Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

The key security management scheme of cloud storage based on blockchain and digital twins

Jie Huang[1,2*] and Jiangyi Yi[2]

Abstract

As a secure distributed ledger technology, blockchain has attracted widespread attention from academia and industry for its decentralization, immutability, and traceability characteristics. This paper proposes a cloud storage key security management scheme based on blockchain. To resist brute-force attacks launched by adversaries on ciphertexts, the scheme uses an oblivious pseudo-random function (OPRF) to generate randomized convergent keys and improve data confidentiality. Second, the scheme enhances the reliability of concurrent key management through a secret sharing mechanism, where convergent keys are split into key fragments and distributed on blockchain for storage. Even if a certain number of key fragments are lost or damaged, users can still recover complete key information through block transaction records. In addition, the scheme effectively supports file-level and block-level data security deduplication. Security analysis and experimental performance evaluation indicate that this scheme can ensure the security of keys and the confidentiality of data, and it has a low computational overhead for generating file-level encryption keys under this scheme. Even for a 100 MB file, the computational overhead required for generating encryption keys is less than 2 s, which improves computational efficiency.

Keywords Blockchain, Cloud storage, Key security management scheme, Digital twins

Introduction

Cloud computing [1] as a new type of computing model and service category, uses technologies such as virtualization, distributed computing, and dynamic scheduling to provide flexible demand allocation, scalable computing services, and elastic resource scheduling for enterprises and users. This effectively addresses issues such as imbalanced resource sharing and low storage efficiency, greatly improving the utilization rate of computing resources [2]. In the cloud computing service model, enterprises and users can reasonably configure resources, obtain the corresponding computing and storage resources as needed through cluster functions, distributed systems, and heterogeneous storage devices, and thus reduce their own data management costs. Cloud storage, as an important technology extension of cloud computing development [3], provides remote and efficient data storage services for enterprises and users through network services and collaboration with a large number of heterogeneous storage devices. This not only changes the traditional local data storage mode but also reduces the high data management and maintenance costs for enterprises and users. With advantages such as high efficiency, high scalability, low cost, and low energy consumption, cloud storage has become the mainstream data storage model.

The increasing volume of user data has placed a significant burden on cloud storage service providers. According to a report by the Internet Data Center

*Correspondence:
Jie Huang
huangjie918@163.com
[1] Hunan Provincial Engineering Research Center for Missile Maintenance, Changsha, Hunan 410024, China
[2] Department of Aviation Electronic Equipment Maintenance, Changsha Aeronautical Vocational and Technical College, Changsha, Hunan 410024, China

(IDC) [4], in 2016, the total global data volume reached approximately 16ZB, and by 2025, it is estimated that the global data volume will reach about ten times that of 2016, reaching 163ZB. In the face of such a massive amount of data, cloud storage service providers typically employ data deduplication techniques to detect redundant data and optimize data storage [5, 6]. To reduce storage costs, deduplication technology has been widely adopted by cloud storage service providers and has become an indispensable data optimization technology on cloud servers [3]. However, users will apply secure encryption algorithms to their data to ensure data security. Implementing traditional encryption algorithms will produce randomized ciphertexts, which will cause the deduplication technology to lose effectiveness [7]. Converging encryption algorithm has been proposed to solve the compatibility issue between encryption algorithms and deduplication technology. In this algorithm, the keys required for encryption (also known as converging keys) are derived from the data content itself, ensuring the consistency of the data plaintext and ciphertext [8], i.e., the same plaintext data can generate the same ciphertext. Therefore, converging encryption algorithms [9] have been widely used in cloud storage systems. However, securing converging keys in assembling encryption algorithms is always a considerable challenge. If the converging key is stored locally by the User, although it can ensure the security of the key, as the volume of user data increases, the number of converging keys will also present a geometric growth trend, which will be a considerable storage burden for the User. Most cloud storage systems that support vital management will choose to introduce a trusted third party to assist users in managing their keys to share the computation and communication overhead on the user side. However, raising the third-party management server will make the system vulnerable to collusion attacks launched by attackers, resulting in the leakage of the original keys and ciphertexts. As early as 2018, over 50 million user profiles on Facebook were leaked, obtained and utilized by another data analysis company without the users' knowledge. In the same year, the MyFitnessPal platform, owned by the famous American sports brand Under Armour, experienced a data breach affecting 150 million users, including leaked information such as user account passwords and home addresses. In 2021, a large-scale data breach occurred on Twitch, a live streaming platform owned by Amazon, where at least 100 GB of platform data was stolen by hackers. Therefore, how to resist collusion attacks and ensure the confidentiality and security of convergent keys remains a significant research topic in cloud security storage systems [10, 11].

In recent years, blockchain technology has attracted widespread attention from the academic community, industry, and government institutions. Its decentralization, immutable data, and traceability characteristics have become the focus of many researchers. In a blockchain network, all participants can complete the processes of generating, transmitting, packaging, and verifying transaction information without relying on trusted third-party authorities or management centers. Once the transaction information is successfully added to the chain, participants cannot tamper with or forge the transaction information on the block. Therefore, blockchain provides users with security guarantees such as immutability, transparency, and traceability for their transaction information, which has led to the widespread application of blockchain technology in various fields, including the education industry, financial sector, industrial manufacturing, supply chain, medical system, and intelligent transportation. Similarly, the decentralization characteristics of blockchain will also provide new solutions and directions for the security issues in the cloud mentioned above storage systems.

In response to the issues mentioned above, this paper proposes a cloud storage key security management scheme based on blockchain, which mainly contributes as follows:

- A cloud storage key security management scheme based on blockchain is proposed, which ensures the security of encryption keys without introducing a trusted third-party key management server. In this scheme, the user's critical information is uploaded to the blockchain as block transactions, ensuring the vital information cannot be maliciously tampered with.
- A distributed critical management method is constructed using a secret sharing mechanism to improve the reliability of key management. In this scheme, the encryption key is cut into key fragments and distributed to the blockchain for storage. The original key can be recovered only when a pre-set threshold number of critical pieces is obtained.
- Different secret factors are introduced in file-level secure storage and block-level secure storage to enhance the secrecy of the convergent encryption algorithm. The adversary cannot obtain the original data through brute-force attacks even if the user data belongs to a predictable dataset. Security analysis shows that this scheme can ensure data confidentiality and critical security.

