Blockchain-based Integrity Verification of Data Migration in Multi-cloud Storage

Kun Xu\textsuperscript{1,a}, Weiwei Chen\textsuperscript{2,b*}, Yanan Zhang\textsuperscript{3,c}
\textsuperscript{1}Army Engineering University, Nanjing, China
\textsuperscript{a}email: 1255284410@qq.com, \textsuperscript{c}email: 3554138477@163.com
\textsuperscript{b*}Corresponding author’s e-mail: \textsuperscript{b}email: njcww@qq.com

Abstract: In the process of multi-cloud storage data migration, data integrity is vulnerable to corruption, but the existing data integrity verification schemes for data migration across clouds are not highly reliable. To address this problem, a blockchain-based data integrity verification scheme for migration across clouds is proposed in this paper. In this scheme, a blockchain network is used instead of a third-party auditor. For each migration, a multi-cloud broker will send an integrity verification request to blockchain at three different times, and a smart contract will verify the data integrity according to the RSA-based homomorphic verification tags. Then, the security of the scheme is analyzed. Finally, simulation experiments and tests are conducted on Ethereum, and the results show the feasibility of the scheme.

1. Introduction
In multi-cloud storage, users often update data placement for better performance, so data migration is necessary. During the migration, the multi-cloud broker rebuilds the complete data using data blocks retrieved from CSPs, and then turns the data into new blocks by encoding and slicing, which should be stored in CSPs according to the new data placement strategy. Data integrity is more likely to be compromised during data migrations than during static placement due to uncertainty of the network. In reality, however, the real causes of compromised integrity during data migrations may not be network-induced data loss, but also malicious CSPs. Uncertainties of the network at the time of migration serve as an excuse for CSPs to cover up malicious breaches of data integrity. First, CSPs may normally lose data but hide their faults or intentionally delete unused data to reduce the storage burden. When the data migration comes, the broker retrieves the data from the multi-cloud storage system and finds that the data integrity is corrupted, and the CSPs can claim that it is due to network reasons. Second, after the broker sends the data to CSPs, the malicious CSPs may still have the above-mentioned integrity-breaking behaviors, and when data loss is found, the CSPs may also claim that it is due to data migration failure and evade responsibility. To sum up, in the process of data migration, it is not only necessary to avoid data integrity damage caused by uncertainties in the network, but also to pay attention to data integrity damage caused by malicious CSPs.

In order to solve the above problems, this paper sets up three integrity verifications in the process of multi-cloud data migration, which correspond to different purposes. As shown in Figure 1, the first integrity verification is done before the multi-cloud broker retrieves the data blocks, which aims to verify whether the CSPs break the data integrity and assign responsibility, and prevent the phenomenon that malicious CSPs break the data integrity but claim that it is due to the network. The second integrity verification is done after the broker retrieves the data blocks, with the purpose of verifying whether the
data has suffered integrity deficiencies during network transmission; the third integrity verification is done after the broker distributes the new data blocks to CSPs and before the broker deletes the data, with the purpose of verifying whether the data is successfully sent to each CSP, preventing data blocks from being lost before it is successfully sent, and preventing malicious CSPs from deleting the data but claimed to be due to migration failure.

Figure 1 Three integrity verifications in data migration

The integrity verification in Figure 1 may still fail to identify malicious CSPs if implemented using a traditional scheme. Traditional multi-cloud data integrity verification schemes typically rely on trusted third-party auditors (TPAs). However, in practice it is difficult to find honest TPAs. Dishonest TPAs may conspire with malicious CSPs to falsify evidence of data integrity, or to save compute resources by not performing integrity verification but sending falsified messages to users that the data integrity is intact. Blockchain, which has been developing hotly in recent years, is decentralized in nature, and in blockchain networks, decisions are made jointly by multiple nodes with decentralized authority, thus the content stored and the programs running on the blockchain are trusted.

