Privacy protection technology: integrating mechanisms such as zero knowledge proof and homomorphic encryption to verify their effectiveness without exposing the original data, achieving a "usable but invisible" data environment.

8 Conclusion

This article proposes a method for constructing network security for sustainable information model operation and maintenance data based on partitioned blockchain. Through steps such as decentralized data storage, automated execution of smart contract policies, role-based access control, and real-time monitoring mechanisms, the security and scalability of the information operation and maintenance system are effectively improved. The research results indicate that this method not only significantly reduces the risk of single point of failure, but also enhances the overall security protection capability of the system and improves the efficiency and scalability of blockchain technology while ensuring security.

Compared to the previous studies, the results of the study in this article shows that there are significant differences between the partitioned blockchain method and non-blockchain methods in terms of security, decentralization, data integrity, and efficiency, in which the partitioned blockchain method has much stronger performance. This result is certainly also better result.

However, this study also has certain limitations, such as inadequate response to complex attack patterns in large-scale distributed networks and the need for further optimization of adaptability to networks of different scales. Future research can explore more efficient consensus algorithms to support a wider range of application scenarios such as the related applications on various construction projects and strengthen defense measures against emerging threats, providing more comprehensive security for information operation and maintenance data.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

FC: Writing – original draft, Formal Analysis, Project administration, Conceptualization, Methodology. JY: Funding acquisition, Writing – review and editing, Resources, Data curation, Investigation, Visualization. ZX: Writing – review and editing, Validation.

Funding

The author(s) declare that financial support was received for the research and/or publication of this article. This research was supported by Research on the Service Ecology and Optimization Strategy of Jiangsu Design Museum Empowered by Metacosmos, Jiangsu Province, China, 2022 (22YSB022); Research on Digital Ecology and Scenario-based Service Strategy of Design Education Resources in Local Colleges and Universities, Yancheng College of Technology, Jiangsu Province, China, 2022.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Generative AI was used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

References

Bin, L. I., and Zhang, X. (2024). GBFT: an improved practical byzantine fault tolerant algorithm. *Comput. Digital Eng.* 52 (1), 87–93.

Cai, G. U. O., Xuran, L. I., and Chen, Y. (2021). Overview of the application of blockchain technology in the internet of things. *Chin. J. Internet Things* 5 (1), 72–89.

Chaoyi, Q. I. N., Zhang, Y., and Fang, B. (2024). Survey of RPKI decentralized security enhancement technology. *J. Commun.* 45 (7), 196–205.

Chen, H., and Chen, J. (2020). Distributed denial of service attack detection of internet of things based on statistics. *J. Jilin Univ. Eng. Sci.* 50 (5), 1894–1904.

Cheng, D. I., Yang, Z., and Han, Y. (2020). Real-time processing and servitization system for streaming data. *J. Chongqing Univ.* 43 (7), 75–83.

Das, A. K., Nikum, A. K., Krishnan, S. V., and Pratihar, D. K. (2020). Multi-objective bonobo optimizer (MOBO): an intelligent heuristic for multi-criteria optimization. *Knowl. Inf. Syst.* 62 (11), 4407–4444. doi:10.1007/s10115-020-01503-x

Ding, Z., Kai, L. I. U., and Liu, B. (2021). A review of research on network security knowledge graph. *J. Huazhong Univ. Sci. Technol. Nat. Sci. Ed.* 49 (7), 79–91.

Gansen, ZHAO, Zhijian, X. I. E., and Wang, X. (2020). Contract guard: an intrusion detection system for smart contracts on ethereum blockchain. *Chin. J. Netw. Inf. Secur.* 6 (2), 35–55.

Gao, J. (2023). Research on network security technology based on blockchain. *Electron. Commun. Comput. Sci.* 5 (5), 76–78.

Gupta, M., Awaysheh, F. M., Benson, J., Alazab, M., Patwa, F., and Sandhu, R. (2020). An attribute-based access control for cloud enabled industrial smart vehicles. *IEEE Trans. Industrial Inf.* 17 (6), 4288–4297. doi:10.1109/tii.2020.3022759

Hellings, J., and Sadoghi, M. (2021). ByShard: sharding in a byzantine environment. *Proc. VLDB Endow.* 14 (11), 2230–2243. doi:10.14778/3476249.3476275

Hidayat, I., Ali, M. Z., and Arshad, A. (2023). Machine learning-based intrusion detection system: an experimental comparison. *J. Comput. Cognitive Eng.* 2 (2), 88–97. doi:10.47852/bonviewjcce2202270

Hu, Z., Leng, S., and Yuan, S. (2022). Intelligent O&M management method based on BIM and data. *J. Tsinghua Univ. Nat. Sci. Ed.* 62 (2), 199–207.

