International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18 (2026) pp. 1-12

https://doi.org/10.70917/ijcisim-2026-0036
Article

Research on Security of Distributed Storage System Based on Blockchain Technology

Wenwen Huang [1], Xiaofang Hu [2,*] and Fengfei Yang [3]

[1] Department of Electronic Information Engineering, Jining Polytechnic, Jining 272103, Shandong, China
[2] Department of Art and Design, Jining Polytechnic, Jining 272103, Shandong, China
[3] Comprehensive Support Department, Jining Polytechnic, Jining 272103, Shandong, China;
* Correspondence author: xiaowenzi2025@163.com

Abstract: The application of blockchain technology brings more possibilities for the improvement of system security. In this paper, blockchain technology is utilized to strengthen the security of distributed storage system. A multi-tier distributed storage system is constructed by designing the block structure of the transaction blockchain and realizing the two-tier architecture of data management and content storage. All-or-nothing transformation (AONT) method of corrective deletion code is introduced to realize file encryption slicing and data integrity verification to control access rights. Performance validation shows that the write latency of the distributed storage system based on blockchain technology is below 3000ms, and the read response is <30ms. Only 10 out of 1500 sets of data were successfully tampered with. The replica corruption rate is less than 0.2, and the maximum data block corruption rate is not more than 0.056. The node vulnerability expectation is in the range of [0.021,0.431] for the replica corruption rate of the system at node vulnerability expectation of 15-95, and the node load rate fluctuates less than 0.005.

Keywords: blockchain technology; distributed storage; AONT; security; file encryption slicing

1. Introduction

In recent years, along with the rapid iteration of cloud computing, big data, artificial intelligence, 5G and other technologies and the digital transformation of traditional industries, information technology has been developing rapidly, and the scale of various types of data has been growing geometrically. The development of information technology promotes the improvement of infrastructure, and the application ecology also changes continuously, which leads to the continuous growth of the data volume on the one hand, and the data types presenting the characteristics of multi-source heterogeneity on the other hand [1-3]. Based on this background, the industry is increasingly inclined to use distributed storage systems to manage and store data files [4]. Distributed storage systems have been extremely successful in terms of data storage scale, performance, price, and stability by using large-scale inexpensive machine hardware, splitting data files for storage, and improving the overall read/write throughput and other technical means [5-8]. At the same time, the rise of open cloud has led to the rapid growth of the demand for multi-user shared clustered isolated storage, and the data security of distributed storage systems has also attracted more and more attention and emphasis [9-11]. However, under this existing distributed storage system model, there are many risks and problems in data storage, and once the system is breached, it will cause great data leakage risks and losses [12-13].

As part of the data security of the application system, the data storage process is an important part of preventing data leakage. Traditional stand-alone databases have the service layer and the storage engine layer deployed in the same machine, and their security issues are mainly reflected in the fine-grained access control of data in multi-user mode [14-15]. In contrast, distributed storage systems face more severe security challenges due to their distributed nature, not only data permission control problems, but also data security problems such as data transfer between multiple components and metadata information leakage from large-scale servers [16-19]. And for certain applications involving user data, the need for data security assurance is very critical, so there is an urgent need to design and implement a distributed

storage system security mechanism to ensure data security [20-21].

In this paper, based on the blockchain non-tampering characteristics and the system file content addressing mechanism, we construct a multi-layer distributed storage system. In the data layer, dual blockchain architecture is adopted, the transaction blockchain is based on Merkle tree to ensure that the transaction is not tampered with, and the calculation result blockchain stores the historical data in linear arrays to realize efficient traceability. AONT encryption model is introduced, combined with RS correction code to build a double protection mechanism to ensure the encryption performance of node communication. Multi-layer distributed storage system architecture is designed to form a closed-loop security protection system.

2. Analysis of Blockchain-Based Distributed Storage Systems

2.1. Data Layer Architecture Design

The data layer of the designed blockchain is used to encapsulate the data of the underlying chain structure of the blockchain, which is the basis for the tampering and traceability of the blockchain data. This includes the transaction blockchain and the calculation result blockchain. Figure 1 shows the block structure of the transaction blockchain.

