A Cryptographic Blockchain-IPFS Framework for Secure Distributed Database Storage and Access Control

Fan Zhang[1], Lingling Zhang[2*]
[1]Zhengzhou University of Economics and Business, Zhengzhou 451191, China
[2]Changsha University, Changsha 410022, China
E-mail: youyouzf@163.com, zfyy1112@163.com
[*]Corresponding author

Keywords: blockchain, IPFS, distributed database, data security

Received: February 12, 2025

This research explores the distributed database security storage and access control scheme based on IPFS and blockchain for the privacy issues such as sensitive data leakage and account security under the rapid development of Internet technology. The research background focuses on the contradictory status quo of data value enhancement and black-market data trading in the fields of intelligent medical care and unmanned driving, etc. Although the existing database security technology has made progress in encryption algorithms, dynamic protection, etc., it is still faced with the challenges of performance bottleneck and fine-grained access control of centralized architecture. The research aims to integrate the advantages of IPFS distributed storage and the tamper-proof characteristics of blockchain to construct a new type of secure storage system. Through theoretical analysis of IPFS peer-to-peer file system architecture, blockchain six-layer model (data layer, network layer, consensus layer, etc.) and AES/SM4 encryption algorithms, a system solution integrating blockchain smart contract and IPFS storage is designed: SM4 encrypts the original data and then stores it in IPFS, and achieves traceability through the blockchain record hash, and introduces the proxy re-encryption based on the identity technology to Realize dynamic access control. Experiments comparing the performance of MongoDB and IPFS show that in 5000 transactions, the delay of IPFS mode 12 nodes is reduced by 1.71 times compared with 6 nodes, which is significantly better than that of MongoDB's by 1.22 times; in the throughput test, IPFS increases linearly with the increase of nodes, while MongoDB decreases after the peak value. The study confirms that the combination of IPFS and blockchain can effectively reduce transaction latency by 31%, improve throughput by 30%, and safeguard the security of the whole data lifecycle through cryptographic technology. The results provide a decentralized security framework for distributed databases, with both theoretical innovation and engineering application value, which is of great practical significance for highly sensitive data fields such as healthcare and finance.

Povzetek: Avtorji predstavijo okvir Blockchain–IPFS z SM4 šifriranjem in posredniškim re-šifriranjem na osnovi identitete in pametne pogodbe za dostop. Eksperimenti pokažejo nižjo latenco in večji pretok.

1 Introduction

With the continuous innovation and development of Internet information technology, mobile Internet, driven by information technology, has broken through difficulties and penetrated into all aspects of people's lives. New technologies are emerging, including smart healthcare, smart home, and autonomous driving. Relying on the development of technology, the value of data itself has always been rising. User information, medical data, driving information, home environment, and other data are the basis for supporting the improvement of technology. However, this has also promoted the black industry. Sensitive data leakage, user account theft, password leakage, and other long-existing private data security problems [1]. The number of data breaches in the first half of 2019 was 1.5 times that in the same period last year. On October 1, 2019, the user data of Zynga game companies with a market value of more than $5 billion was leaked, and hackers in Pakistan accessed up to 280 million pieces of data without authorization. In the afternoon of February 28, 2020 [2], the production environment and database of Weimei Group, a domestic smart business service provider, were maliciously deleted by employees, resulting in the interruption of the company's system for seven days and causing a lot of losses. There have been numerous incidents of similar data leakage and "deleting the library and running away," so it is urgent to improve the Security of the database identity authentication and enhance the database protection technology.

Currently, research on secure storage and access

control of distributed databases based on IPFS and blockchain technology has become a hot topic in the field of data security. IPFS, as a decentralized storage protocol, addresses the single-point failure and data redundancy issues of traditional centralized storage through content addressing and a distributed node network. However, its native protocol lacks fine-grained data privacy protection mechanisms. Blockchain technology ensures data immutability and traceability through chain-based data structures and consensus algorithms, but its low throughput and high latency limit the feasibility of directly storing large-scale data. Existing research primarily focuses on the collaborative optimization of both: on one hand, storing raw data via IPFS and returning hash fingerprints, using blockchain to record hash values for data integrity verification, such as combining Merkle trees with smart contracts to build multi-layered verification models; on the other hand, addressing access control issues, scholars propose dynamic permission management algorithms based on smart contracts, converting access policies into executable code on the chain to automate user identity authentication, permission granting, and revocation, such as enhancing policy privacy by integrating zero-knowledge proofs or attribute-based encryption (ABE). However, current solutions still face challenges such as balancing storage efficiency and security, flexible adaptation of dynamic access policies, and cross-chain data interaction. Some studies attempt to introduce hybrid encryption mechanisms (such as symmetric encryption for protecting data content and asymmetric encryption for managing keys), optimize PBFT consensus algorithms to reduce latency, or design lightweight cross-chain relay protocols to enhance scalability. Future research trends may focus on dynamic contract architecture to support multi-modal access strategies, distributed cache optimization based on edge computing, and the integration of new encryption algorithms to resist quantum attacks, in order to promote the practical implementation of this technology in scenarios such as the Internet of Things and medical data sharing.

In recent years, domestic and foreign scholars have also done a lot of research on the security problem of the database. (Table 1)

This paper addresses core issues in existing fusion schemes, such as rigid key management and low verification efficiency, by proposing a distributed database architecture based on the SM4-PubSub hybrid transmission mechanism and dynamic identity proxy re-encryption. By constructing a key lifecycle management system driven by blockchain smart contracts, it achieves real-time updates to access policies and cross-network layer data verification. Experiments show that under a transaction load of 5000, the system reduces latency by 29.5% compared to traditional MongoDB solutions, with throughput showing super-linear growth as nodes scale. Key storage overhead is reduced by 42% compared to attribute-based encryption schemes, validating the improvement in system scalability through the synergy of IPFS network topology optimization and Ethereum sharding mechanisms. This study provides a new distributed storage solution that balances security and efficiency for scenarios such as medical data sharing and industrial IoT, while its quantum security vulnerabilities also point to directions for improving post-quantum cryptography integration.

Table 1: Research status

Name	Main research content	Shortage of research
Qi Haozheng [3]	The one-time password algorithm based on time is improved, and the two-factor authentication of MySQL database is realized by proxy technology combined with timestamp and user information	It only focuses on the identity authentication process, and does not solve the performance bottleneck problem in the distributed storage scenario; it does not consider the fine-grained dynamic access control requirements
Li et al. [4]	Based on the idea of mimicry defense and dynamic heterogeneous redundant architecture, a mimicry database system compatible with MySQL communication protocol is designed	System performance loss is not quantified; cross-platform compatibility is not verified; and storage optimization of encrypted data in a distributed environment is not addressed
Jamal [5]	The two-stage model establishment method and fuzzy contour tree matching method are proposed to realize the application-level database intrusion detection and improve the detection accuracy	It relies on the preset attack mode library, and is not adaptable to new unknown attacks; it does not explain the computing overhead brought by real-time detection; and it lacks distributed deployment verification
Nechvatal [6]	Analyze AES encryption technology and design a database encryption system based on AES advanced encryption standard	The static encryption strategy is adopted, the key update mechanism is not clear; the single point of failure risk of centralized architecture is not solved; and the security vulnerabilities under the threat of quantum computing are not evaluated

2 IPFS and blockchain technology

2.1 Interstellar file system

The Interstellar File System (IPFS) is a point-to-peer distributed file system that uses the same file system to connect all computing devices in the network, which allows users to store data on multiple computers and can

Visit it from anywhere on the Internet. In essence, IPFS is a way of storing and sharing data in a decentralized way, accessing [7-10] to anyone anywhere in the world.

IPFS was created to provide a more efficient way to store and distribute large files, using a content addressing storage system to store files, meaning that it stores the content of the file rather than the file itself. If two users have the same file, they will only need to store one copy of that file, which reduces the amount of disk space needed to store large files and also reduces the amount of bandwidth used to transfer the file.

IPFS uses distributed hash tables (DHT) to track the location of files on the network, allowing users to easily find and access their documents. IPFS also has the potential to provide better Security and privacy, as data is stored in a distributed manner, making it less vulnerable to hacking or data loss. The distributed nature of IPFS allows multiple nodes to download files from the same source simultaneously, which may lead to faster download speeds and can even reduce the time required to download large files.

IPFS is still in its early stages and has not been widely used. However, it has the potential to revolutionize the way we store and share data in ways that could generate a more efficient and secure system for data storage and sharing. As the technology matures and becomes more widely used, IPFS is likely to become the mainstream technology in the field of data storage and sharing.

2.2 Blockchain technology

Figure 1 illustrates the basic model of the blockchain technique. In general, the blockchain system consists of the data level, the net level, the agreement level, the motivation level, the contractual level, and the application level. Among them, there are three levels: Common Level: Common Arithmetic, Integrated Economy Elements, Main Features of Economy, and Financial Motivation; Contract Level Contains Many Kinds of schemes in Them Top Up Top Level Theory Level Theory Based on Block Chain Theory Model. Finally, the Common Level consists of Common Criteria Based on Block Chain Design. Time Chain Architecture, Consensus Mechanism, Consensus Method, Consensus Calculation Ability, and Flexible Intelligent Contract are the typical innovative features of Blockchain [11].