This article is mainly divided into six parts, conducting in-depth research on the cloud storage key security

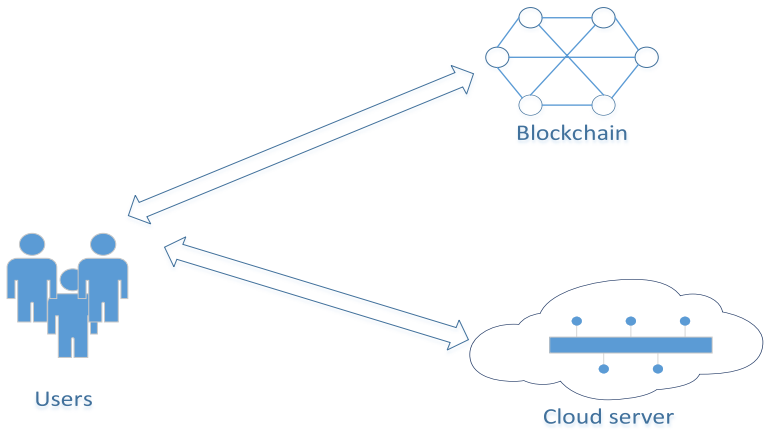

Fig. 1 System model

management scheme. The specific arrangement is as follows:

The first section is the introduction, which describes the research background of cloud storage and analyzes the shortcomings and defects of existing cloud storage key schemes.
The second section is the framework of the scheme, introducing the blockchain system model, explaining its threat model, and providing the system's security objectives.
The third section is the implementation of the system scheme, proposing a blockchain-based cloud storage key security management scheme to achieve key security management without a trusted third party and ensure the security of the encryption keys.
The fourth section on security analysis and the fifth section on experimental analysis show that this scheme has a low computational overhead in terms of time.
Finally, there is a summary.

Scheme framework

System model

As shown in Fig. 1, we will introduce the system model of the proposed scheme, which mainly includes the User, cloud storage provider (CSP), and blockchain (BC).

- User entity. Due to limited storage space, users choose to outsource their data to cloud storage service providers to save local storage space. To ensure data security, users need to perform secure encryption on the data and generate ciphertext, and finally, entrust the cloud storage service provider to store the ciphertext on the cloud server [12].
- Cloud storage provider. Cloud storage service providers can offer users sufficient storage space. In a cloud storage system that supports data deduplication optimization, cloud service providers must perform duplicate detection on user data and then allocate appropriate storage space to eliminate redundant data [13].
- Blockchain. Blockchain is a distributed, innovative database technology that has garnered widespread public attention since its proposal in 2008. Bitcoin and Ethereum, applications designed with blockchain as the underlying technology, have thrived and gained increasing recognition and importance among researchers. The development of blockchain has gone through three stages so far: blockchain 1.0, blockchain 2.0, and blockchain 3.0, with each stage seeing a broader scope of applications compared to the previous one. The security issues inherent in blockchain systems have also become a focus of many researchers. Malicious attackers can still exploit blockchain protocols to devise targeted attacks, such as Sybil attacks [14–16], dust attacks, double-spend attacks [17, 18], DDoS attacks [19], and mining attacks [20–23], among others. These can disrupt blockchain systems, preventing them from functioning normally, or even causing them to collapse altogether. The foundational architecture of blockchain generally consists of a six-layer model, from top to bottom: the application layer, the contract layer, the incentive layer, the consensus layer, the network layer, and the data layer. Each layer is responsible for specific functions, with the layers working together to construct a decentralized, distributed ledger system.

Blockchain, as an open and decentralized distributed management system, comprises various technologies

such as cryptography, distributed network architecture, and consensus algorithm [24], which can provide high-security guarantees for users' transaction information [25]. In the proposed solution of this paper, we will use the Ethereum blockchain to achieve secure and reliable convergence key management, and design a cloud storage key security management scheme.

Threat model

Based on the above system model, we will elaborate on the existing threat model. For the blockchain system model, we will provide a detailed explanation of the threat model, mainly for the following reasons:

Ensuring security: By analyzing the threat model, we can better understand and predict the various factors that may pose threats to the blockchain system, thus taking corresponding security measures to prevent these threats and ensure the security of the system.
Preventing potential attacks: The threat model can help us identify potential security vulnerabilities and weaknesses in the blockchain system, so that we can take measures to repair and strengthen them before attackers exploit these vulnerabilities, preventing potential security attacks.
Improving system robustness: The analysis of the threat model can help us enhance the robustness of the blockchain system, enabling it to better cope with and resist various attacks when facing external and internal adversaries, and maintain stable system operation.
Guiding security strategy formulation: By analyzing the threat model, we can better understand the security risks faced by the blockchain system, thus providing strong support for formulating targeted security strategies and measures el, which mainly includes two kinds of adversaries: external adversaries and internal adversaries.

This model mainly involves two types of adversaries: external adversaries and internal adversaries.

- External adversaries. External adversaries can be considered unauthorized users, lacking legitimate identity or permission information, and cannot directly interact with the system. External adversaries will attempt to collude with entities within the system, such as cloud storage service providers, to obtain information about user data, which will pose a considerable threat to the proposed approach.
- Internal antagonists. Internal adversaries can be considered as entities within the system who have honest and curious characteristics; on the one hand, they will honestly follow the agreements stipulated by the system; on the other hand, they will try to obtain information about user data and expose as much user data information as possible to external adversaries for profit. Generally speaking, internal adversaries have a greater opportunity to access user data than external adversaries, and their threat to user data security is higher.

Security objectives

Based on the above security threats, this article will propose a cloud storage key security management scheme based on blockchain, which needs to meet the following security objectives:

- Data confidentiality. The proposed scheme must ensure that external adversaries cannot obtain any information about user data, even if the user data belongs to a predictable data dictionary. In addition, the proposed scheme must consider a more practical issue: external adversaries collude with each other to steal user data, with the most common case being cloud storage service providers colluding with external adversaries and exposing the stored data on the cloud server to external adversaries. Therefore, the proposed scheme must ensure the confidentiality of data on the cloud storage server.
- Security of encryption keys. In the proposed scheme, blockchain technology provides secure and reliable key management, where critical information is integrated into a block transaction and uploaded to the blockchain via NBM(Node Business Management). However, the transaction information on the blockchain is publicly transparent, and adversaries may obtain essential details directly. Therefore, the proposed scheme must ensure that only valid users can access the transaction information on the blockchain. In addition, the proposed method still needs to address the single point of failure issue existing in key management.

Scheme implementation

Basic notations

The symbols in this paper are defined as shown in Table 1 below.

System settings

- Set the ID of the user U to ID_U and file ID to ID_F. The ID of CSP is ID_S. The corresponding Ethereum account is EA_U. The identity identifier of CSP is

Table 1 Basic notations

Symbolic definition	Description
F	File
Bi	The i-th database
ID	Identity
n	The number of database
C	Ciphertext
T	File label
Tk	Current time
σ	Point
MAC	Message authentication code
Pay_{stor}	Storage service fees
Pay_{comp}	Compensation fees
PoW	Proof of work

ID_{CSP}, and its corresponding Ethereum account is EA_U.