Lang, F. (2021). A new interpretation of smart contracts to contracts under blockchain technology. *J. Chongqing Univ. Soc. Sci.* 27 (5), 169–182.

Liu, F., Wang, L., and Xiao, D. (2021). Application of machine learning model in landslide vulnerability assessment. *Chin. J. Geol. Hazard Control* 32 (6), 98–106.

Muhtadi, A., Pandit, D., Nguyen, N., and Mitra, J. (2021). Distributed energy resources based microgrid: review of architecture, control, and reliability. *IEEE Trans. Industry Appl.* 57 (3), 2223–2235. doi:10.1109/tia.2021.3065329

Peng, QIAN, Liu, Z., and He, Q. (2021). Research on security vulnerability detection technology of smart contract. *J. Softw.* 33 (8), 3059–3085.

Saheed, Y. K., Abiodun, A. I., Misra, S., Kristiansen Holone, M., and Colomo-Palacios, R. (2022). A machine learning-based intrusion detection for detecting internet of things network attacks. *Alexandria Eng. J.* 61 (12), 9395–9409. doi:10.1016/j.aej.2022.02.063

Santoso, M. H. (2021). Application of association rule method using apriori algorithm to find sales patterns case study of indomaret tanjung anom. *Brill. Res. Artif. Intell.* 1 (2), 54–66. doi:10.47709/brilliance.v1i2.1228

Shao, H. (2024). Research on reliability technology of civil aviation communication network. *Electron. Commun. Comput. Sci.* 6 (4), 163–165.

Shijie, J., Lu, Z., and Du, D. (2020). Review of network intrusion detection technology. *Chin. J. Inf. Secur.* 5 (4), 96–122.

Shiyu, S. H. U., and Zhang, Q. (2021). Research on partition management mode of parts warehouse based on COI model. *Logist. Eng. Manag.* 43 (3), 78–80.

Wenwu, Yu, Qian, X. U., and Lei, W. (2022). Smart distribution grid information system and distributed energy storage planning method based on distributed architecture. *Sci. CHINA Technol. Sci.* 50 (6), 811–818.

Wu, X., and Rong, X. (2023). Data encryption technology in computer network information security. *Eng. Technol. Innovation Dev.* 1 (1), 174–176.

Wu, N. A. N., Wang, R. U., and Zhang, Z. E. M. (2024). Quality data control of whole process of model project based on blockchain perspective. *Eng. Manag.* 5 (9), 86–88.

Xiaoqing, C., Yao, D., and Liang, Z. (2021). Blockchain principles and core technologies. *Chin. J. Comput.* 44 (1), 84–131.

Xiaoyi, H. U., Wang, R., and Wang, X. (2020). Review on the development of model-based security and reliability analysis techniques for complex systems. *J. Aeronautics Astronautics* 41 (6), 140–151.

Yang, Y., Wan, W., and Zhang, S. (2022). Blockchain consensus mechanism for heterogeneous identity alliance risk assessment model. *J. Appl. Sci.* 40 (4), 681–694.

YANG Jun (2022). Analysis and scheme design of information system disaster recovery technology. *Railw. Comput. Appl.* 31 (8), 52–56.

Yanzhe, A., Zhu, Y., and Wang, J. (2021). Analysis of applicable conditions of distributed hash table in big data scenario of internet of things. *Chin. J. Comput.* 44 (8), 1679–1695.

Yao, Y. (2023). Research on data security sharing methods of enterprise fintech from the perspective of blockchain technology. *J. Int. Econ. and Manag.* 4 (5), 112–114.

YU Mingzhou (2024). Analysis on the effective application of transmission technology in information and communication engineering. *Smart City Appl.* 7 (7), 44–46.

Zhang, ming. (2021). Centralized IT O&M information retrieval algorithm based on bayesian network. *J. Jilin Univ. Inf. Sci. Ed.* 39 (5), 576–582.

Zhang, L., and Chen, X. (2021). Feature selection algorithm for dynamic weighted conditional mutual information. *J. Electron. and Inf. Technol.* 43 (10), 3028–3034.

Zheng, L., and Wang, X. (2022). Secure cloud storage system based on merkle tree. *J. Comput. Syst. Appl.* 31 (4), 81–90.

Zheng, L., Xu, Q., and Xiaofeng, F. (2022). Research on short-term load forecasting based on hierarchical clustering algorithm and ISA-LSSVM. *Electr. Power Demand Side Manag.* 24 (5), 51–57.