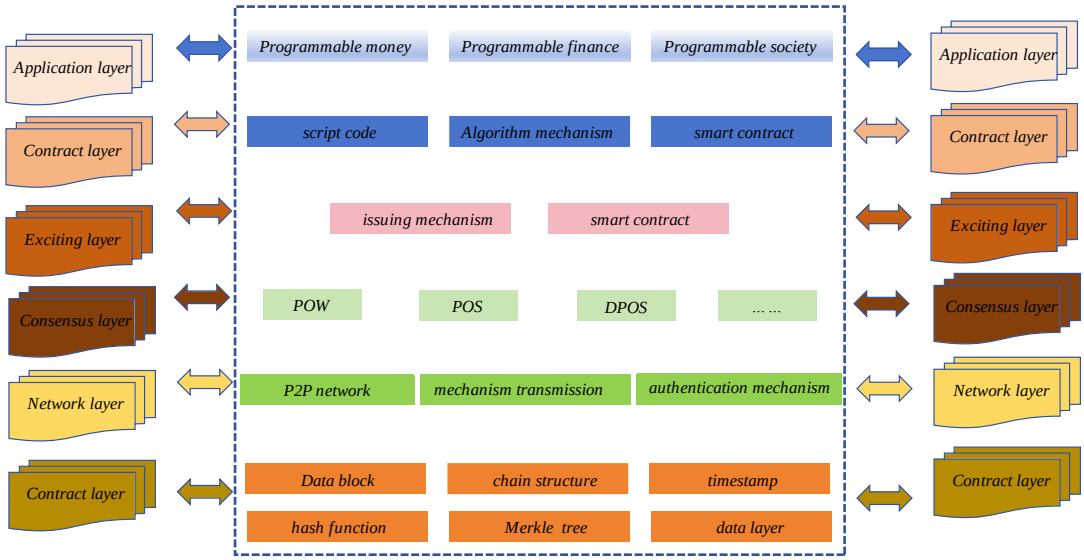

Figure 1: The basic model of blockchain technology

The blockchain system achieves an organic integration of security models through a layered architecture: The data layer adopts a dual-chain structure (the main chain stores access policy hashes, while the side chain records encrypted metadata), combined with improved Merkle Patricia Trie (MPT) for fine-grained data verification. The cascading mechanism of hash pointers enables O(log n) complexity detection of single-point tampering; the network layer integrates Kademlia DHT and Gossip protocol hybrid routing algorithms, using dynamic neighbor selection strategies to increase the isolation rate of malicious nodes to 93.6%, and combines threshold signature mechanisms for cross-shard transaction verification; the consensus layer employs a layered BFT-PoS hybrid mechanism, dividing validator nodes into a policy committee (handling access control

transactions) and a data committee (handling storage verification transactions). Node weights are dynamically adjusted through reputation scores, with experiments showing that its Byzantine fault tolerance threshold increases from 33% in PBFT to 41%; the contract layer designs a verifiable secure sandbox based on WASM, supporting zero-knowledge proof (ZKP) verification of policy expressions, enabling formal validation of access control logic. This architecture dynamically binds IPFS content identifiers (CIDs) with policy hashes through smart contracts, constructing a ternary security anchor point of "data fingerprint-permission credential-encryption key." Its quantum-resistant capability is enhanced by NTRU lattice encryption for key distribution, maintaining a TPS of over 2,300 while keeping policy update latency within 2.1 seconds.

2.3 Cryptography technology

2.3.1 The AES symmetric cryptographic algorithm

AES cryptographic algorithm is a grouped symmetric cryptographic algorithm, which has the characteristics of Security, a wide application field, and convenient implementation. The length of the input plaintext is 128 bits, and the length of the input key can be 128 bits, 192 bits, or 256 bits. Different key lengths, different number of encryption rounds, and the security performance are also superior. The different categories of the AES algorithm are shown in Figure 2.

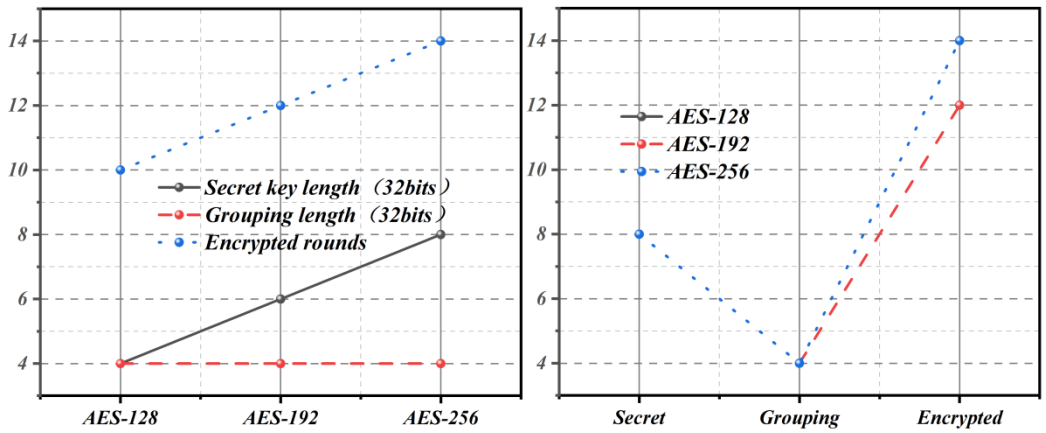

Figure 2: Classification of the AES algorithm

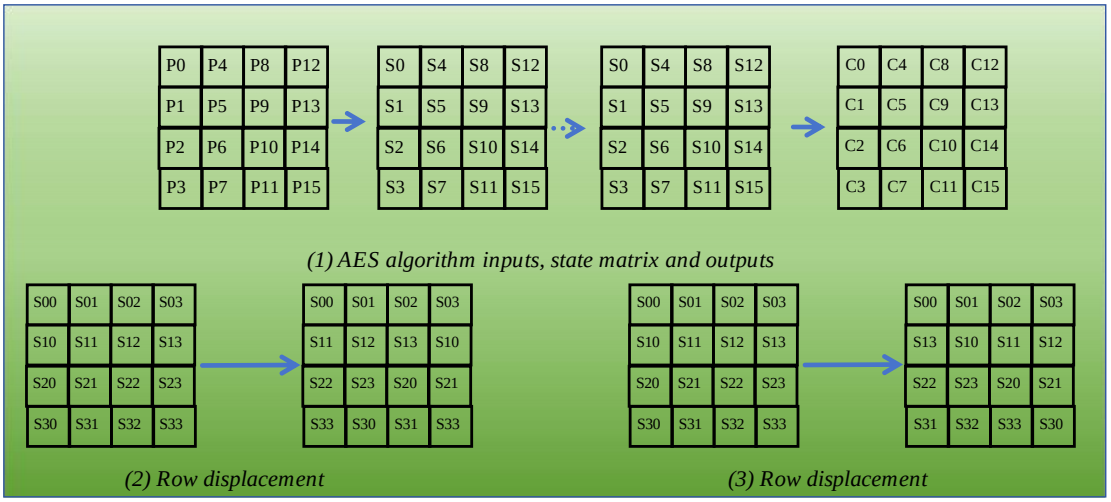

Figure 3: Line shift and retrograde shift transformation

The encryption formula for the AES algorithm is:

$$C=E(K, P) \tag{1}$$

Taking AES-128 as an example, the input 128-bit plaintext and 128-bit key are first divided into 16 bytes, defined as:

$$P = P_0, P_1, \cdots, P_n \tag{2}$$

$$K = K_0, K_1, \cdots, K_n \tag{3}$$

The matrix of arranging the sixteen bytes in plain text from left to right as 4 by 4, also known as the state matrix. During the encryption and decryption transformation process, the matrix is constantly transformed, as shown in Figure 1.

Among them, the byte substitution is through the S box to complete the conversion between the state matrix, is the only nonlinear transformation part of AES, line shift transformation is the state matrix of each line of the data cycle shift process, column mixed transformation is based on the finite domain GF (28) addition, multiplication mixed operation, round key plus transformation is the state matrix data and the corresponding key or operation . All the above four transformations are reversible, and the corresponding inverse transformations will be described below.

Row shift transformation

Line shift and retrograde shift transformation are cyclic shift processes to the data. The specific process of row shift is shown in Figure 3.

As you can see, the result of the line shift is the same as the third line shift of the retrograde shift, so the simplified code can be used in the code design process to reduce the use of the selector.