- Take safety parameter 1^λ as input parameter and P as output system parameter. In addition H, H_1 and H_2 as three safe hash functions, $H, H_1 : \{0,1\} \rightarrow Z_N, H_2 : \{0,1\}^k \leftarrow Z_N$.
- The user U can, through formula $ed \equiv 1 mod \Phi(N)$ to calculate the parameters N, d, in this formula, e is RSA index ($N < e$), and modulus N is the product of two different prime numbers. In this case, the public key is $pk = (N, e)$, and the private key is $sk = (N, e)$.
- In traditional cloud storage systems, when user data is successfully stored on cloud servers, users need to pay storage fees to CSP. In addition, when user data is lost or damaged, CSP needs to compensate users for a certain loss. Specifically, we represent storage fees and compensation fees as Pay_{stor} and Pay_{comp}.

Blockchain node management

A blockchain node is one of the main components of the blockchain network systems, which can receive, transmit, and verify transaction information in a peer-to-peer network. To facilitate the description of the interaction process between the blockchain system and the proposed system model, we introduce the concept of node business management (NBM) in this scheme, which serves as the execution agent for blockchain nodes. NBM has a specific storage space and computing power, and no one can access the storage space. In addition, NBM can execute smart contracts to perform calculations and output calculation results, and it can assign specific blockchain nodes to each user to assist them in creating and uploading blockchain transactions.

File-level secure storage

This section will elaborate on the data security storage scheme at the file level. To resist violent attacks from adversaries, the proposed method adopts OPRF protocol to generate encryption keys, in which OPRF protocol can be implemented based on RSA blind signature [26, 27]. The specific file-uploading process is shown in Fig. 2.

Step-1(file-level repeatability detection): In order to upload file F, user U calculates the label:

$$T_F = TagGen(F) \tag{1}$$

User U uploads file label T_F to the cloud storage server. Then, the cloud storage server will perform file repetition detection according to the file label T_F. If the file F has already been stored in the cloud storage server, the server will return the file pointer $\sigma(F)$ to the user U, and the server does not need to upload the file F again. If file F has not been stored in the cloud storage server, the user U needs to encrypt file F to ensure data security. The details are in Step 2.

Step 2(File encryption): User U needs to convert the encryption key K_F. Specifically, user U computes

$$h' = H_I(F) \tag{2}$$

User U sends h' to NBM.

NBM can select a random number $r \leftarrow Z_N$, set up a database inside the node, record r and h', and finally send r to user U.

User U calculates blind information x

$$x = h' \cdot r^e \ mod \ N \tag{3}$$

User U sends x to NBM. NBM signs by blind message x.

$$y = x^d \ mod \ N \tag{4}$$

NBM sends the signature y to U. User U can compute:

$$z = y \cdot r^{-1} \ mod \ N \tag{5}$$

User U can verify the correctness of the formula:

$$h' = z^e \ mod \ N \tag{6}$$

Only if this formula is true, U can calculate the encryption key K_F:

$$K_F = KeyGen(F, P, H_I(z)) \tag{7}$$

Finally, U encrypts files through a symmetric encryption algorithm and obtains ciphertext C_F:

$$C_F = Encrypt(K_F, F) \tag{8}$$

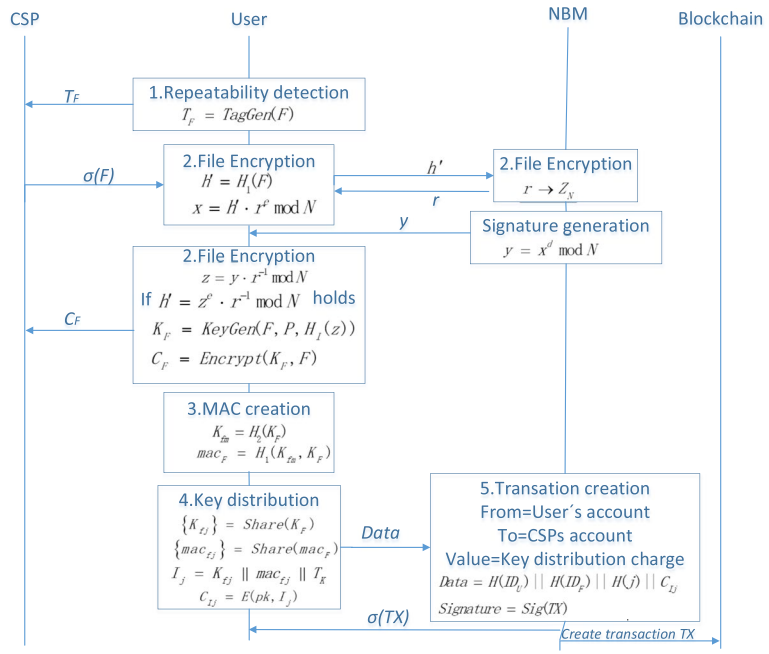

Fig. 2 File upload

Step-3(MAC generation): To verify the correctness of the encryption key, U can calculate the message authentication code (mac_F) of the encryption key K_F:

$$K_{fm} = H_2(K_F) \tag{9}$$

$$mac_F = H_1\left(K_{fm}, K_F\right) \tag{10}$$

In addition, U performs a deterministic (n, k, r)-RSSS secret sharing scheme and shits MAC through the share algorithm of the partition function, that is:

$$mac_{fj} = Share(mac_F) \tag{11}$$

Step 4(Key distribution): U can perform the Share algorithm of the partitioning function to divide the key KF, that is:

$$K_{fj} = Share(K_F) \tag{12}$$

U Compute:

$$I_j = K_{fj} \| mac_{fj} \| T_K \tag{13}$$

T_K is the current time information. In the end, U uses the public key to encrypt I_j and get the ciphertext:

$$C_{Ij} = E\left(pk, I_j\right) \tag{14}$$

$E(.)$ is an asymmetric encryption algorithm.

Step 5(Transaction generation): To generate a valid critical transaction, U calculates:

$$Data = H(ID_U) \| H(ID_F) \| H\left(j\right) \| C_j \tag{15}$$

User U sends the $Data$ to the NBM, TX NBM will create a transaction record, including TX has a value that is: TX = From||To||Value||Data||Sig(TX), the value of the parameter From is the account of U, the value parameter To is CSP account, And the value of number value is the cost of the key distribution, the value of the number of references $Data$ is $H(ID_U) \| H(ID_F) \| H\left(j\right) \| C_j$ and the Signature is Sig (TX). Eventually, NBM will deal with the TX chain and send the pointer σ(TX) to the User U. The complete transaction information is shown in Fig. 3.