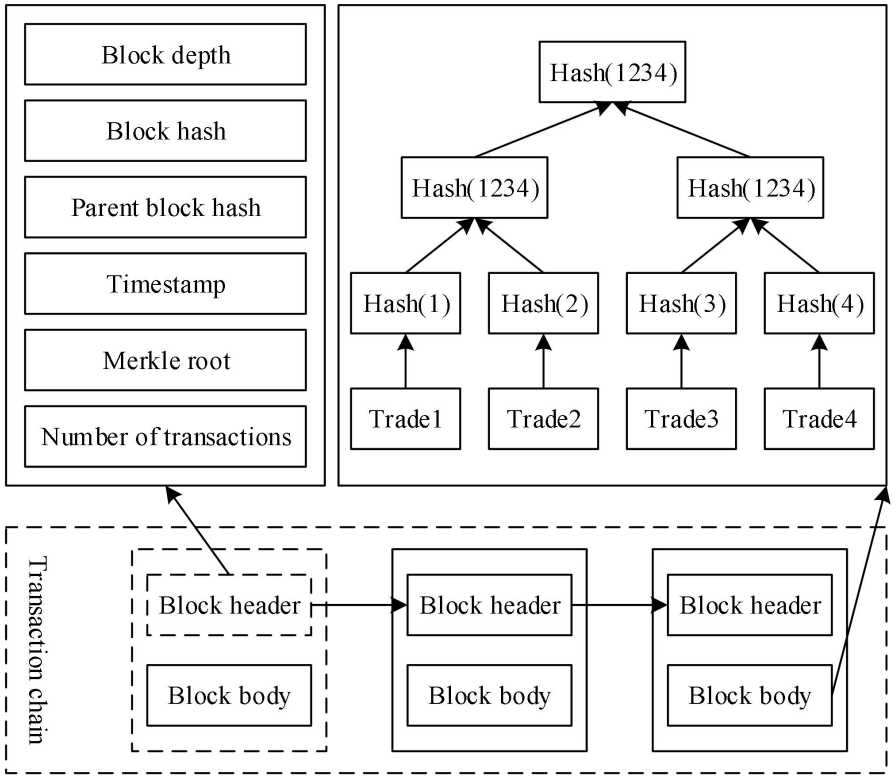

Figure 1. Structure of a transaction blockchain block.

Each transaction block contains multiple transactions in the block body, which are stored in the leaf nodes of the Merkle tree data structure. Such a data structure can verify whether a transaction is in the block in $O(\log 2n)$ time complexity, which greatly improves the efficiency of indexing and querying transactions in the block.

Each transaction contains a transaction header, multiple inputs and multiple outputs. Each input in a transaction is an output of another transaction, and these outputs contain all the available outputs of the originator. Each output contains the transaction status, transaction amount, lock script and receiver for that output.

Assuming that the transaction amount for that output is C, the receiver's public key is KU, the receiver's private key is KR, the locking script is $Lock$, and the status of the transaction is S, the locking script can be represented as:

$$Lock = \begin{cases} RSAEncrypt(C, KU) & S = 1 \\ NULL & S = 0 \end{cases} \tag{1}$$

If the locked script is $NULL$, the script has been unlocked and the corresponding output is not available. The transaction status can be represented as:

$$S = \begin{cases} 1 & RSADecrypt(Lock, KR) = C \\ 0 & RSADecrypt(Lock, KR) \neq C \end{cases} \tag{2}$$

where $RSADecrypt$ stands for RSA decryption. Each input in a transaction contains the depth of the block in which its corresponding output is located and the number of the transaction in which it is located. The depth of the block and the number of the transaction in which it is located identifies a unique output.

The transaction header contains the transaction number, initiator, and hash value. The transaction number identifies the transaction's position in the block, and the transaction numbers in each block start at 1.0 and increase sequentially. The hash value is the corresponding hash value of the transaction in the Merkle tree, assuming that the transaction n inputs, respectively, denoted as $I_1, I_2, \cdots, I_n$, m outputs, respectively, denoted as $O_1, O_2, \cdots, O_n$, inputs corresponding to outputs located in the depth of the block is D , the corresponding transaction number is N , the hash value is H , then the hash value can be expressed as:

$$H = SHA256\left(SHA256\left(\sum_{i=1}^{m} (C_{O_i} + KU_{O_i}) + \sum_{i=1}^{n} (D_{I_i} + N_{I_i}) \right) \right) \tag{3}$$

Locked scripts become $NULL$ when unlocked, so they are not involved in the hash calculation, and the transaction status, as an identifier that may change, is only used to increase the efficiency of the transaction status query, and is not involved in the hash calculation.

The block header of each transaction block contains the block depth, block hash, parent block hash, timestamp, Merkle root, and number of transactions. Assuming that the hash of the block is Hb , the block contains n transactions and the hashes are $H_1, H_2, \cdots, H_n$, the hash of the block can be expressed as:

$$Hb = SHA256\left(\sum_{i=1}^{n} (H_i) \right) \tag{4}$$

Block hashing is the key to making the blockchain tamper-proof, block hashing is calculated using SHA256 encryption, which is a one-way irreversible encryption, and any small change in the data in the block will cause the result of SHA256 encryption to change drastically.

Each block in the calculation result blockchain contains two parts: the block header and the block body. The block body of each computed result block contains multiple computed results, which are stored using a linear data structure-array because they are not used for indexing and querying in the blockchain, but only stored as historical data. Each calculation result contains a section header and multiple subtasks. The section header contains the result type, result number, algorithm number, and initiator. There are two types of result types, the first is the processing result of a distributed computing task with one or a limited number of computational results, such a task can be split into multiple subtasks and each subtask can produce a computational result, the blockchain network will record the computational process and results of such a task. The second type is the processing result of a distributed big data processing task, which usually does not need to record the result. The result type is determined based on whether the calculation result is stored in a subtask in the block. The result number identifies the location of that computation result in the block, with the result number starting at 1.0 in each block. The initiator identifies the address of the node that initiated this computation, which is also the distribution node for this computation. Each subtask in the result of the computation consists of the subtask number, the executor's public key, the executor's IP, a timestamp, and the result of the computation.

Except for the first block which has no parent block hash, the block header of each calculation result block contains the block depth, block hash, parent block hash, timestamp and result number. Block depth indicates that the block in the calculation of the result of the block chain in the number, the calculation of

the result of the first block in the block chain in the number of 1.0, the number of other blocks in increasing order; block hash is a mathematical summary of the information that exists in the block, assuming that the hash value of the block is Hc, the block contains n calculation results, and the hash value were $H_1, H_2, \cdots, H_n$, the hash value of the block can be expressed as:

$$Hc = SHA256\left(\sum_{i=1}^{n}(H_i)\right) \tag{5}$$

Assuming that the result type in the computation result is R_T, the result number is R_I, the algorithm number is R_C, the initiator is R_S, there are m sub-tasks in the computation result, each of them is numbered as I, the public key of the executor is K, the timestamp is T, and the result of the computation is R, the hash value of the result of the computation, H, can be expressed as:

$$H = SHA256\left(R_T + R_I + R_C + R_S + \sum_{i=1}^{m}(I_i + K_i + T_i + R_i)\right) \tag{6}$$

The IP address of the executor in a subtask is only used as a reference for the state of that node when the task is being computed and does not participate in the calculation of the hash value.