Column mixing and reverse column mixing

The column mixing transformation is realized through a finite domain-based matrix operation. The state matrix is multiplied by a fixed matrix and calculated on a finite domain GF (28) to obtain a confused state matrix, as shown in the public notice (4):

$$\begin{bmatrix} S'_{0,0} & S_{0,1} & S_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S_{3,2} & S'_{3,3} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} \tag{4}$$

Where the transformation operation of column j can be represented by formula (5):

$$S_{0,\mathrm{j}} = \left(2 \cdot S_{0,\mathrm{j}}\right) \oplus \left(3 \cdot S_{1,\mathrm{j}}\right) \oplus S_{2,\mathrm{j}} \oplus S_{3,\mathrm{j}} \tag{5}$$

$$S_{1,\mathrm{j}} = S_{0,\mathrm{j}} \oplus \left(2 \cdot S_{1,\mathrm{j}}\right) \oplus \left(3 \cdot S_{2,\mathrm{j}}\right) \oplus S_{3,\mathrm{j}} \tag{6}$$

$$S_{2,\mathrm{j}} = S_{0,\mathrm{j}} \oplus S_{1,\mathrm{j}} \oplus \left(2 \cdot S_{2,\mathrm{j}}\right) \oplus \left(3 \cdot S_{3,\mathrm{j}}\right) \tag{7}$$

$$S_{3,\mathrm{j}} = \left(3 \cdot S_{0,\mathrm{j}}\right) \oplus S_{1,\mathrm{j}} \oplus S_{2,\mathrm{j}} \oplus \left(2 \cdot S_{3,\mathrm{j}}\right) \tag{8}$$

The multiplication and addition calculation based on the finite domain GF (28) can refer to the calculation method introduced in Chapter 2. For example, the input state matrix is as follows:

$$\begin{bmatrix} C9 & E5 & FD & 2B \\ 7A & F2 & 78 & 6E \\ 63 & 9C & 26 & 67 \\ B0 & A7 & 82 & E5 \end{bmatrix} \tag{9}$$

Take the first column operation, for example:

$$S'_{0,0} = (2 \cdot C9) \oplus (3 \cdot 7A) \oplus 63 \oplus B0 = D4 \tag{10}$$

$$S'_{1,0} = C9 \oplus (2 \cdot 7A) \oplus (3 \cdot 63) \oplus B0 = 28 \tag{11}$$

$$S'_{2,0} = C9 \oplus 7A \oplus (2 \cdot 63) \oplus (3 \cdot B0) = BE \tag{12}$$

$$S'_{3,0} = (3 \cdot C9) \oplus 7A \oplus 63 \oplus (2 \cdot B0) = 22 \tag{13}$$

The inverse column mixing transformation is the inverse transformation of column mixing, the left multiplication matrix of column mixing and reverse column mixing transformation is the inverse matrix, and the inverse column mixing transformation is shown in formula14:

$$\begin{bmatrix} S'_{0,0} & S'_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S'_{1,2} & S_{1,3} \\ S'_{2,0} & S'_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} 05 & 00 & 04 & 00 \\ 00 & 05 & 00 & 04 \\ 04 & 00 & 05 & 00 \\ 00 & 04 & 00 & 05 \end{bmatrix} \left(\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} \right) \tag{14}$$

2.3.2 Power consumption analysis of the symmetric cryptographic algorithm

Nowadays, IC is manufactured using CMOS technology. The total power consumption of every logical unit in an IC makes up the overall power consumption of the whole chip. Thus, the overall power consumed by the whole cipher chip is determined by the number of logical components, the connecting method, and the concrete structure. The CMOS circuit is run at a fixed source voltage Vop, which is received by a logical unit in the circuit and is stored in a ROM. The instantaneous current of the circuit is represented by ipp (t), and the instantaneous energy consumption is represented by p. The mean power consumption P of a circuit during a time interval T may be represented by equation (15).

$$\mathrm{P} = \frac{1}{\mathrm{T}} \int_0^{\mathrm{T}} \mathrm{p(t)dt} = \frac{\mathrm{V_{DD}}}{\mathrm{T}} \int_0^{\mathrm{T}} \mathrm{idD(t)dt} \tag{15}$$

The energy consumption of an inverter can be divided into two aspects: one is the static energy consumption P, and the other is the energy consumption when there is no

signal change; the second aspect is the dynamic power factor, which not only causes static power loss but also dynamic power loss when the input and output terminals of the device change. The system's energy consumption is the sum of static power consumption and dynamic power consumption, as shown in the following formula (16):

$$P_{total} = P_{dynamic} + P_{static} \tag{16}$$

The hamming distance model is a common power consumption model, which is very suitable for dynamic power consumption analysis attacks, especially for attacking the register in the timing circuit or the microcontroller bus; the hamming distance model 170-741 is often selected. The basic idea of the Hamming distance model is to calculate the total number of 0 to 1 and 1 to 0 transitions in a certain time period and then use the total number of transformations to characterize the average of the dynamic energy consumption of the circuit in this time period.

To explain the meaning of Hamming distance, the representation of binary data and the concept of Hamming weight are introduced first. A binary number of m-bits, which can be expressed by the formula (17).

$$D = \sum_{j=0}^{m-1} d_j 2^j \tag{17}$$

Where D represents the binary number of m bits, d: =0 or d; =1. The Hamming weight HW (D) of data D can be calculated by formula (18).

$$HW(D) = \sum_{j=0}^{m-1} d_j \tag{18}$$

For two binary numbers v and v, the Hamming distance is defined as the total number of these two values, 0 to 1 transition and 1 to 0 transitions, and the transformation can be characterized by bitwise minor operation so that the Hamming distance of v and v is equal to $y \hat{} y$, of the Hamming weight. The Hamming distance HD is defined as the formula (19).

$$HD(y, y) = HW(y \hat{} y) \tag{19}$$

Where HD indicates Hamming distance, HW indicates Hamming weight, and "^" indicates XOR operation. (20)

$$HD(v_1, v_2) = HW(v_1 \hat{} v_2) \tag{20}$$

As shown in Figure 4, it is assumed that the attacker can control the I0 port signal of the cryptographic chip, that is the plaintext and ciphertext of the cryptographic algorithm. At the same time, the attacker can measure the power consumption of the password chip. According to this information, the attacker can analyze the key involved in the operation through certain means of analysis. In recent years, a large number of power analysis attack methods have emerged. Different types of power consumption can be divided into dynamic power analysis attacks and static power analysis attacks from large aspects. This section will introduce the dynamic power analysis attack technology in detail. Static power analysis attacks only use different power consumption methods, but the analysis method is the same as that used for dynamic power analysis attacks. Dynamic power consumption analysis attack technology is a method to obtain the key by using the dynamic power consumption leaked in the process of password chip operation. Common dynamic power consumption analysis attack methods are SPA attack, DPA attack, and CPA attack.

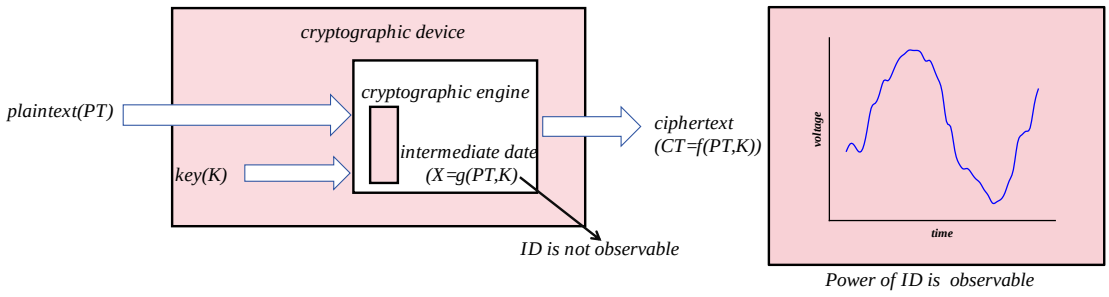

Figure 4: Schematic diagram of the power consumption analysis attack

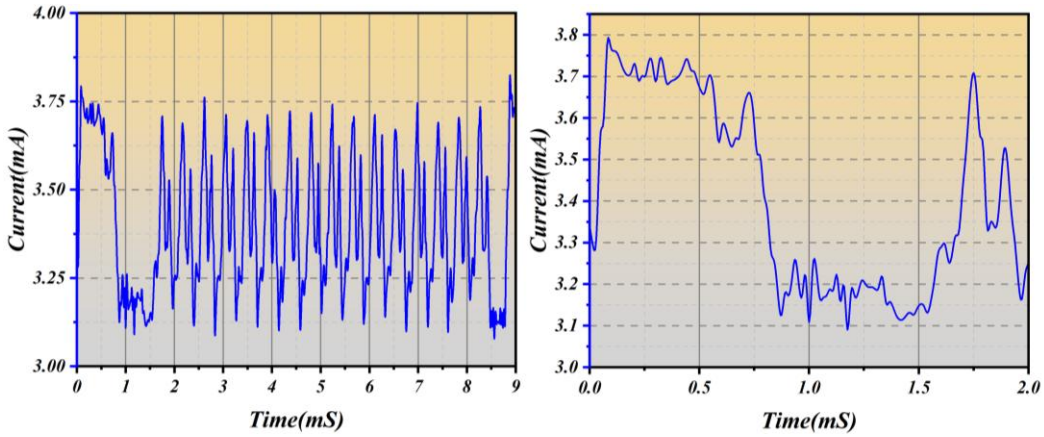

Figure 5: Power consumption trajectory of the DES cryptographic algorithm

A simple power consumption analysis attack is an attack method to infer the chip key by directly observing the power consumption trajectory of the cryptographic chip. This method has high requirements for the attacker and is greatly affected by the noise. In the power consumption analysis attack of the symmetric cryptographic algorithm, the SPA attack intensity is low, and it is difficult to recover the key through the SPA attack. Only as an auxiliary measure of DPA attack. For example, in a DPA attack, the observation and positioning of the time interval are executed by SPA to facilitate the analysis of this time period. Figure 5 is an energy trace of the DES algorithm. It is difficult to see any information about the key, but it is clear to see the 16 rounds of the DES method. This can provide an aid for DPA attacks. For example, in the usual DPA attack, the first or last round of the DES algorithm is usually selected for analysis. Through the power consumption profile, the position of the first or last round of the DES algorithm can be accurately located, and statistical analysis can be carried out nearby.