Step 6(File download): In order to download file F, user U needs to reconstruct the recovery key K_F and verify the correctness of the critical K_F. Specifically, user U can obtain file F through the following steps:

- User U gets $Data = H(ID_U) \| H(ID_F) \| H\left(j\right) \| C_j$ from the blockchain via NBM.
- User U calculates $H(ID_U) \| H(ID_F) \| H\left(j\right) \| C_j$ to parse ciphertext C_{Ij} from $Data$ and uses the private key sk to decrypt the plaintext $I_j = K_{fj} \| mac_{fj} \| T_K$.
- User U verifies the correctness of T_K. If that is right, I_j is resolved from $I_j = K_{fj} \| mac_{fj} \| T_K$。
- User U can repeat the preceding steps. When k copies of K_{fj} and mac_{fj} are obtained, user U can reconstruct and recover the encryption keys K_F and MACmac_F, that is:

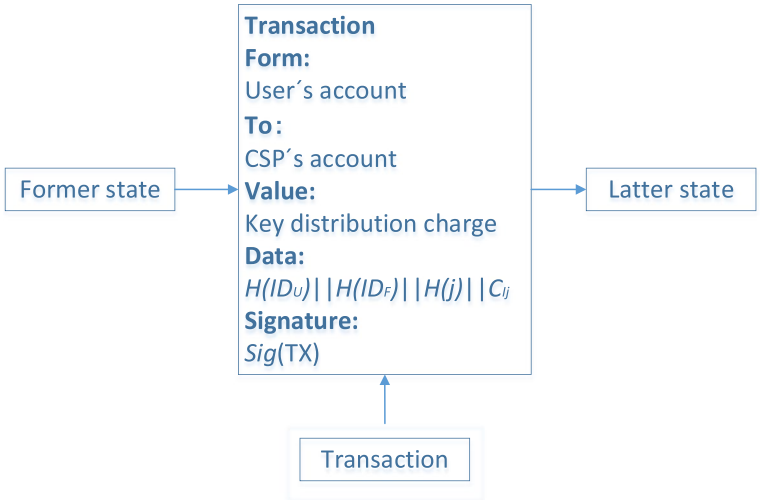

Fig. 3 Transaction information

$$K_F = Recover\left(\left\{K_{fj}\right\}\right) \tag{16}$$

$$mac_F = Recover\left(\left\{mac_{fj}\right\}\right) \tag{17}$$

- User U verifies the correctness the MACmac_F. If that is right, user U can download C_F ciphertext from CSP and obtain file F using the key F_K, $F = Decrypt(K_F, C_F)$.

Block-level secure storage

In this section, we will elaborate on the block-level secure storage scheme. The users must first perform the file-level secure storage scheme in this scheme. Cloud storage, on the other hand, stores data on cloud servers, providing flexible and scalable storage services. Block-level storage involves dividing data into multiple blocks and storing them on different physical devices, which can improve data read and write speeds and storage efficiency. When the repetitive detection result shows that the file is not repeated, the user needs first to perform block technology on the file and then perform the block-level secure storage scheme. The specific uploading process of data blocks is shown in Fig. 4.

Step-1(block-level repeatability detection) User U divides file F into n data blocks as:

$$F = B_i(1 \le i \le n) \tag{18}$$

The ID of data block Bi is ID_{Bi}, and user U calculates that the label of data block is:

$$T_{B_i} = TagGen(B_i) \tag{19}$$

User U uploads the data block tag T_{Bi} to the cloud storage server. The cloud storage server executes file duplication detection based on the data block tag T_{Bi}. If the data block Bi is stored in the cloud storage server, it returns the file pointer σ(Bi) to the user U; If data block Bi is not stored in the cloud storage server, user U will execute Step-2.

Step-2(data block encryption): User U treats the data block Bi as a leaf node and calculates the value of the root node by constructing a Merkle hash tree:

$$\tau = Mtr(\{B_i\}) \tag{20}$$

User U calculates the data block encryption key:

$$K_{B_i} = KeyGen(B_i, P, H_I(\tau)) \tag{21}$$

User U uses key K_{Bi} to get ciphertext C_{Bi}:

$$C_{B_i} = Encrypt\left(K_{B_i}, B_i\right) \tag{22}$$

User U uploads the ciphertext C_{Bi} to the CSP:

Step-3(MAC generation): User U can calculate the message authentication code mac_{Bi} of the encryption key K_{Bi}:

$$K_{B_m} = H_2\left(K_{B_i}\right) \tag{23}$$

$$mac_{B_i} = H_I\left(K_{bm}, K_{B_i}\right) \tag{24}$$

User U executes the deterministic (n, k, r)-RSSS secret sharing scheme and shits MAC through the segmentation function Share algorithm, that is:

$$\left\{mac_{b_{ij}}\right\} = Share\left(mac_{B_i}\right) \tag{25}$$

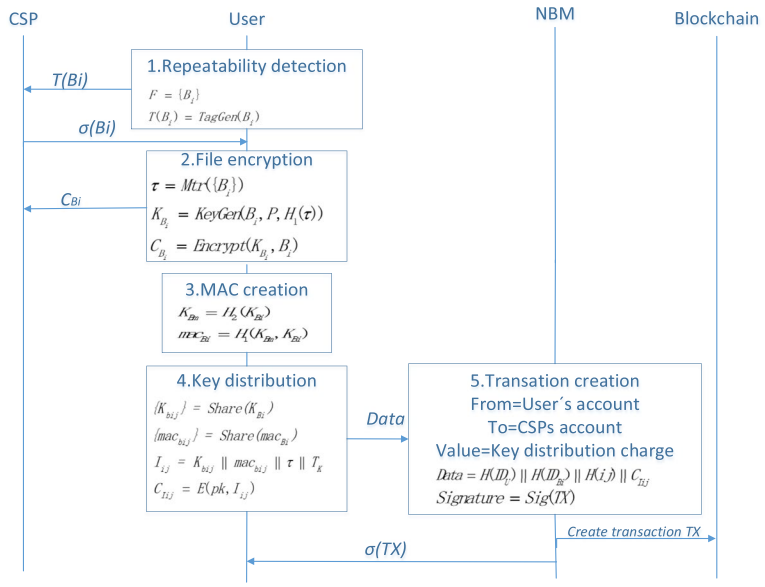

Fig. 4 Block upload

Step-4(Key distribution): User U performs the sharing function Share algorithm to slice the data block key K_{Bi}, that is:

$$\left\{K_{b_{ij}}\right\} = Share\left(K_{B_i}\right) \tag{26}$$

User U compute:

$$I_{ij} = K_{bij} \| mac_{bij} \| \tau \| T_K \tag{27}$$

T_K indicates the current time. User U uses the public key pk to encrypt I_j and get the ciphertext C_{Iij}:

$$C_{Iij} = E\left(pk, I_{ij}\right) \tag{28}$$

In it, $E(.)$ is an asymmetric encryption algorithm.

Step-5(transaction generation): User U calculates the parameter $Data$ as:

$$Data = H(ID_U) \| H(ID_{Bi}) \| H\left(ij\right) \| C_{Iij} \tag{29}$$

User U the parameter $Data$ sent To the NBM, NBM through the parameters of the received $Data$ To create a deal TX = From||To||Value||Data||Sig(TX), The parameter From is the Ethereum account of user U, the parameter To is the Ethernet account of CSP, and the parameter Value is the key distribution fee. The value of the $Data$ is $H(ID_U) \| H(ID_{Bi}) \| H\left(ij\right) \| C_{Iij}$, and the value of Signature is Sig (TX). The complete transaction information is shown in Fig. 5.