2.2. File Encryption and Slicing Functions

All-or-nothing transformations (AONT), which are based on transformations without secret keys, map a series of input message blocks $(x_1, x_2, \cdots x_n)$ to sequences of $(y_1, y_2, \cdots y_n)$. These sequences have the following characteristics:

1) Given all $(y_1, y_2, \cdots y_n)$ sequences, $(x_1, x_2, \cdots x_n)$ can be found.

2) If any y_i is lost, information about any x_i cannot be recovered computationally.

AONT technique for distributed storage can improve the security of the system. A general packet encryption re-storage scheme transforms a plaintext grouping $(x_1, x_2, \cdots x_n)$ into a series of ciphertext groupings $(y_1, y_2, \cdots y_n)$. In this case, the encryption and decryption of each group is independent, which means that the information corresponding to x_i can be obtained directly through one or more y_i. Moreover, the remaining all information can be calculated after getting part of the plaintext, this model is called separable encryption model. And this AONT technology encryption and packing algorithm, before getting the original text, it is necessary to complete the decryption of all the ciphertexts, that is to say, the first step is to get all the ciphertexts before decrypting them together. Contrary to the previous way, this encryption model is called non-separable encryption model.

The input message of the AONT algorithm is $(m_1, m_2, \cdots m_s)$, and subsequently the output $(m_1', m_2', \cdots m_s', m_{s+1}')$ is obtained by assigning a random key of K' as the grouping cipher, and for each plaintext grouping m_i converted into a ciphertext grouping m_i' the is computed as:

$$m_i' = m_i \oplus E(K', i) \; for \; i = 1, 2, 3, \cdots, s \tag{7}$$

where $\mathrm{E}(K', i)$ means that i is encrypted using a random key K' and E can be encrypted using AES encryption. For m_{s+1}':

$$m_{s+1}' = K' \oplus h_1 \oplus h_2 \oplus \cdots \oplus h_s \tag{8}$$

$$h_i = E(K_0, m_i' \oplus i) \; for \; i = 1, 2, \cdots, s \tag{9}$$

where K_0 is a determined public key.

The inverse of the above process is as follows:

$$K' = m'_{s+1} \oplus h_1 \oplus h_2 \ldots \oplus h_s \tag{10}$$

$$m_i = m_i' \oplus E\left(K', i\right) for i = 1, 2, \ldots s \tag{11}$$

If any segment of the message is lost the random key K' cannot be computed and the original message cannot be recovered.

Corrective Coding is a coding technique used to detect and correct errors that occur during transmission or storage. During digital communication or storage, errors are likely to occur in the data due to noise, interference or corruption, etc., and the corrective censoring technique can enhance the reliability of data transmission or storage to some extent. The Reed-Solomon (RS) corrective censoring code, which is a linear corrective censoring code, is introduced in the corrective censoring code based implementation. The basic principle of this technology is as follows: given k original data blocks and a positive integer n, RS calculates n check blocks according to the original data blocks, and finally gives $k+n$ each data to the user, and the user can also recover the initial data under the condition of obtaining at least any k data blocks, that is, the positive integer n is the upper limit of the number of data blocks allowed to be lost under the premise of recovering the complete data.

RS guarantees data fault tolerance by supplementing a Vandermonde matrix below the unit matrix. The number of rows of the Vandermonde matrix is the fault tolerance of the storage system.

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & \cdots & t \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2^{m-1} & 3^{m-1} & \cdots & t^{m-1} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_t \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_t \\ c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} \tag{12}$$

In order to realize asymmetric encryption when the node encrypts the information and sends it to the opposite node, this system references the encryption method of the open source project, utilizing the Curve25519, Salsa20 and Poly1305 algorithms. The encryption principle is as follows: use the private key of this node A and the public key of the opposite end node B to generate a shared secret key using the Curve25519 algorithm, realize symmetric encryption through the Salsa20 algorithm, and finally complete the authentication of the message using Poly1305. When the peer node B receives the message encrypted by the above encryption method, it can use the private key of this node B and the public key of the peer node A to calculate the sharedKey required for decryption and complete the decryption. Under the premise that the public key of the node is visible in the whole network and the private key of the node is kept secret, it can be guaranteed that the encrypted information can be decrypted only by the two communicating parties.

2.3. Distributed Storage System Overall Architecture

Figure 2 shows the system architecture of the distributed storage system (BCDS) system. The BCDS system as a whole is divided into: the user layer, the operation support layer, the service application layer, the data layer, the resource layer, and the operation environment.

1) The user layer of BCDS system includes web version, mobile client, and PC.

2)The operation support layer is mainly the API interface of the system, including file storage API, object storage API, and de-blockchain API.

3) The data application layer is the main business of BCDS, including file system and object storage, in addition to the file data block distribution module of distributed storage. Among them, the file system includes: distributed storage file system, including file upload service, file download service, file deletion service, file modification service, file list service, file version service, file encryption service, file sharing service.

4) The data layer is the data to be stored and counted, including file metadata database, file sharing

database, data block reference database, file system database, file data block database, and storage node database, in which the file metadata database records the metadata of all uploaded files of the user, and generates the file system database after organizing the user's directory, and the specific data blocks of the files are saved in the The file data block database, the data block reference database records the number of references to each data block, the file sharing database records the file sharing records, and the storage node data block is for IPFS node statistics.