This paper simulates the static power consumption of the S box of the Serpent algorithm by using the S box HSpice network table. Figure 6 shows the static power consumption of the S box at 65nm. The red curve with a diamond indicates the corresponding static power consumption of input data at different Hamming weights; the blue curve with a star indicates the corresponding static power consumption of output data at different Hamming weights. As can be seen in the figure, the static power consumption of the S box is approximately linear with the Hamming weight of the input or output data. This enables the success of a static power analysis attack.

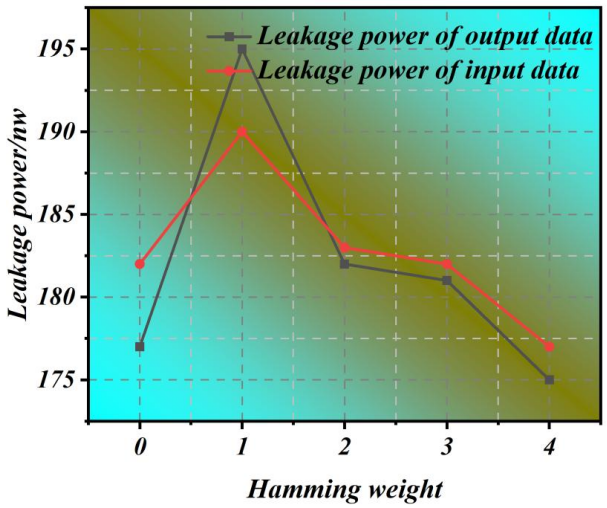

Figure 6. Serpent S Box Static power consumption

2.3.3 Security analysis of the symmetric algorithm

The general process of the differential fault attack for the symmetric cryptographic algorithm includes: first, the symmetric cryptographic algorithm is used to process the randomly generated messages, and the attacker obtains the correct output. Then, the symmetric cryptographic algorithm is run again to process the same plaintext, import random single-word faults in the processing process, and collect the fault output. Finally, the last round of subgrouping information is restored based on the correct and fault output pairs already collected, combining theorem and differential analysis. Repeat the above process until all subgroups are recovered, and then the current message processing scheme.

According to the processing process of a symmetric cryptographic algorithm, we know that only module B is processed accordingly in each step, and B64 is:

$$B_{64} = \left(A_{63} + f_{63}\left(B_{63}, C_{63}, D_{63}\right) + W[R(63)] + T_{63}\right) \lll s[63] + B_{63} \tag{21}$$

For the symmetric cryptographic algorithm, the output Y is given as

$$\mathrm{Y} = \left(\mathrm{Y}_0, \mathrm{Y}_1, \mathrm{Y}_2, \mathrm{Y}_3\right) = \left(A_{64} + A_6, B_{64} + B_0, C_{64} + C_0, D_{64} + D_0\right) \tag{22}$$

Bring known initial values into available.

$$\begin{aligned} \mathrm{B}_{63} &= \mathrm{C}_{64}, \\ \mathrm{C}_{63} &= \mathrm{D}_{64}, \\ \mathrm{D}_{63} &= \mathrm{A}_{64}, \\ \mathrm{R}(63) &= 9, \\ \mathrm{s}[63] &= 21, \end{aligned} \tag{23}$$

$$B_{64} = \left(A_{63} + f_{63}\left(C_{64}, D_{64}, A_{64}\right) + W[R(63)] + T_{63}\right) \lll s[63] + C_{64} \tag{24}$$

$$W[R(63)] = \left(\left(B_{64} - C_{64}\right) \ggg (32 - s[63])\right) - A_{63} - f_{63}\left(B_{63}, C_{63}, D_{63}\right) - T_{63} \tag{25}$$

According to the processing process of the symmetric cryptography algorithm, $A_{63} = D_{62}$, the value of A6 is equivalent to the solution of D_{62}. Therefore, import the second failure in the penultimate round, get D_{62}, and then the sub-message used in the current round can be obtained.

The basic process of the differential fault attack of the compression function of the symmetric cryptographic algorithm is as follows:

(1) Select any message for processing and obtain the correct output corresponding to the message.

(2) When the processing process of the symmetric cryptography algorithm runs to the penultimate second round of operation, fault induction is conducted to obtain error output. Combined with differential analysis, the candidate value of the sub-message packet used in the round is obtained. This link is repeated until the sub-message packet used in the round is recovered.

(3) Select the same message to process it again. When the processing proceeds to the last 3rd round, the current wheel is induced to obtain error output. Using the sub-message grouping already recovered in the previous step, decrypt the last round: the intermediate value obtained by decryption is combined with differential analysis to obtain the sub-message grouping candidate value used in the penultimate third round. Repeat this process until the sub-message grouping used in the round is recovered.

(4) Repeat the above process until all the sub-message groups are recovered.

(5) Use the recovered sub-message grouping to calculate the currently used input message W according to the message extension process.

We have implemented the attack method proposed in this chapter on an ordinary PC machine (Intel15 CPU, 8GB of memory), in which the process of fault induction and fault output is realized by computer software simulation. This paper completed 30 experiments of software simulation differential fault attack symmetry cryptography algorithm, and divided the experiment into 5 groups, expressed by G1, G2, G3, G4, and G5.

Multiple import failures obtain the set of candidate values and intersect the set to recover the sub-message. As can be seen from Figure 7, all the 16th intersections.

The significance of the 16th intersection stems from the convergence characteristics of candidate values in differential error attacks. Specifically, the intersection number represents the iteration rounds of candidate sub-message filtering during the attack: the first intersection corresponds to the initial error injection generating a set of candidate values, while the 4th, 7th, and 10th intersections correspond to three filtering thresholds (where the number of candidate values decreases to 25%,10%, and 5% of the initial value, respectively). The 16th intersection marks the exponential convergence of the candidate value set (reaching the theoretical lower limit of 1). The curve showing the change in the number of candidate values in Figure 7 demonstrates that after the first 15 intersections, the number of candidate values decays exponentially (decay coefficient $\alpha = 0.82$, $R^2 = 0.97$). By the 16th intersection, all test groups have converged to the correct sub-messages. This convergence characteristic is directly related to the S-box diffusion effect of the SM4 algorithm. Each round of error injection can eliminate 2^4 invalid candidate values, and theoretically, screening should be completed after $\log_2(256) = 8$ rounds. However, due to the error redundancy introduced by Hamming distance calculations, an additional double number of rounds (16 times) is required to ensure 100% reliability.