Sept-6 (Fair Payment): After the data is uploaded to the cloud server, users need to pay for the storage service provided by the CSP. As shown in Fig. 6, user U can submit a smart contract to the blockchain, which will be automatically activated and executed. If formula Γ_1 holds true, it means that the data block $\{B_{ij}\}$ has been completely stored on the cloud server, and at this time, user U needs to pay the corresponding storage service fees to the CSP. Otherwise, if formula Γ_1 does not hold true, it means that the data block $\{B_{ij}\}$ has been damaged or lost, and the CSP needs to assume corresponding responsibility and provide certain compensation for the damaged or lost data.

$$\Gamma_1 : e(L,g) = e((\prod\prod H(W_i)^{Z_i}.u^{\mu}), y) \tag{30}$$

Step-7(File downloads): To restore file F, user U can perform the following steps:

- User U sends pointer $\sigma(TX)$ to NBM and gets $Data = H(ID_U) \| H(ID_{Bi}) \| H\left(ij\right) \| C_{Iij}$
- User U calculates $H(ID_U) \| H(ID_{Bi}) \| H\left(ij\right)$ and parses ciphertext C_{Iij} from Data.
- User U uses the private key sk to get the plaintext $I_{ij} = K_{bij} \| mac_{bij} \| \tau \| T_K$.
- User U verifies the correctness of T_K. If correct, $K_{bij} \| mac_{bij} \| \tau$ is resolved from I_j.
- Repeat the preceding steps for user U. When k copies of K_{bij} and mac_{bij} are obtained, user U can recover the keys K_{Bi} and MACmac_{Bi}, that is:$K_{Bi} = Recover\left(\left\{K_{bij}\right\}\right)$,$mac_{bi} = Recover\left(\left\{mac_{bij}\right\}\right)$.

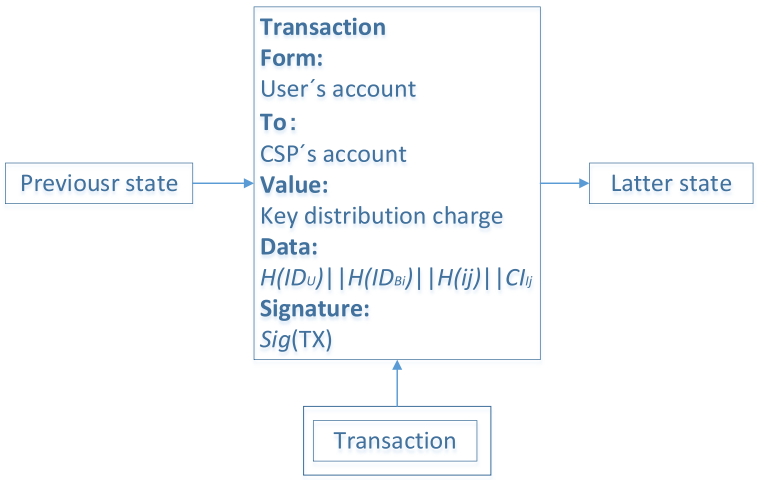

Fig. 5 Transaction information

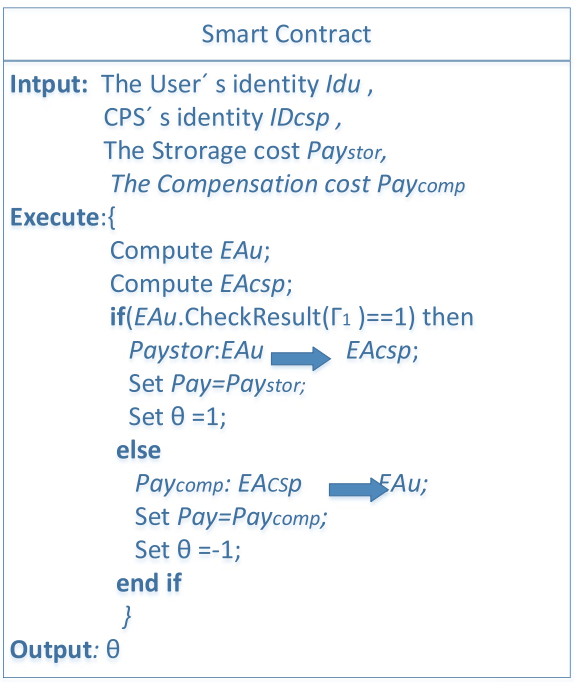

Fig. 6 Smart contract

- User U verifies the correctness of MAC mac_{Bi} and root node τ. If correct, user U can get the data block $B_i = Decrypt(K_{Bi}, C_{Bi})$, and the file: $F = B_i$.

Security analysis

In this section, we will analyze the proposed scheme's security from data confidentiality and critical security. The specific analysis is as follows.

Confidentiality of data

Theorem 1: Even if Data F does not have a sufficiently sizeable minimum entropy, the proposed scheme can resist the violent attacks launched by the enemy against the ciphertext.

Proof: In this scheme, user data must be uploaded and stored on the cloud server through an encryption algorithm. The adversary cannot obtain information about the original plaintext from the ciphertext itself. Thus, an adversary may infer the original plaintext F by brute-force attacks from the known ciphertext information and the predicted file dictionary. The scheme proposed in this paper can realize data confidentiality protection under different fine-grained de-duplication, in which it is assumed that the enemy is z and the challenger is x.

1) To remove the duplication of the file storage, challenger x establishes a system protocol to encrypt file F. Specifically, OPRF is used to generate random secret information z for challenger x, and the process is as follows:

$$h' = H_I(F) \tag{31}$$

$$x = h' \oplus \gamma^e mod\ N \tag{32}$$

$$y = x^d(F) \tag{33}$$

$$z = y \cdot r^{-1}\ mod\ N \tag{34}$$

$$K_F = KeyGen(F, P, H_I(z)) \tag{35}$$

From the above, if the enemy z wants to obtain the original file F by brute force attack, it must first know the variables P and z, in which variable P is the system parameter, and it is assumed that there is no confidentiality in this scheme. According to formulas 31, 32, 33, and 34, it can be seen that enemy z can successfully pass the system challenge established by challenger x, and it needs to guess the value of parameter r. However, since parameter r is generated randomly by challenger x, the probability of enemy z guessing parameter r's value is almost negligible. Enemy z cannot obtain the original file F by brute force attack.

2) To remove the duplication of block-level storage, opponent z successfully passed the system challenge established by challenger x, and it is necessary to guess the variables P and τ. τ is generated by constructing a Merkle hash tree and is used as a secret factor to calculate the data block encryption key K_{Bi}. Therefore, enemy z cannot obtain the original file F by brute force attack if he cannot obtain all the data blocks $\{Bi\}$.