5) The resource layer is mainly the Ethernet blockchain and the IPFS file system, in which the file metadata database, the file sharing database, the data block reference database, the file system database, and the storage node database are saved in the blockchain, and the file data block database is saved in the IPFS.

6) In the operation environment, blockchain and IPFS can run on different operating systems, including Linux, windows, Mac and even Android. In the user layer, operation support layer, service application layer, data layer, has been accompanied by the user authentication service, the user can only manipulate their own files in order to safeguard the security of the user's files.

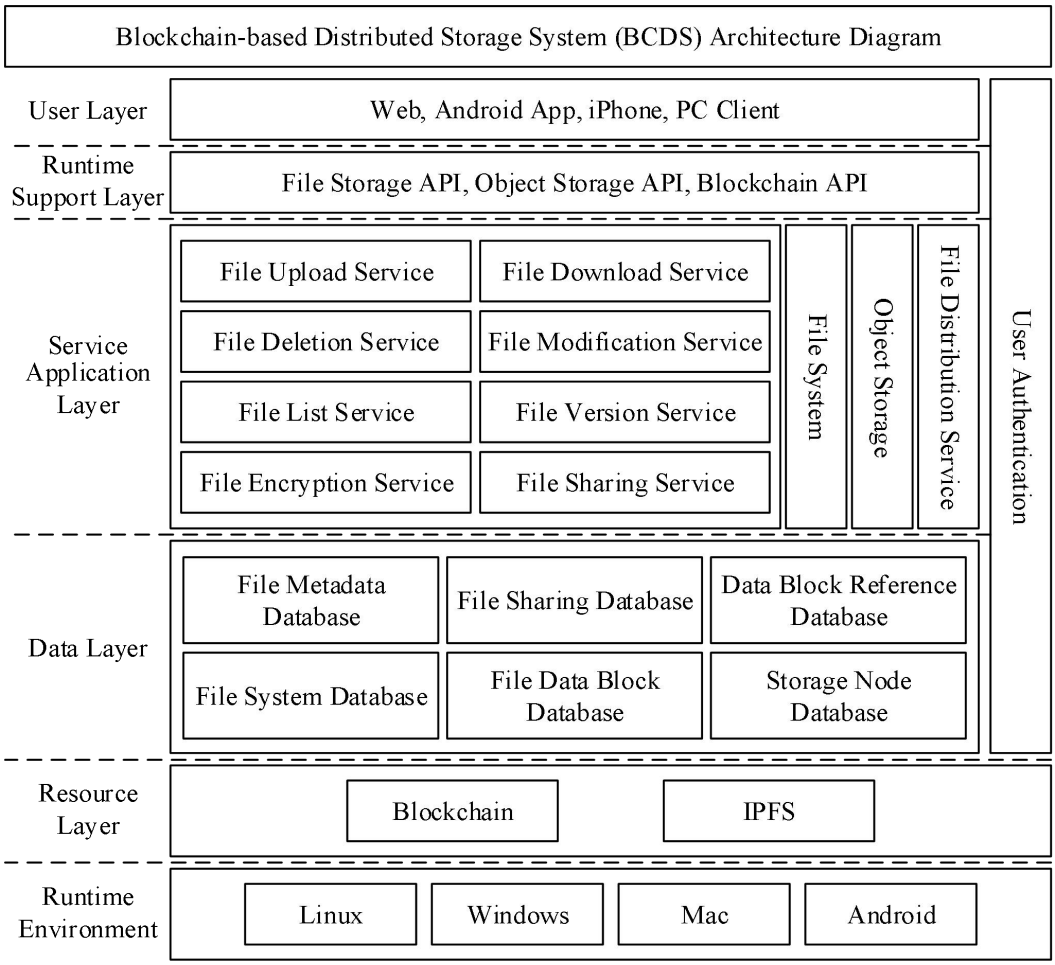

Figure 2. Architecture of the BCDS System.

3. Performance Validation of Blockchain-Based Distributed Storage Systems

3.1. System Performance Analysis

3.1.1. System Write Performance

The write performance and read performance of a distributed storage system based on blockchain technology determines whether the system can quickly realize the encryption and storage of files. Based on this, in this section, files of different sizes are uploaded to the distributed storage system to evaluate their upload latency, where the latency refers to the time between the completion of file upload and the return of response from the server. Note that the evaluation latency ignores the network transmission time of the file upload to the server, and the purpose is to test the stress on the distributed storage system

caused by files of different sizes. Figure 3 shows the system upload file size and processing latency relationship. Files with sizes between 0 and 10000 KB are uploaded to the distributed storage system, and the write latency always stays below 3000ms despite the large differences in file sizes, indicating that the system has a better write performance and is able to process files of varying sizes at the same time.