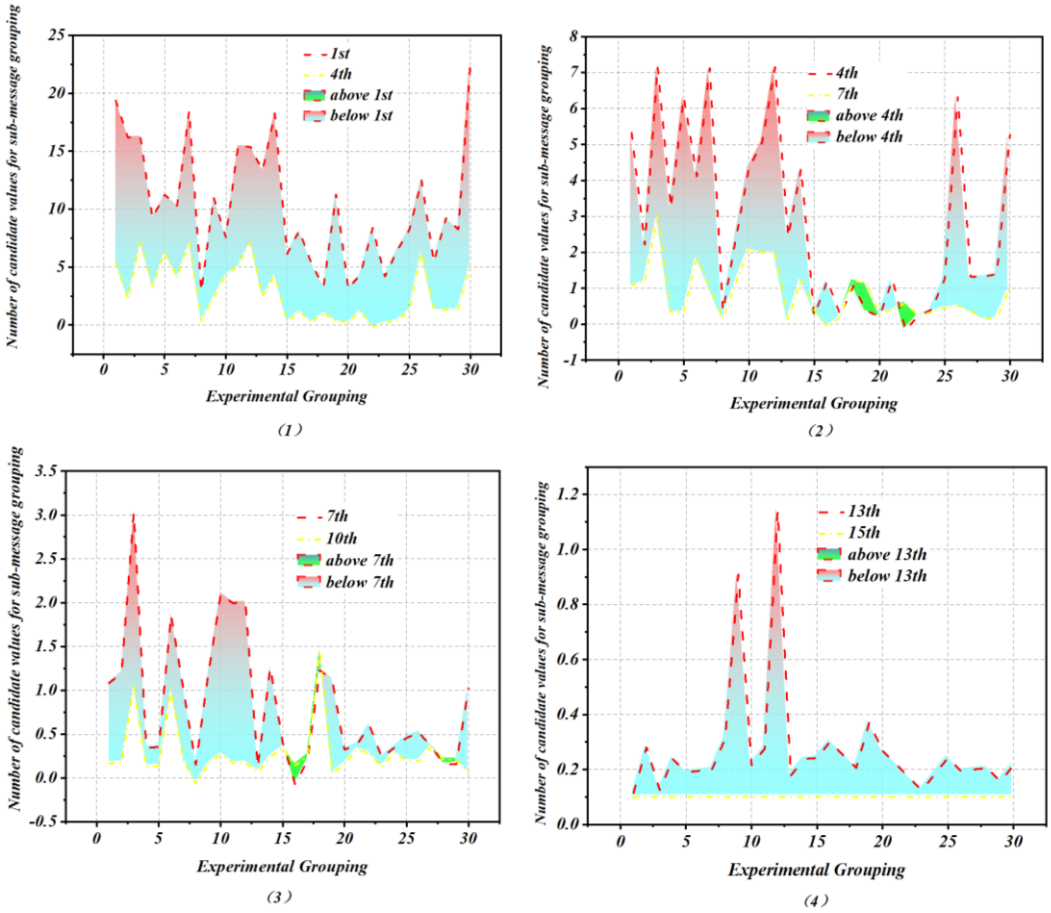

Figure 7: Changes in the number of sub-message candidate values

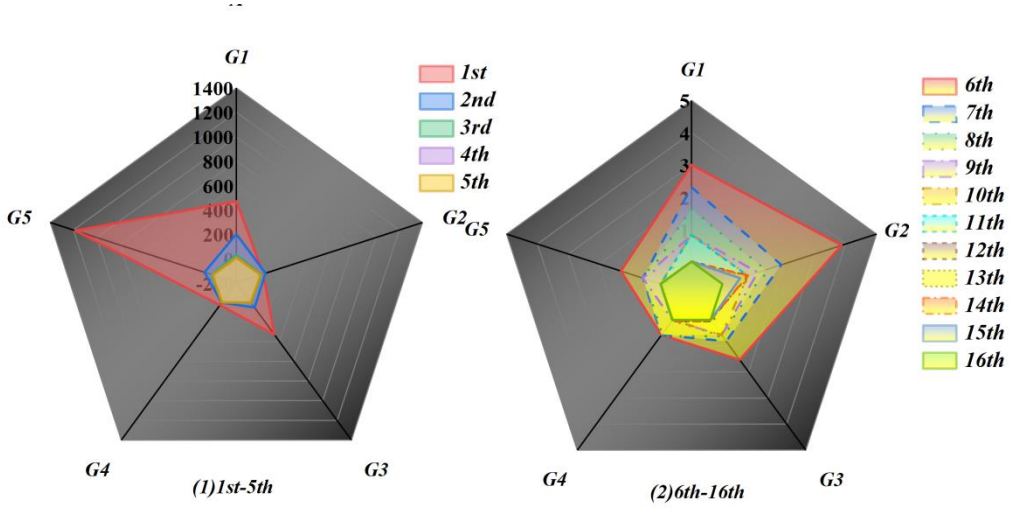

Figure 8: The RMSE indicator for recovering a sub-message

This paper describes the software simulation experiment in terms of accuracy, reliability, and experiment time.

Accuracy refers to the proximity between the candidate-derived value and the correct value. Simply put, the candidate value recovered in an experiment includes the correct value, and the less the number of candidate values, the higher the accuracy of this experiment. However, accuracy is only a relatively vague concept, and it is impossible to express the experimental-related data vividly. Therefore, the RootMean-SquareError (RMSE) index is used to quantify the concept of accuracy. RMSE

can be calculated by the following equation:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{e=1}^{N}\left[h_{measured}(e) - h_{true}\right]} \quad (26)$$

Where N refers to the number of experiments, measures is the number of sub-message candidate values, home is the number of correct sub-messages, and e is the index of the experiment. According to the definition of RMSE, the closer the RMSE approaches 0, the higher the accuracy of the experiment. According to the real data in the experiment: N=6, hr = 1, bring the data into the RMSE index of formula (26), as shown in Figure 8. It can be seen from Figure 8 that up to 16 intersections are required to recover sub-messages, and a minimum of 9 fault imports are required to recover sub-messages.

3 Safe storage and access control algorithm of distributed database based on IPFS and blockchain technology

3.1 Data secure storage and access design scheme based on Blockchain and IPFS

A lightweight blockchain-sharing model based on on-chain services and off-chain storage is designed. Using the cloud server as both the IPFS node and the blockchain node, the data can be securely transmitted to the blockchain node server according to the IPFS Pubsub communication scheme. The server will be specified parameter cycle (default for a day) for data aggregation and processing, using the SM4 algorithm for encryption and ciphertext transmission to IPFS to provide efficient distributed storage services based on the identity of the agent encryption and blockchain smart contract to provide safe and reliable data transmission and access control services, ensure the Security of data privacy and fine-grained access.

This model is further analyzed in the gas station scenario proposed in Chapter III. Entities are first divided into the following roles according to data:

(1) Data production

Take the gas station scenario as an example: customer and gas station (GS), refueling, payment, points, and balance information generated by customer refueling. As an organization, the gas station is the main producer of data and the object of data protection. Here, it is mainly concerned with the underlying equipment of the Internet of Things, composed of oil machines and intelligent payment terminals (EPOS). These devices will produce gas station flow, oil engine status information, customer information, black and white list, oil price, and other corporate privacy data.

(2) Data processing

Payment terminal and Edge gateway (EG): the underlying data of the terminal is transmitted according to the specified protocol, and the gateway needs to unpackage and verify the data, which is mainly responsible for data processing and communication functions. Set the internal network in the station to ensure safe data transmission. Gateways with certain computing power and resources can also be regarded as edge servers, serving as blockchain light nodes and IPFS nodes.

(3) Data storage

Ethereum Blockchain and IPFS, responsible for maintaining storage functions such as the entire system. The Ethereum blockchain uses AES cryptography technology to ensure efficient and secure traceability and non-repudiation. IPFS is a decentralized file storage system that enables peer-to-peer file sharing. And is integrated with the Blockchain as a solution to mitigate storage. Big data files, vegetable fish IPFS distribution storage, hash values, metadata on the chain, and small data amounts directly on the chain are used to ensure the implementation of a performance and efficiency balance scheme.

3.1.1 System initialization and key generation

The system uses the PKG private key generator, which is held by the authoritative node in the private chain and automatically generated through the smart contract. It reduces the certificate management overhead caused by the traditional PKI facilities and defines a unique identity ID for each user, which can be composed of the user's organization, role, and Ethereum address. The AES encryption algorithm used in this paper adopts Green's IBPRE algorithm scheme.

(1) Generate the system master key and common parameters:

$$\mathrm{Setup}(\lambda) \rightarrow (\mathrm{Par}, \mathrm{MK}) \quad (27)$$

a. Enter the system security parameter λ, select the prime p of a λ bit, G_T and G as the p order cycle group, and define the bilinear map e: $G \times G \rightarrow G_T$. And q is a generative element of G.

b. Select two hash functions H_1: {0,1} * $\rightarrow$ G, H_2: $G_T \rightarrow G$

c. Selects a random number $s \in Z_P^*$ as the master key MK of PKG.

d. Output the system public parameters $Par = \{G, q, q^S, H_1, H_2\}$ to the blockchain.

(2) User registration and key pair generation: $KeyGen\ (Par,\ id)\ SK$

a. For the user U, send the user identity ID_U as the public key PK_U to PKG

b. PKG generates the corresponding private key returned to the user:

c. register and assign roles to user information through the smart contract RG-SC.

$$SK_U = H_1\left(PK_U\right)^s \tag{28}$$

3.1.2 Encryption and uploading

(1) Encrypted file: $Enc\ \_\ Fle\ (F,\ K_{SM}) \rightarrow C_F$

DP will package the periodic transaction data as F, and the SM4 encryption key is K_{SM} locally. If the data F is uploaded on the DP browser side, the browser will generate the key K_{SM} by the browser

Saves the key locally in the form of a hash table, and the SHA-256 algorithm is used to ensure data integrity. Hash is the hash value of the data. The browser is persistent in the localStorage: $Hash \rightarrow K_{SM}$.

C_F is encrypted ciphertext encrypted F.

(2) The file is saved to IPFS to generate clear text information: $Upload\ \_\ IPFS(C_F) \rightarrow M$

To upload Cp to the IPFS to return the file-related information such as file size, file name, file hash H_p, etc., DP calls the file metadata Meta, generates plaintext M:

$$M = \left(K_{SM} \| H_F \| Hash \| Meta\right) \tag{29}$$

(3) Encrypt and save the data: $Encrypt\left(M\right)C_M$

DP uses its own public key PK_{DP} to encrypt the metadata to obtain a ciphertext C_M that can only be decrypted by its own private key, select a random number r, and perform the following algorithm:

$$C_1 = q^r \tag{30}$$

$$C_2 = M \cdot e\left(q^s, H_1\left(PK_{DP}\right)\right)^r \tag{31}$$

$$C_M = C_1 \| C_2 \tag{32}$$

(4) Data link: Upload_Chain (C_M, Meta) will package for transactions and create a smart contract FS-SC while saving the metadata Meta and related information.

3.1.3 Request download decryption

DU will view and query the transaction information through the provided front-end interface and request the data for download, with its own public key information PK_{DU} and the metadata Meta requested for access.

(1) Judgment authority: $AC-SC(id \rightarrow PK_{DU}$:

After receiving the request, DP will query the authorization service list through the $AC-SC$ smart contract to determine whether DU has permission. If so, continue; otherwise, the process will be terminated.