Security of the key

Theorem 2: The encryption key distribution and recovery process is secure in this scheme.

Proof: According to theorem 1, the generation of the key has certain confidentiality, and enemy z cannot obtain the key through brute force attacks to recover the original data. In this scheme, a malicious CSP may act as an internal adversary to collude with an external adversary to obtain the encryption key (CK). To ensure the security of CK, this scheme adopts two methods to resist such collusive attacks. 1) Dividing CK into key fragments by a secret sharing mechanism can avoid single-point failure and provide security and reliability for CK. Even if any r of n key fragments are damaged or replaced by the enemy, the original key CK can be recovered through the remaining key fragments. 2) CK information will be integrated into a transaction in this scheme and uploaded to the blockchain. Due to the decentralized characteristics of blockchain, if opponent z does not have more than 51% of the total network computing power, it cannot modify CK. Therefore, CK information stored on the blockchain is secure. In addition, CK information is protected by the private key sk held by the User, which can make CK information on the blockchain to anyone (except the User who has the secret key sk) unordered ciphertext, which will ensure that enemy Z cannot access any CK information from the blockchain.

Security of data

Theorem 3: If H is a collision-resistant hash function, under the assumption of the CDH problem being difficult, the proposed scheme can resist forgery attacks initiated by the adversary X and ensure data integrity.

Proof: Assume there exists an adversary X and a challenger ϵ. Furthermore, g is the generator of the group G, v and u are elements in the CDH problem, which v and u satisfy the equation $v=g^x$ and $u=g^y$.

Initialization: The challenger ϵ sends the system's public information, such as the generator g of the group G, the public key spk and pk, to the adversary X.

Hash query: The adversary X sends a hash query to the challenger, who obtains the hash function H and returns it to the adversary X.

Signature query: The adversary X forges an invalid aggregate signature L^* according to the system protocol and passes the verification of the challenger ϵ. The adversary X can calculate:

$$\begin{aligned} e\left({}^{L^*}\!/_{L}, g\right) &= e\left(\prod_{i\in I} {}^{\sigma_i^{*Z_i}}\!/_{\sigma_i^{Z_i}}, g\right) \\ &= e\left(\prod_{i\in I}(H(W_i)\cdot u^{C_{Bi}^*})^x, g\right) / e\left(\prod_{i\in I}(H(W_i)\cdot u^{C_{Bi}})^x, g\right) \\ &= e\left(\prod_{i\in I}(H(W_i)\cdot u^{\Delta C_{Bi}}), g\right) = e\left(g^{\sum_{i\in I} xy\Delta C_{Bi}}, g\right) \end{aligned} \tag{36}$$

Therefore,

$$g^{xy} = ({}^{L^*}\!/_{L})^{-(\sum_{i\in I}\Delta C_{Bi})} = ({}^{L^*}\!/_{L}) \cdot \frac{1}{\sum_{i\in I}\Delta C_{Bi}} \tag{37}$$

At this point, the adversary X has broken the CDH difficult problem with a negligible advantage, but according to the definition of the CDH difficult problem, this is not feasible computationally.

Experiment and performance analysis

1) Comparison of different schemes.

Regarding cloud storage key security management, Li [28] first proposed Dekey, a cloud storage security deduplication scheme, to achieve secure and reliable control of convergent keys. However, when user data belongs to a predictable data set, the method is vulnerable to brute-force attacks launched by adversaries. Wang [5] proposed a cloud storage security deduplication scheme

Table 2 Comparison with other schemes

Schemes	Li et al.	Wang et al.	Kwon et al.	This sheme
Importing entity	KS	IS	IS + KS	Blockchain
Avoiding single-point failure	Y	N	Y	Y
Resisting attack I	N	Y	Y	Y
Resisting attack II	N	N	N	Y

Table 3 Notations and descriptions

Notations	Descriptions	Memory size(Bytes)
pk, sk	*Public key, Secret key*	*16*
K	*Encryption key*	*16*
T	*File label*	*32*
P	*System parameters*	*32*
τ	*The value of the root of Merkle*	*32*

Table 4 Computation comparison in different stages

Schemes	Li et al.	Wang et al.	Kwon et al.	This scheme
Stage I	*Sha*	*tSha*	$2Sha+2R_F+R_D+M_G$	*Sha*
Stage II	$Sha+R_E+R_D$	$2Sha+2P_O+X_G$	$Sha+2P_E+3M_G+X_G$	$3R_E+2R_D$
Stage III	R_E+R_D	$2Sha+2P_O+X_G$	$Sha+P_O+3M_G+X_G$	$3Sha+R_E+2R_D$

that supports key sharing to improve the management efficiency of convergent keys. However, this scheme introduces the third-party index server IS to manage the data, which may cause the information of user data to be leaked by the index server [29]. Kwon [30] proposed a cloud storage security deduplication scheme, which uses pair-based cryptography technology to achieve scalable and reliable converging key management. However, due to the introduction of the third-party key management server KS [31, 32], this scheme is also prone to conspiratorial attacks launched by adversaries. The method proposed in this article realizes secure and reliable key management by using blockchain and secret sharing mechanisms and ensures the security of the key [33]. Table 2 provides a security comparison between the scheme proposed in this article and other methods. Compared with other cloud storage key security management schemes, the method proposed in this article can resist violent attacks and collusion attacks launched by adversaries and has a higher security guarantee [14, 34]. (Attack I means violent attack, and attack II means conspiratorial attack).

Table 5 Storage comparison for key

Schemes	Li et al.	Wang et al.	Kwon et al.	This sheme
User	—	16	161	16
CSP	—	16	$n \times 16$	—
KS	$n \times 16$	—	16	—
IS	—	16	—	—
BC	—	—	—	$n \times 16$

2) The computing cost of file label generation, key distribution, and key recovery is evaluated.

In Table 3, we provide the symbols used in the proposed scheme and the amount of memory they occupy. In Table 4, we summarize the computational costs of different methods in three phases: file label generation (phase I), key distribution (phase II), and critical recovery (phase III). As seen from Table 4, the calculation cost of the proposed scheme is lower in the three stages, especially in stage I, which is lower than Wang's scheme and Kwon's scheme.

3) Evaluate the amount of storage space required for the key.

Table 5 summarizes the different entities in the different scenarios, such as the User, provider, key management server(KS), index management server (IS), and blockchain (BC) need to allocate storage space for encryption keys. The table shows that the scheme proposed in this article needs to give less storage space for encryption keys.

4) Test the time cost of encryption key generation.

This scheme includes a file-level encryption key (F-CK) and a block-level encryption key (B-CK). Therefore, this experiment will be divided into two parts for testing.

- The effect of file size on the time overhead of file-level encryption key generation.