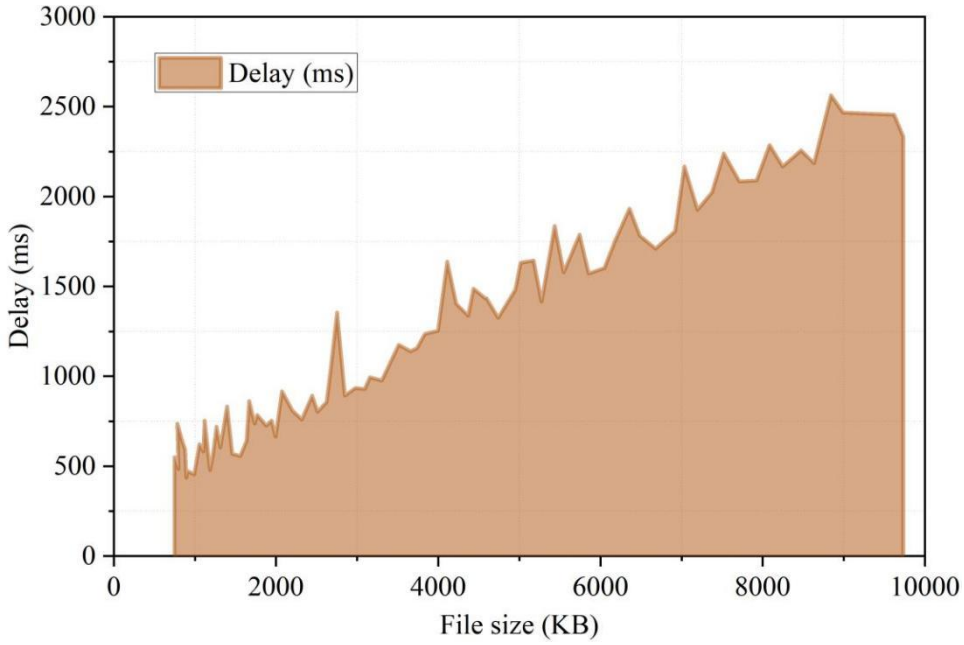

Figure 3. File size and delay uploaded by the system.

3.1.2. System Read Performance

The read latency of the distributed storage system is tested using the jmeter script, again the latency statistics is the time consumed by the server to process the download request and does not take into account the latency of the file network transfer. Distributed storage system for the first time to read the file authorized by other organizations need to find the file stored in the system, download the file piece after the local assembly into a file, and then read the file directly back to the local cache. Therefore, the server processing latency for "reading file slices" and "reading cache" is different. Figure 4 shows the test results of these two reading methods. From the test results, we can see that when reading the cache, the server processing delay is basically stable at less than 30ms, which has almost no relationship with the file size. While when reading file slices for the first time, the server processing delay increases from 30.93ms to 130.36ms with the increase of the file, which is linearly correlated. Due to the inclusion of the key, the response latency of reading the file are milliseconds, the file reading speed is very fast, and can quickly respond to the needs of the system users.

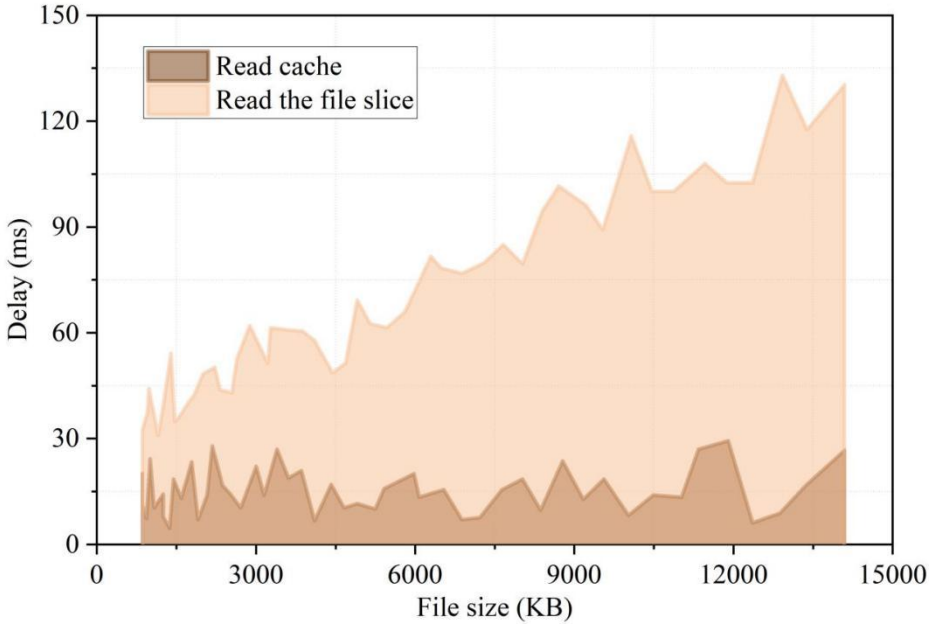

Figure 4. The delays of the two reading methods.

3.2. Comparison of Security of Different Data Storage Methods

Setting the total amount of data to be stored as 1500 groups of simulation experiments, in order to present more intuitively the data security of the distributed storage system under different storage methods, the simulated network attacker launches a network attack to the distributed storage system respectively, and the simulated network attack issued by the attacker has the following two methods of data tampering: one is to invade the storage nodes of the system to tamper with the data; and the other is to invade the transmission channel to forge the data packets. At the end of the simulated network attack, the amount of data successfully tampered with in the distributed storage system under each method is counted and organized respectively, so as to assess the security of the stored data. Table 1 shows the results of the simulation experiments. From the data in Table 1, it can be seen that the probability of stored data being successfully tampered with by network attacks under the two traditional methods (cloud computing and edge computing) increases as the amount of stored data data in the distributed storage system continues to increase. The number of successful tampering groups in cloud computing surges from 9 to 155, while the number of successful tampering groups in edge computing also increases from 10 to 120. In contrast to the blockchain technology approach of this paper, the total number of successful tampering of stored data is only 10 groups, which is much less than the comparison approach. Distributed storage system based on blockchain technology has higher data storage security than traditional methods.