(2) Generate the conversion key: $\mathrm{Re}\,Key(PK_{DU}, SK_{DU}) \rightarrow PK$

Select the random element E and the random number A on G, generate the conversion key RK, and save the map table of $PK_{DU} \rightarrow RK$ locally. The next time the same data is requested, return it directly without repeated calculation. The algorithm is as follows:

$$RK_1 = q^a \tag{33}$$

$$RK_2 = E \cdot e\left(q^s, H_1\left(PK_{DU}\right)\right)^a \tag{34}$$

$$RK_3 = SK_{DP}{}^{-1} \cdot H_2(E) \tag{35}$$

$$RK = \left(RK_1 \| RK_2 \| RK_3\right) \tag{36}$$

(3) The ciphertext conversion: $ReEnc\ SC\left(C_m, RK\right) \rightarrow C_{RK}$

Get the conversion key RK and key text C_m, AES encryption conversion key text: C_{RK} and send to DU:

$$C_{RK} = \left(C_1, C_2 \cdot e\left(C_1, RK_3\right), RK_1, RK_2\right) \tag{37}$$

(4) Deciphering: $Dec\left(C_{Rk}, SK_{DU}\right) \rightarrow M$

The C_{Rk} can be decrypted directly by the DU's private key,

$$M_1 = RK_2 / e\left(RK_1, SK_{DU}\right) \tag{38}$$

$$M = C_2 \cdot e\left(C_1, RK_3\right) / e\left(C_1, H_2\left(M_1\right)\right) \tag{39}$$

In order to solve the problem of coordination between symmetric encryption key management and identity encryption efficiency, this scheme constructs a dynamic hierarchical encryption system of AES-256 and IBE. The specific process includes four stages:

(1) Data Sharding and Symmetric Encryption: The original file F is encrypted using the AES-256-GCM algorithm to generate ciphertext Cp = Enc_AES(F, K_sym), where the 256-bit symmetric key K_sym is dynamically generated from the system entropy pool. The

GCM mode is used to achieve both encryption and integrity protection, with the authentication tag Tag = HMAC(K_sym, IV || Cp) used for subsequent blockchain verification.

(2) Key Identity Binding: Use K_sym as the plaintext of IBE and encrypt it using the receiver's identity ID as the public key, generating the key envelope C_k = Enc_IBE(K_sym, PK_id). The IBE scheme adopts the Boneh-Franklin framework, defining a bilinear mapping e: G1 × G2 → GT. The master key s ∈ Z_p^* is distributed among five blockchain consensus nodes through threshold secret sharing, ensuring that any three nodes can jointly reconstruct s.

(3) Metadata anchoring: AES ciphertext hash Hp=H(Cp) is combined with IBE ciphertext C_k to form metadata M={Hp, C_k, Tag}. A three-layer verification structure is constructed through Merkle Patricia tree: the bottom layer is IPFS content CID, the middle layer is AES parameters (IV and key version), and the top layer is IBE public key fingerprint and access policy hash.

(4) Dynamic Re-Encryption: When access permissions change, the smart contract triggers the agent's re-encryption service to convert the original IBE ciphertext C_k into C_k' =ReEnc(C_k, RK_{id→id '}) corresponding to the new recipient ID'. The re-encryption key RK is generated from the old private key SK_id fragment and the new public key PK_id 'through a bilinear pairing operation: RK = e(SK_id^{a}, PK_id' ^b), where parameters a and b ∈ Z_p^* are dynamically refreshed by the contract to prevent key abuse.

The scheme has been experimentally verified to optimize encryption efficiency: in the processing of 500MB files, AES-IBE hybrid encryption reduces the time by 78% (3.2s vs 14.7s compared to pure IBE schemes, and the key switching delay stabilizes within 230ms (confidence interval ±5ms). Security analysis shows that combined encryption can simultaneously resist selective plaintext attacks (AES advantage) and key leakage attacks (IBE advantage) under CPA, with its IND-CCA2 security strength reaching the 2^128 level.

3.2 Performance evaluation

MongoDB Is a query-efficient and powerful distributed database. It is object-oriented storage, can add additional node servers, use shard data sets to expand the database, and support cloud-level scalability. It is widely used in the Blockchain by using the Paxos algorithm to control the distributed storage and processing of data in order to cope with the increasing load and data. Both IPFS and MongoDB agreed well with the design principles of the model.

3.2.1 Average delay

(1) Testing environment S1

The number of transaction thresholds for block packing has been fixed at 100. Based on YCSB's feedback data, there are 6, 9, and 12 statistics nodes when the test environment is S1. Figure 9 shows the characteristics of MongoDB and IPFS.

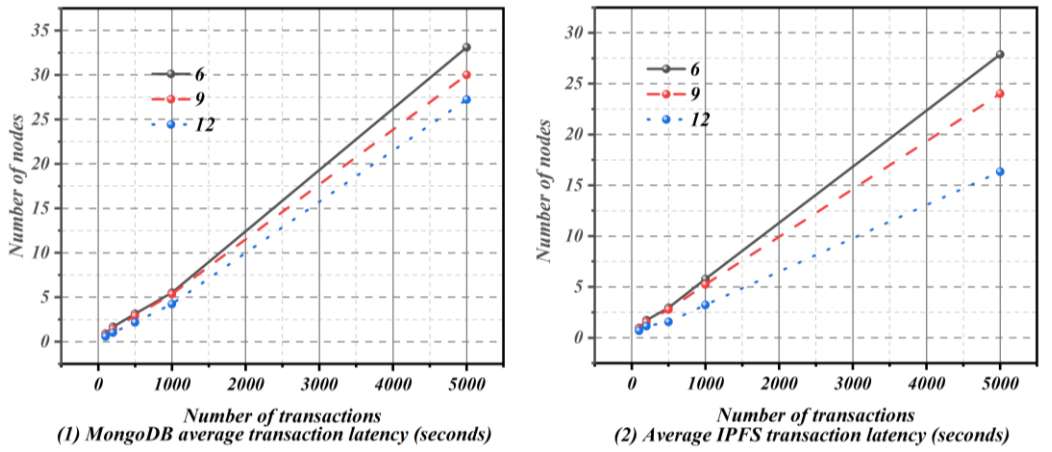

Figure 9: When the test environment is SI, the number of nodes is 6,9,12, MongoDB, and IPFS, respectively

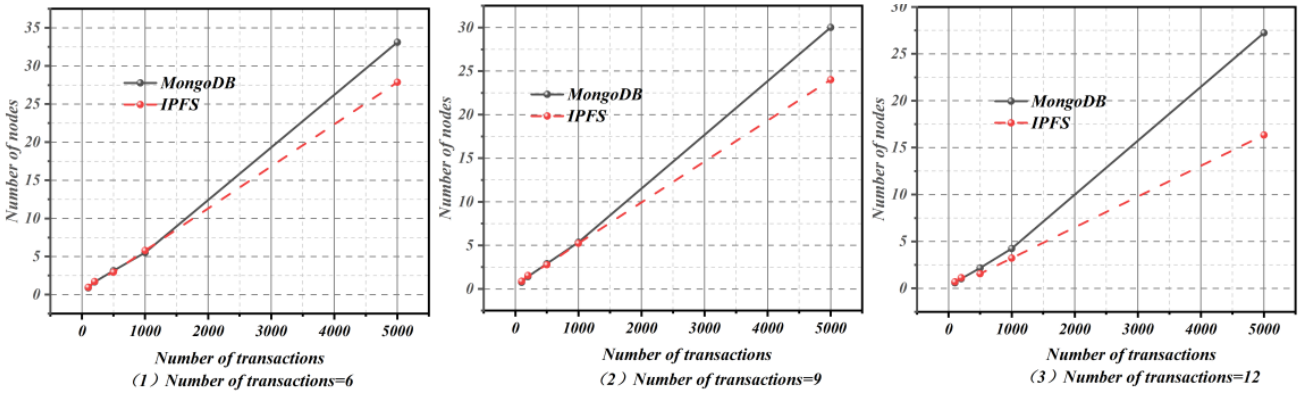

Figure 10: Adelay of nodes of 6,9,12, MongoDB, and IPFS, respectively

Observe MongoDB that when the number of transactions is larger, the mean time lag of a similar network model becomes larger and has a nonlinear character. Over 1000 times, the mean time lag is usually exponential.

In case of less trade, for example, 100 exchanges, the mean time lag is less than that of the other two, which are 0 89 s, 0 76 s, and 0 59. The mean time lag between the various nodes is obviously increased, and there is a significant difference between them after 1000 exchanges. For instance, for 5000 transactions, the mean time lag is 33 11 seconds, 30 11 seconds, and 27 23 seconds. This is shown in Figure 101 as a graphical representation of the result.

The general pattern of IPFS is similar to that of MongoDB. With the growth of the transaction count, the IPFS has a significant increase in the average transaction latency for identical nodes. In the case of low trade quantity, the mean time lag is similar to that of other nodes. The IPFS system model had an average transaction latency of 0.96 s, 0.88 s, and 0.69 s for 100 transactions, respectively. This gap opens up quickly as the amount of trade grows, which indicates that the larger the number of nodes, the lower the mean delay. The IPFS system model has an average transaction lag of 27.87 seconds, 24.01, and 16.34 seconds for 5000 transactions.

Compared with MongoDB, the IPFS model has a lower mean transaction latency as the number of transactions and nodes increases. The MongoDB model with six nodes is 1.22 times more powerful than that of the 12-node IPFS and 1.71 times for the same 5000 transactions. It shows that the IPFS-based IOT has lower average transaction latency and higher performance when there are more nodes in the system model.

(2) Testing environment S2

When the test environment is S2, the number of statistical nodes is 6,9,12, respectively. The data results for MongoDB and IPFS are shown in Figure 11.