In this scheme, the file-level encryption key is generated mainly through the OPRF protocol construction [35], and its calculation cost primarily depends on the original file's size. As shown in Fig. 7, a set of files is selected for testing in the experiment, whose file sizes (unit: MB) are 20, 40, 60, 80, and 100 as the abscissa variables. The experimental results show that this scheme has low computational cost in file-level encryption key

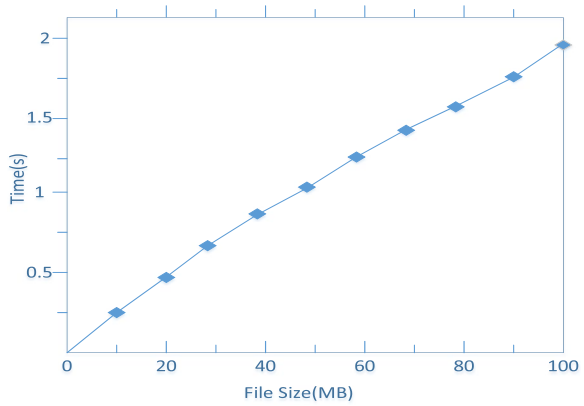

Fig. 7 The computing overhead of F-CK generation

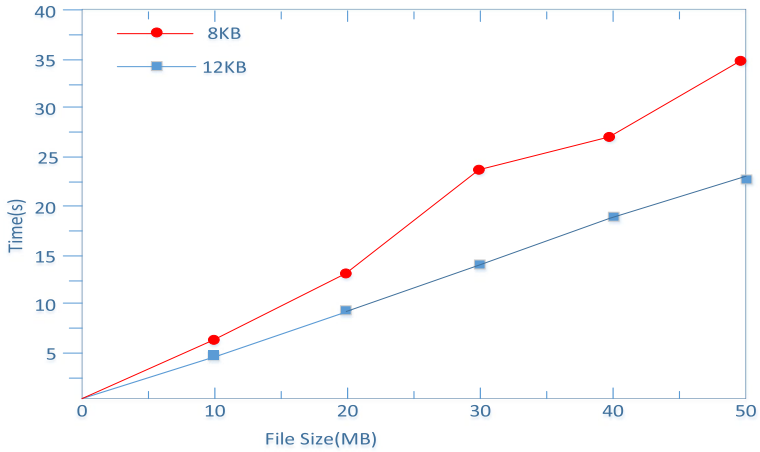

Fig. 8 The computing overhead of B-CK generation

generation. The encryption key is generated even for 100 MB files, and the calculation cost is only about 2 s.

- The effect of file size on the time overhead of block-level encryption key generation.

In this scheme, generating a block-level encryption key includes three stages: file segmentation, Merkel hash tree construction, and hash operation. Therefore, the computational cost of a block-level encryption key primarily depends on the file size and data block size [36]. As shown in Fig. 8, a group of files (unit MB) was selected in the experiment with lengths of 10, 20, 30, 40, and 50 and a group of data blocks with sizes of 8 KB and 12 KB, respectively. The experimental results show that the proposed scheme has a low computational cost in terms of block-level encryption key generation, and the larger the data block is, the smaller the time cost of block-level encryption key generation.

Application

In the ever-expanding world of the Internet of Things (IoT), managing security and privacy has become crucial. The number of IoT resources, including smart sensors, is continuously increasing, necessitating a comprehensive approach to ensure access and protect sensitive data. To address these issues, we have introduced a groundbreaking method that integrates the concepts of digital twins and blockchain to revolutionize IoT management. Digital twins are virtual representations of physical or virtual entities that are constantly updated and capable of autonomous communication. They encapsulate data access and view configurations, providing a layer of protection for IoT resources. Stakeholders do not directly access IoT resources but interact with digital twins, ensuring an additional layer of security. This method helps prevent unauthorized access to sensitive data and protects personal privacy. Blockchain technology is the cornerstone of our IoT management solution. It provides decentralization, transparency, and reliability, making it an ideal choice for protecting IoT systems. By validating digital twins and storing their configurations, blockchain ensures the legitimacy and identity of IoT resources. Smart contracts are programmed as trust structures to manage access to digital twins. This prevents third parties from directly accessing IoT resources, enhancing security and protecting privacy. The application of digital twins and blockchain has expanded to various IoT ecosystems, including healthcare, supply chains, and smart agriculture. In smart city scenarios, different stakeholders

need to access data streams from smart sensors for different purposes. Digital twins play a key role in providing views of data streams, enhancing security, and protecting privacy. Our method abstracts the complexity of data access management, making it applicable to a wide range of IoT use cases.

Conclusions

In this paper, firstly, a blockchain-based critical management system model is introduced, and the external and internal threats of attacks faced by this system model are described. Among them, under the assumption of honest and curious external and internal enemies, they will follow the system protocol honestly on the one hand. On the other hand, they will try to peep at the related user data information. To resist the malicious attacks launched by the adversaries, this article proposes a blockchain-based cloud storage key security management scheme to address the security issues faced by traditional critical management systems that rely too much on trusted third-party management servers. The method mainly cuts the converging key into several key fragments through the secret sharing scheme and submits them to achieve the reliable management of the converging key through the blockchain system while ensuring that the key management has a certain degree of fault tolerance and security. Security analysis shows that the proposed scheme has high safety and can resist brute-force attacks and conspiracy attacks launched by adversaries. Experimental and performance evaluation show that the proposed method has low time calculation overhead.

The proposal mentioned in this article aims to construct a secure and reliable cloud data storage system by combining blockchain technology, mainly utilizing the features of smart contracts, block structure, and transaction information in the blockchain system to achieve this goal. However, the blockchain system includes other features like consensus mechanisms, mining and broadcasting protocols, etc. The question of how to apply more characteristics of the blockchain system and digital twins to the cloud storage model and make the blockchain better compatible with existing cloud storage systems is also worth further exploration.

Abbreviations

OPRF	Oblivious Pseudo-Random Function
CSP	Cloud Storage Provider
BC	Blockchain
PoW	Proof of Work
NBM	Node business management
ID	Identification
O-PRF	Oblivious-Pseudo Random Function
CK	Encryption Key
MAC	Message Authentication Code
KS	Key management Server
IS	Index management Server
MB	MegaByte
KB	KiloByte
K-CK	File-level Encryption Key
B-CK	Block-level Encryption Key
CDH	Diffie–Hellman Problem

Authors' contributions

Conceptualization, J.H. (Jie Huang); methodology, J.H. (Jie Huang); software, J.H. (Jie Huang); investigation, J.H. (Jie Huang); resources, J.H. (Joseph Harding); data curation, J.H. (Jie Huang); writing—original draft preparation, J.H. (Jie Huang) and J.Y.Y; writing—review and editing, J.H. (Jie Huang), J.Y.Y; visualization, J.Y.Y; funding acquisition, J.H. (Jie Huang). Authors have read and agreed to the published version of the manuscript.