Table 1. Simulation experiment results.

Data volume/group	Successfully tampered with the data/group		
	Blockchain	Cloud computing	Edge computing
150	0	9	10
300	0	15	14
450	0	19	17
600	0	27	20
750	1	45	30
900	1	65	33
1050	1	79	45
1200	2	92	72

1350	2	123	87
1500	3	155	120

3.3. Distributed Storage System Security Check

3.3.1. Impact of Data Volume on System Security

Further compare the security of distributed storage systems based on the three methods under different conditions. Under the condition of choosing default values for other parameters, the total amount of files of [15T,25T,35T,45T,55T,65T,75T,85T,95T] are selected to compare the security indexes of the three methods, and meanwhile evaluate the stability of the distributed storage system based on the three methods under the actual application scenarios. The replica corruption rate and data block corruption rate are used as the evaluation indexes. Figure 5 shows the copy bad rate under different data volumes. Figure 6 is the data block corruption rate under different data volumes. Both replica corruption rate and data block corruption rate are the lowest in blockchain-based distributed storage system. In a file data volume of 15-95T, the replica corruption rate of blockchain-based distributed storage system always stays below 0.2, while the replica corruption rate of both cloud-based and edge-based computing is above 0.3. The highest data block corruption rate of the blockchain-based distributed storage system is only 0.056, while the data block corruption rates of the cloud-based and edge-based computing are 0.146 and 0.099. Under different data volumes, the blockchain technology-based distributed storage system has less possibility of corruption and higher operational stability.

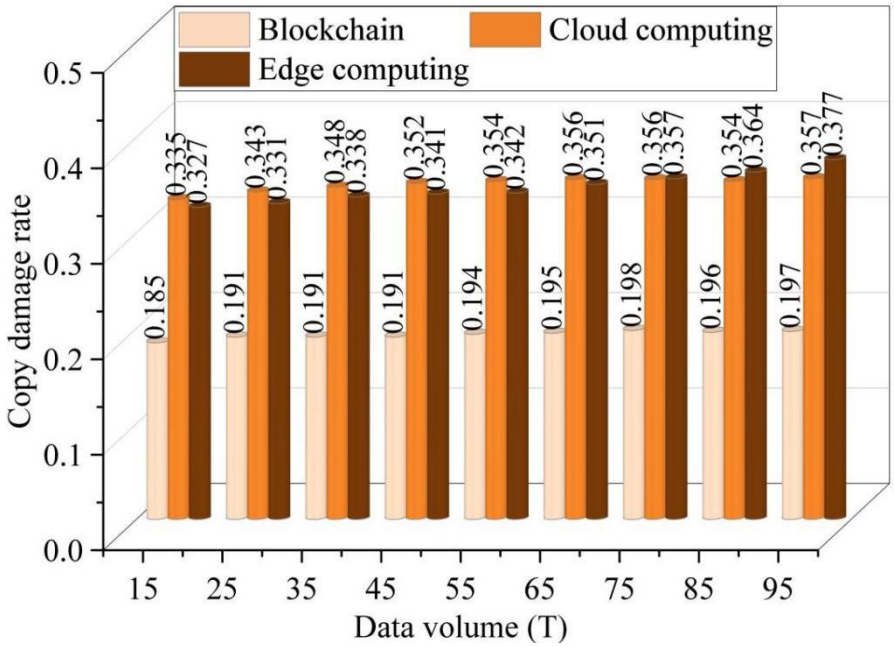

Figure 5. Corruption rate of replicas under different data volumes.

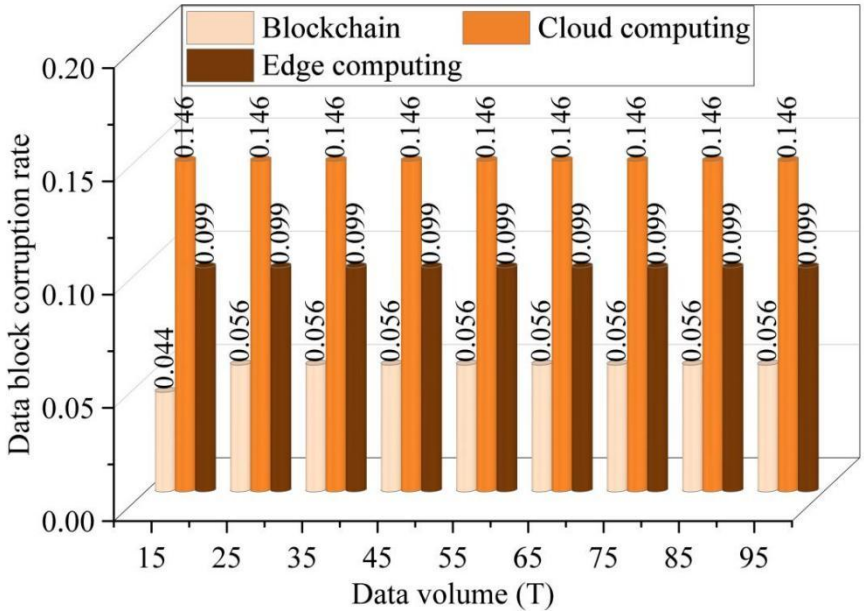

Figure 6. Corruption rate of data blocks under different data volumes.