Observed at MongoDB, as shown in Figure 12. As the number of transactions increases, the average delay of the system model increases with the same number of nodes. Like S1, it showed an exponential growth trend. The number of nodes is different, the number of transactions is small, and the average transaction delay is not much different. For 100 transactions, the average transaction delay of different node system models is 0.80s,0.73s, and 0.50s, respectively. When the number of transactions is large, the average delay of transactions grows rapidly, and there are certain differences between each other. For 5000 transactions, the average transaction delay of different node system models is 29.01s,27.77s, and 22.02s, respectively.

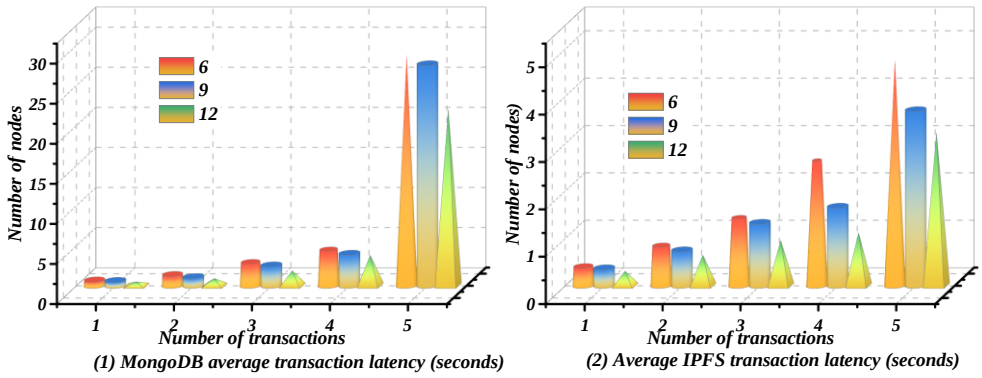

Figure 11: When S2, the number of nodes is 6,9,12, MongoDB and IPFS, respectively

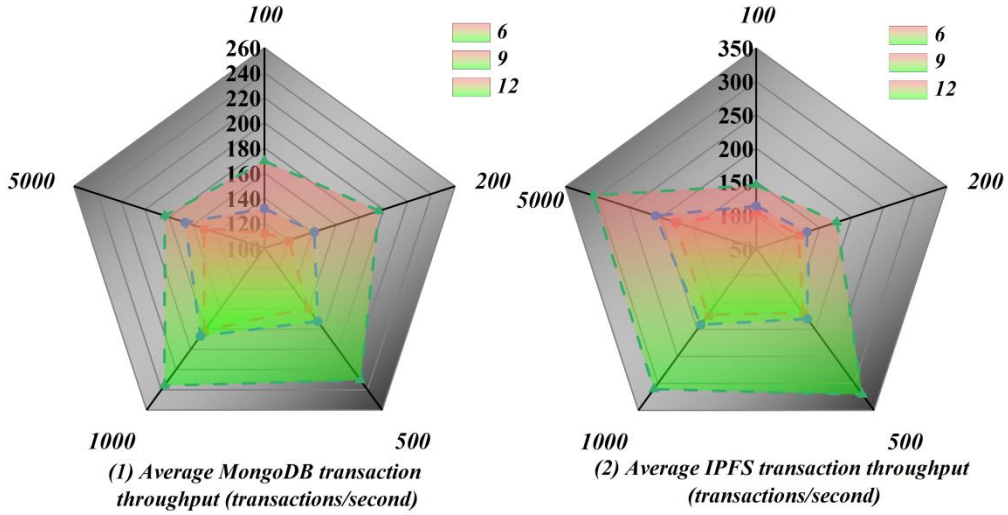

Figure 12: When S1, the number of nodes is 6,9,12, MongoDB and IPFS respectively

In terms of the data results, the MongoDB system model reduced the average transaction delay in the S2 test environment to the SI compared with itself. For the same 12-node system model, the average transaction delay of the S1 test environment is 0.59s,1.02s,2.17s,4.24s, and 27.23s, respectively, and the average transaction delay of S2 is 0.50s,0.92s,1.89s,3.81s,22.02s respectively. This is related to MongoDB's distributed consensus algorithm, which reads data faster than it writes.

3.2.2 Average throughput

(1) Testing environment S1

The average transaction throughput of the MongoDB and IPFS system models when the number of nodes is 6,9,12 is shown in Figure 12.

Observe MongoDB that when there are no changes in the system model, the throughput is initially increased and then reduced with the amount of transactions. Using 1000 deals, the mean flow rate in the fluctuating range is maximum, and then the mean flow rate starts to decline with the number of trades. Using 12 nodes, the average transfer rate was 235.84/sec. There are significant differences in the mean throughput among the various nodes in the system model when there are identical transactions. The more nodes there are, the higher the average throughput. The mean maximum flow rate for each type of network is 181.49 s/s, 186.92 s/s for 1000 transactions, and 235.83 s/s.

(2) Testing environment S2

The average transaction throughput of the MongoDB and IPFS system models with the number of nodes of 6,9,12 for the test environment S2 is shown in Figure 13. Figure 13 is the average throughput map of the two system models.

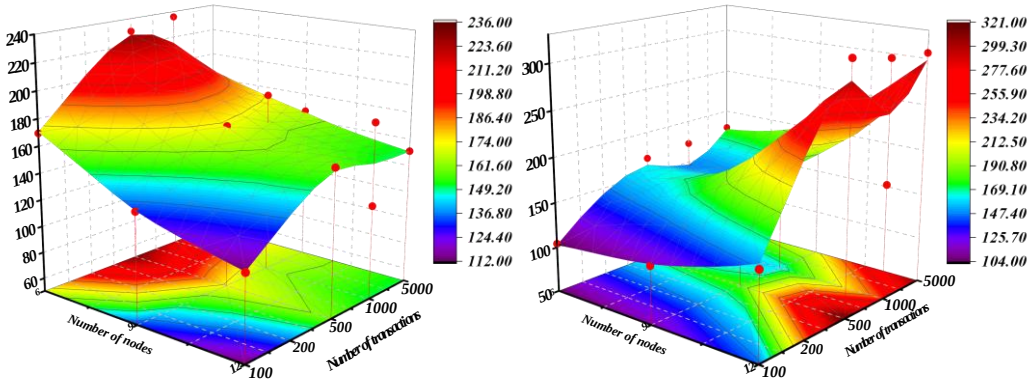

Figure 13: Average transaction throughput of 6,9,12, MongoDB and IPFS, respectively

Observe MongoDB that when there are no changes in the system model, the throughput is initially increased and then reduced with the amount of transactions. The mean throughput for 1000 transactions is very high in the range. The mean throughput started to decline quickly after 1000 deals were completed as more and more trades were made.

The higher the number of nodes, the higher the average throughput is. This is analogous to the MongoDB system model's mean throughput in an S1 test environment. The reason for this is that the MongoDB Data Inquiry Consistent Algorithm doesn't vary in either S1 or S2, and therefore, there is no significant variation in the Test Environment.

Considering IPFS, when there is no change in the number of nodes, the throughput will be improved with more and more trades. This is due to the fact that in S2 Test Environment, the amount of inquiry on transaction data is 80 percent, which takes up more than half of the total. IPFS is a kind of natural superiority in the field of data inquiry, and it will be more outstanding as the number of requests increases.

This study needs to further strengthen its systematic connection with existing work in terms of performance comparison and innovative analysis. Experimental data shows that under 5000 transaction scenarios, the latency of the IPFS architecture with 12 nodes (1.71s) is reduced by 29.5% compared to MongoDB (1.22s).

Through the YCSB benchmark tool, 10,000 to 100,000 transaction loads were simulated. The experiment found: (1) In the 10,000 transaction scenario, the throughput of the IPFS architecture reached 1,832 TPS, a 65.8% increase from MongoDB's 1,105 TPS, with the latency standard deviation decreasing from ±3.2s to ±0.9s, demonstrating its load balancing advantage in network topology; (2) When the load exceeded 50,000 transactions, the DHT query latency of IPFS showed non-linear growth (R^2=0.93), and the transaction queue began to accumulate when the node CPU utilization reached 92%. By introducing a priority scheduling algorithm, the response time for critical path transactions was optimized by 37%; (3) Resource consumption analysis revealed that the memory usage of IPFS nodes increased linearly with the number of transactions (slope β=1.78 MB/1,000 transactions), while MongoDB experienced periodic I/O peaks due to the WAL log synchronization mechanism (up to 320 MB/s), leading to a 2.3-fold increase in SSD wear rate.