Funding

This study was funded by Natural Science Foundation of Hunan Province in 2022: " Design and application of cloud storage security architecture based on blockchain". (Research funder: Jie Huang, Grant number:2022JJ60091).

Availability of data and materials

No datasets were generated or analysed during the current study.

Declarations

Ethics approval and consent to participate

This article does not contain any studies with human participants or animals performed by any of the authors.

Competing interests

The authors declare no competing interests.

Received: 30 November 2023 Accepted: 31 December 2023
Published online: 10 January 2024

References

1. Liang H, Li M, Chen Y et al (2021) Architectural protection of trusted system services for Sgx enclaves in cloud computing[J]. IEEE Trans Cloud Comput 9(3):910–922
2. Huang Q, Yang Y, Yue W et al (2021) Secure data group sharing and conditional dissemination with multi-owner in cloud computing[J]. IEEE Trans Cloud Comput 9(4):1607–1618
3. Chen N, Li J, Zhang Y et al (2022) Efficient Cp-Abe scheme with shared decryption in cloud storage[J]. IEEE Trans Comput 71(1):175–184
4. Xia W, Jiang H, Feng D et al (2016) A comprehensive study of the past, present, and future of data deduplication[J]. Proc IEEE 104(9):1681–1710
5. Wang L, Wang B, Song W et al (2019) A key-sharing based secure deduplication scheme in cloud storage[J]. Inf Sci 504:48–60
6. Zhang Y, Su H, Yang M et al (2018) Secure deduplication based on Rabin fingerprinting over wireless sensing data in cloud computing[J]. Secur Commun Netw 2018:1–12
7. Cai Z, Li X, Ruiz R (2019) Resource provisioning for task-batch based workflows with deadlines in public clouds[J]. IEEE Trans Cloud Comput 7(3):814–826
8. Liu H, Liu Y (2021) Integrated monitoring algorithms for software data security situation on private cloud computing platform[J]. Int J Internet Protoc Technol 14(1):1–9
9. Nahmias D, Cohen A, Nissim N et al (2020) Deep feature transfer learning for trusted and automated malware signature generation in private cloud environments[J]. Neural Netw 124:243–257
10. Liu C, Yang C, Zhang X et al (2014) External integrity verification for outsourced big data in cloud and Iot: a big picture[J]. Futur Gener Comput Syst 49:58–67
11. Wei L, Zhu H, Cao Z et al (2014) Security and privacy for storage and computation in cloud computing[J]. Inform Sci Int J 258:371–386

12. Zhou T, Chen Z, Ming X (2021) A Novel hesitant fuzzy linguistic hybrid cloud model and extended best - worst method for multicriteria decision making[J]. Int J Intell Syst 37(1):596–624
13. Manikandan M, Subramanian R, Kavitha MS et al (2022) Cost effective optimal task scheduling model in hybrid cloud environment[J]. Comput Syst Sci Eng 42(3):935–948
14. Zhang S, Lee J (2019) Double-spending with a Sybil attack in the bitcoin decentralized network[J]. IEEE Trans Industr Inf 15(10):5715–5722
15. Yu B, Xu C-Z, Xiao B (2013) Detecting Sybil attacks in vanets[J]. J Parallel Distrib Comput 73:746–756
16. Mohiuddin I, Almajed H, Guizani N et al (2020) Ftm-lomt: fuzzy-based trust management for preventing Sybil attacks in internet of medical things[J]. IEEE Internet Things J 8(6):4485–4497
17. Zhang S, Lee J-H (2020) Mitigations on Sybil-based double-spend attacks in bitcoin[J]. IEEE Consum Electron Mag 10(5):23–28
18. Nicolas K, Wang Y, Giakos GC et al (2021) Blockchain system defensive overview for double-spend and selfish mining attacks: a systematic approach[J]. IEEE Access 9:3838–3857
19. Liu T, Sabrina F, Jang-Jaccard J et al (2021) Artificial intelligence-enabled ddos detection for blockchain-based smart transport systems[J]. Sensors 22(1):1–22
20. Eyal I, Sirer EG (2018) Majority is not enough: bitcoin mining is vulnerable[J]. Commun ACM 61(7):95–102
21. Motlagh SG, Mišić J, Mišić VB (2021) The impact of selfish mining on bitcoin network performance[J]. IEEE Trans Netw Sci Eng 8(1):724–735
22. Kang H, Chang X, Yang R et al (2021) Understanding selfish mining in imperfect bitcoin and ethereum networks with extended forks[J]. IEEE Trans Netw Serv Manage 18(3):3079–3091
23. Albrecher H, Goffard P-O (2022) On the profitability of selfish blockchain mining under consideration of ruin[J]. Oper Res 70:179–200
24. Chang V (2015) Towards a big data system disaster recovery in a private cloud[J]. Ad Hoc Netw 35:65–82
25. Laili Y, Tao F, Zhang L et al (2013) A ranking chaos algorithm for dual scheduling of cloud service and computing resource in private cloud[J]. Comput Ind 64:448–463
26. Sufi FK, Alsulami M, Gutub A (2023) Automating Global threat-maps generation via advancements of news sensors and AI. Arab J Sci Eng 48(2):2455–2472
27. Bellare M, Keelveedhi S, Ristenpart T. Dupless: Server-Aided Encryption for Deduplicated Storage[C]. USENIX Security Symposium, 2013: 179–194.
28. Li J, Chen X, Li M et al (2014) Secure deduplication with efficient and reliable convergent key management[J]. IEEE Trans Parallel Distrib Syst 25(6):1615–1625
29. Erdin E, Cebe M, Akkaya K et al (2020) A bitcoin payment network with reduced transaction fees and confirmation times[J]. Comput Netw 172:107098
30. Kwon H, Hahn C, Koo D, et al. Scalable and Reliable Key Management for Secure Deduplication in Cloud Storage[C]. Int Conf Cloud Comput. 2017:391–398.
31. Singh A, Gutub A, Nayyar A (2023) Redefining food safety traceability system through blockchain: findings, challenges and open issues. Multimed Tools Appl 82(14):21243–21277
32. Wei PC, Wang D, Zhao Y et al (2020) Blockchain data-based cloud data integrity protection mechanism[J]. Future Gener Comput Syst 102:902–911
33. Bitcoin: A Peer-to-Peer Electronic Cash System[EB/OL]. https://bitcoin.org/bitcoin.pdf.
34. Li L, Liu J, Chang X et al (2020) Toward conditionally anonymous bitcoin transactions: a lightweight-script approach[J]. Inf Sci 509:290–303
35. Li J, Xie D et al (2016) Secure auditing and deduplicating data in cloud[J]. IEEE Trans Comput 65(8):2386–2396
36. Al-Sharani F, Gutub A (2022) Increasing participants using counting-based secret sharing via involving matrices and practical steganography. Arab J Sci Eng 47(2):2455–2477

Publisher's Note