3.3.2. Impact of Heterogeneity on System Security

A data volume of 15T is selected as the default value for heterogeneous experiments. Reasonable use of node heterogeneity to place replicas can effectively improve data security, and this section will study the impact of different node heterogeneity on data security. With the default values selected for other parameters, the vulnerability density is chosen from 5% to 15%, and the corresponding node vulnerability expectation is 15 to 95, respectively, to compare the security metrics of the three placement strategies. Fig. 7 shows the corruption rate of replicas with different vulnerability densities for the three methods. When the vulnerability density increases, the replica corruption rate increases significantly. The reason for this is that the higher the vulnerability density of the cluster, the higher the probability of common-mode vulnerabilities between nodes, without changing the attack capability. The replica corruption rate of a distributed storage system based on blockchain technology increases from 0.021 to 0.431 as the node vulnerability expectation changes from 15 to 95. While the copy corruption rate improves from 0.027 to 0.455 for cloud computing, the copy corruption rate improves from 0.039 to 0.471 for edge computing. Distributed storage systems based on blockchain technology have the lowest copy corruption rate.

3.3.3. Node Load State Analysis

Figure 8 shows the load state of the cluster after using the blockchain technology-based distributed storage system to store 150T of data, and it can be seen that the load of the majority of nodes is close to 0.33, and only a few nodes appear to fluctuate and the fluctuation range is very small, basically no more than 0.005. Compared with the other two methods, the blockchain technology-based distributed storage system can effectively ensure the load balance of the cluster, and at the same time, in the actual production environment, it can effectively avoid the uneven distribution of clients and the problem of different total amount of data written by each client. At the same time, in the actual production environment, it can effectively avoid the uneven distribution of clients and the problem of different total amount of data written by each client. The above experiments show that the distributed storage system based on blockchain technology can effectively improve the data security and at the same time can better balance the load pressure of nodes.

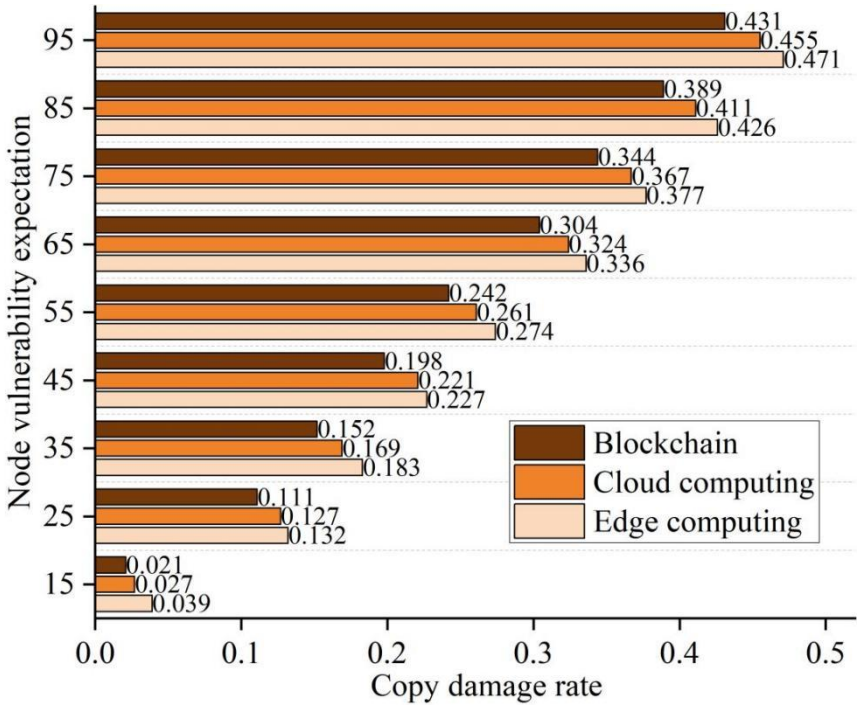

Figure 7. Corruption rate of replicas under different vulnerability densities.

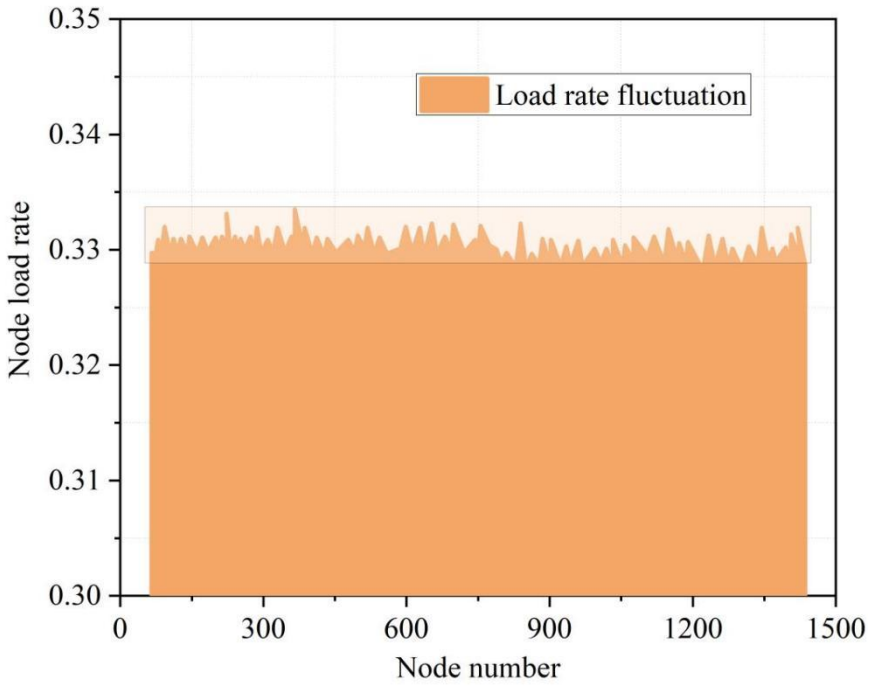

Figure 8. The system load status based on blockchain.