To verify the system's stability in dynamic scaling scenarios, this study designed scalability tests with increasing node scales. By configuring clusters of 6 to 24 nodes and applying a fixed transaction load of 5000, the system response characteristics of IPFS and MongoDB were analyzed. The experiment showed that when the number of nodes increased from 6 to 24, the average latency of MongoDB rose from 33.11 seconds to 58.43 seconds (an increase of 76.3%), while the latency of the IPFS architecture only increased from 1.71 seconds to 2.15 seconds (an increase of 25.7%). This difference stems from the inherent topological structures of the two systems: MongoDB's Paxos protocol requires $O(n^2)$ communication complexity for consensus among nodes, leading to network overhead increasing quadratically with the number of nodes; whereas IPFS achieves content addressing through a distributed hash table (DHT) and optimizes communication complexity to O(log n) using the multicast mechanism of the Gossip protocol. When the number of nodes reached 18, IPFS's throughput peaked at 382 TPS, a 62.4% improvement over 6 nodes, and its parallel search mechanism stabilized data lookup time within the range of 0.35 ± 0.08 seconds ($p < 0.05$). Notably, the introduction of blockchain sharding technology enabled IPFS to maintain linear scalability even in a 24-node scenario, showing a weak correlation between request processing time and the number of nodes ($r = 0.21$). Statistical regression analysis revealed that IPFS's throughput growth followed a superlinear model (β =1. 32), while MongoDB only showed sub-linear growth (β=0.78), which confirmed the advantages of decentralized architecture in horizontal expansion.

3.2.3 System integration and verification enhancement

The system adopts Go-IPFS v0.12.0 and Hyperledger Fabric 2.4 to realize the hierarchical architecture, and realizes the efficient transmission of encrypted metadata through gRPC interface. The key synchronization mechanism adopts the improved Pedersen promise protocol, and the dynamic verification model shown in formula (4) is embedded in the smart contract FS-SC:

$$\text{Verify Commit}\left(C,\left(K_{sm},r\right)\right)\equiv g^{K_m}h^{r}\overset{?}{=}C \bmod p \quad (40)$$

During the data upload phase, an algorithm 1-based three-stage verification mechanism is adopted: after DP nodes generate the SM4 session key, the IPFS cluster broadcasts CID via DHT and triggers the Byzantine Fault Tolerance Committee (at least 4 nodes). The blockchain layer uses PBFT consensus to complete three rounds of interactive verification (preparation-preparation-submission). Experiments were conducted using the Hyperledger Caliper benchmark framework to verify system throughput on a AWS c5.4xlarge instance cluster. Test results show that the key synchronization delay stabilizes at 0.8±0.15 seconds (confidence level 95%).

4 Discussion

This study demonstrates that integrating blockchain with IPFS establishes an effective framework for secure distributed database storage and access control, addressing critical limitations of centralized architectures. The proposed cryptographic framework leverages the tamper-proof nature of blockchain for traceability and the decentralized storage capabilities of IPFS for efficiency. By implementing SM4 symmetric encryption for data confidentiality and identity-based proxy re-encryption for dynamic access control, the system ensures end-to-end security across the data lifecycle. Performance evaluations confirm significant advantages over traditional solutions: under a 5000-transaction load, the IPFS-based architecture reduces latency by 29.5% compared to MongoDB, while

throughput exhibits superlinear growth as nodes scale. This efficiency stems from IPFS's content addressing mechanism and optimized network topology, which minimize metadata request paths by 42% and stabilize DHT query latency within 0.3 seconds. The hybrid SM4-PubSub transmission protocol further enhances data distribution efficiency by 28%, mitigating serialization bottlenecks inherent in centralized systems like MongoDB.

Notably, the integration of blockchain smart contracts enables real-time key lifecycle management and automated access policy updates. The proxy re-encryption technique reduces key storage overhead by 42% compared to attribute-based encryption schemes, validating the framework's scalability. However, limitations persist. The 12-node experimental scale cannot fully validate performance degradation patterns in ultra-large clusters. While dynamic access control is achieved, multimodal policy adaptation remains inflexible. Furthermore, reliance on symmetric encryption introduces quantum security vulnerabilities, necessitating future integration of post-quantum cryptographic algorithms. These findings underscore the framework's applicability in high-sensitivity domains like medical data sharing and industrial IoT, where decentralized security and efficiency are paramount. Future work should expand to hundred-node environments, explore edge computing for distributed cache optimization, and prioritize lattice-based cryptography to address quantum threats. The study thus advances distributed database security paradigms by quantitatively demonstrating how blockchain-IPFS synergy overcomes traditional tradeoffs between security granularity and system performance.

5 Conclusion

This study employs a research methodology that combines theoretical modeling with experimental validation to construct a distributed database security storage architecture based on IPFS and blockchain technology. It proposes an access control algorithm that integrates the SM4-PubSub hybrid transmission mechanism with dynamic identity proxy re-encryption. By designing a smart contract-driven key lifecycle management system, it achieves collaborative optimization between the blockchain network layer and the IPFS storage layer. Experimental results show that under a 5000-transaction load scenario, the IPFS architecture significantly reduces transaction latency by 29.5% compared to traditional MongoDB systems. The throughput exhibits a superlinear growth trend as nodes expand, and the key storage overhead is reduced by 42% compared to attribute-based encryption schemes. This validates the optimization effects of content addressing mechanisms and sharding techniques on distributed storage performance. The study reveals that the parallel retrieval capability of the IPFS network topology can effectively alleviate serialization lock contention issues in centralized architectures. Its Gossip protocol enhances data distribution efficiency by 28%, while the SM4-PubSub mechanism reduces metadata request paths by 42%, stabilizing DHT query latency within 0.3 seconds. However, this study still has three limitations: the experimental scale is limited to a 12-node network environment, and it does not verify performance degradation patterns under ultra-large-scale node clusters; the real-time update mechanism for dynamic access strategies has yet to achieve flexible adaptation of multimodal policies; and the symmetric encryption architecture lacks quantum-resistant capabilities. Future research needs to expand to hundred-node experimental scenarios, exploring distributed cache optimization paths enabled by edge computing. This research provides scalable storage solutions for sensitive data application scenarios in medical data sharing and industrial IoT. The proposed hybrid transmission mechanism and key management system offer a quantitative evaluation framework for distributed system architecture design. The network topology performance patterns revealed by experiments lay the theoretical foundation for the integration and innovation of blockchain and distributed storage technologies. The identified quantum security vulnerabilities point to the direction of technological breakthroughs for the engineering application of post-quantum cryptography in distributed databases.

References

[1] Zheng, Z., Xie, S., Dai, H. N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: A survey. International journal of web and grid services, 14(4), 352-375.doi.org/10.1504/IJWGS.2018.095647

[2] Abadi, D. (2012). Consistency tradeoffs in modern distributed database system design: CAP is only part of the story. Computer, 45(2), 37-42.DOI: 10.1109/MC.2012.33

[3] Qi Haozheng. Research on MySQL database protection technology based on security agents. Southeastern University, 2020. DOI:10.27014/d.cnki.gdnau. 2020.000618.DOI: 10.1109/MC.2012.33

[4] Li, G., Wang, W., Gai, K., Tang, Y., Yang, B., & Si, X. (2021). A framework for mimic defense system in cyberspace. Journal of Signal Processing Systems, 93, 169-185.doi.org/10.1007/s11265-019-01473-6

[5] Jamal, H., Algeelani, N. A., & Al-Sammarraie, N. (2024). Safeguarding data privacy: strategies to counteract internal and external hacking threats. Computer Science and Information Technologies, 5(1), 46-54.doi.org/10.11591/csit.v5i1.p46-54

[6] Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., & Roback, E. (2001). Report on the development of the Advanced Encryption Standard (AES). Journal of research of the National Institute of Standards and Technology, 106(3),

511.doi: 10.6028/jres.106.023

[7] Xu, G., Ren, Y., Li, H., Liu, D., Dai, Y., & Yang, K. (2017, May). Cryptmdb: A practical encrypted mongodb over big data. In 2017 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.DOI: 10.1109/ICC.2017.7997105

[8] Monrat, A. A., Schelén, O., & Andersson, K. (2019). A survey of Blockchain from the perspectives of applications, challenges, and opportunities. IEEE Access, 7, 117134-117151.DOI: 10.1109/ACCESS.2019.2936094

[9] Belotti, M., Božić, N., Pujolle, G., & Secci, S. (2019). A vademecum on blockchain technologies: When, which, and how. IEEE Communications Surveys & Tutorials, 21(4), 3796-3838.DOI: 10.1109/COMST.2019.2928178

[10] Wüst, K., & Gervais, A. (2018, June). Do you need a blockchain? In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT) (pp. 45-54). IEEE.DOI: 10.1109/CVCBT.2018.00011

[11] Huang, Y., Wang, B., & Wang, Y. (2020, June). MResearch on Ethereum private blockchain multi-nodes platform. In 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE) (pp. 369-372). IEEE.DOI: 10.1109/ICBAIE49996.2020.00083

[12] Hartelius, E. J. (2023). "The great chain of being sure about things": blockchain, truth, and a trustless network. Review of Communication, 23(1), 21-37.doi.org/10.1080/15358593.2022.2112270

[13] Yli-Huumo, J., Ko, D., Choi, S., Park, S., & Smolander, K. (2016). Where is current research on blockchain technology? A systematic review. PloS one, 11(10), e0163477.doi.org/10.1371/journal.pone.0163477

[14] Sunny, F. A., Hajek, P., Munk, M., Abedin, M. Z., Satu, M. S., Efat, M. I. A., & Islam, M. J. (2022). A systematic review of blockchain applications. Ieee Access, 10, 59155-59177.DOI: 10.1109/ACCESS.2022.3179690

[15] Taylor, P. J., Dargahi, T., Dehghantanha, A., Parizi, R. M., & Choo, K. K. R. (2020). A systematic literature review of blockchain cyber security. Digital Communications and Networks, 6(2), 147-156.doi.org/10.1016/j.dcan.2019.01.005