4. Conclusion

In this paper, we design and verify the security and stability of a blockchain-based distributed storage system. The system write latency is less than 3000ms, read cache latency is less than 30ms, and read file slice latency varies with file size. Blockchain program data tampering success is only 10 groups, much less than 2 traditional comparison methods. 15-95T of file data volume, the copy of the system based on blockchain technology damage rate <0.2, data block damage rate between 0.044-0.056. The node vulnerability expectation varies between 15-95, and the replica corruption rate increases from 0.021 to

0.431. The load rate of most nodes is around 0.33, with a floating range of no more than 0.005. In the future, lightweight protocols can be developed to further reduce the system latency, and to improve the efficiency of writes and reads while ensuring security.

References

1. Zhuo, W. (2024). Multi-source heterogeneous data storage methods for omnimedia data space. International Journal of Grid and Utility Computing, 15(3-4), 314-322.
2. Wang, Y., Shi, Q., Song, H., Li, Z., & Chen, X. (2016, August). Multi-source heterogeneous data integration technology and its development. In International Conference on Promotion of Information Technology (ICPIT 2016) (pp. 133-138). Atlantis Press.
3. Zhang, Y., Wang, Y., Ding, H., Li, Y., & Bai, Y. (2019). Deep well construction of big data platform based on multi-source heterogeneous data fusion. International Journal of Internet Manufacturing and Services, 6(4), 371-388.
4. Singh, H. J., & Bawa, S. (2018). Scalable metadata management techniques for ultra-large distributed storage systems--A systematic review. ACM Computing Surveys (CSUR), 51(4), 1-37.
5. Elyasi, M., & Mohajer, S. (2020). Cascade codes for distributed storage systems. IEEE Transactions on Information Theory, 66(12), 7490-7527.
6. István, Z., Sidler, D., & Alonso, G. (2017). Caribou: Intelligent distributed storage. Proceedings of the VLDB Endowment, 10(11), 1202-1213.
7. Ganesan, A., Alagappan, R., Arpaci-Dusseau, A. C., & Arpaci-Dusseau, R. H. (2017). Redundancy does not imply fault tolerance: Analysis of distributed storage reactions to file-system faults. ACM Transactions on Storage (TOS), 13(3), 1-33.
8. Kakoulli, E., & Herodotou, H. (2017, May). OctopusFS: A distributed file system with tiered storage management. In Proceedings of the 2017 acm international conference on management of data (pp. 65-78).
9. Ghaffari, K., & Lagzian, M. (2018). Exploring users' experiences of using personal cloud storage services: a phenomenological study. Behaviour & Information Technology, 37(3), 295-309.
10. Zhang, Y., Yu, J., Hao, R., Wang, C., & Ren, K. (2018). Enabling efficient user revocation in identity-based cloud storage auditing for shared big data. IEEE Transactions on Dependable and Secure computing, 17(3), 608-619.
11. Yang, J., He, S., Lin, Y., & Lv, Z. (2017). Multimedia cloud transmission and storage system based on internet of things. Multimedia Tools and Applications, 76, 17735-17750.
12. Yang, P., Xiong, N., & Ren, J. (2020). Data security and privacy protection for cloud storage: A survey. Ieee Access, 8, 131723-131740.
13. Dong, F., Zhou, P., Liu, Z., Shen, D., Xu, Z., & Luo, J. (2017). Towards a fast and secure design for enterprise-oriented cloud storage systems. Concurrency and Computation: Practice and Experience, 29(19), e4177.
14. Acquah, M. A., Chen, N., Pan, J. S., Yang, H. M., & Yan, B. (2020). Securing fingerprint template using blockchain and distributed storage system. Symmetry, 12(6), 951.
15. Zhang, X., Grannis, J., Baggili, I., & Beebe, N. L. (2019). Frameup: an incriminatory attack on Storj: a peer to peer blockchain enabled distributed storage system. Digital Investigation, 29, 28-42.
16. Li, H., Hu, J., Ma, H., & Huang, T. (2017, December). The architecture of distributed storage system under mimic defense theory. In 2017 IEEE International Conference on Big Data (Big Data) (pp. 2658-2663). IEEE.
17. Yang, H., Shin, W., & Lee, J. (2018). Private information retrieval for secure distributed storage systems. IEEE Transactions on Information Forensics and Security, 13(12), 2953-2964.
18. Shahid, F., Ashraf, H., Ghani, A., Ghayyur, S. A. K., Shamshirband, S., & Salwana, E. (2020). PSDS–proficient security over distributed storage: a method for data transmission in cloud. IEEE Access, 8, 118285-118298.
19. Wang, J., Chenchen, H., Xiaofeng, Y., Yongjun, R., & Sherratt, S. (2022). Distributed secure storage scheme based on sharding blockchain. Computers, Materials & Continua, 70(3), 4485-4502.
20. Darwich, M., Khalil, K., Ismail, Y., & Bayoumi, M. (2023, October). Blockchain-Enhanced Distributed Storage for Cloud-Based Video Streaming: Enhancing Security and Scalability. In Proceedings of the Future Technologies Conference (pp. 426-439). Cham: Springer Nature Switzerland.
21. Han, R., Hu, Q., Zhang, H., Ge, Y., Quan, X., & Wu, Z. (2024). Robust allocation of distributed energy storage systems considering locational frequency security. International Journal of Electrical Power & Energy Systems, 157, 109903.