In order to solve the problem of TPAs’ dishonesty, in this paper, we design a system model for integrity verification of data migration based on blockchain, which completely replaces TPA with blockchain network to avoid the phenomenon of integrity evidence falsification caused by TPAs’ dishonesty. Then the specific method of three times verification is designed in detail and the security of the method is analyzed in conjunction with RSA-based homomorphic verification tags. Finally, simulation experiments and tests are conducted on Ethereum to prove the feasibility of the method.

2. Related Works

Some of the studies verify the integrity directly by comparing the original data hash. The literature [1] stores the file hash to the blockchain and determines the data integrity by comparing the hash. The literature [2] uses mobile broker technology to deploy a distributed virtual machine broker model in the cloud and build a blockchain network on top of it, where smart contracts monitor data changes based on the unique hash value corresponding to the files generated by the MHT. Another part of the research verifies the integrity by generating challenge messages. The literature [3] proposed IBPA, an identity-based public auditing scheme for cloud data, to construct unpredictable and easily verifiable challenge messages using random numbers of blockchain, thus preventing malicious TPAs from forging audit results to deceive users. The literature [4] constructs challenge messages using the latest block hash of the blockchain to ensure randomness.

Unlike the above scheme with TPA, the literature [5] proposes a blockchain-based public auditing scheme framework without trusted third parties, where the user first calculates the tags of the data with the private key problemd by KGC, and then uploads the original data with the corresponding tagss to the CSP. when the user has the need for integrity verification, the verification process is completed by the smart contract. The architecture proposed in the literature [6] also omits third-party institutions altogether, with only CSPs of mutually untrusted data owners. data owners store lightweight authentication tagss on the blockchain and prove integrity by using constructed MHT. However, this scheme suffers from the vulnerability that the MHT root provided by the CSP can be generated and permanently stored when the data is first uploaded.
In a multi-cloud storage environment, the literature [7] uses user-provided information to generate random challenges and uses smart contracts to audit the challenge response messages problemd by CSP. In addition to the storage supervision contract for auditing, a trust management contract is deployed in this scheme to evaluate the trust level of each cloud storage service. The literature [8] proposes a blockchain-based data auditing scheme for multi-cloud storage to protect data integrity and accurately arbitrate service disputes. Users outsource their data to multiple CSPs and then work with the CSPs to generate integrity metadata for data auditing. In the auditing phase, blockchain is introduced to record the audit process interaction data and smart contracts are used to detect dishonest CSPs.

3. Background Knowledge

3.1. Blockchain
Since the birth of bitcoin [9] in 2008, the blockchain technology behind it has been gaining increasing attention from developers worldwide as an innovative technology in the field of security. The blockchain can be viewed as an open distributed ledger where new transactions are constantly packaged into new blocks, which grow in an orderly fashion over time as the blocks are continuously updated. As shown in Figure 2, the block contains Merkle Hash Tree (MHT), block hash, parent hash, version number, random number, timestamp, and other information to store the transaction, where the block hash is generated based on other information in the block, including the parent hash. Therefore, if a block in the blockchain is to be changed, all blocks from that block forward must be changed in order not to violate the rules of blockchain composition. Blockchain technology is a new application model of multiple computer technologies, which can realize decentralized peer-to-peer information transmission in a distributed system without mutual trust of nodes by integrating some core technologies, such as cryptographic hash, digital signature, consensus mechanism, timestamp technology, etc. In recent years, with the soaring popularity of blockchain technology, research on blockchain has been carried out in various fields, such as the Internet of Things, cloud computing, finance, healthcare, e-government, and so on. In general, blockchain has the following characteristics [10]:

- **Decentralization.** In a traditional centralized transaction system, any transaction must be verified by a central node, which increases the workload of the central server and carries the risk of service unavailability. In the blockchain network, multiple participants work together to maintain the ledger through P2P transmission and consensus mechanisms.
Persistence. Once a transaction is added to the blockchain, there is little possibility being deleted or modified.

Anonymity. In the blockchain network, users use system-generated addresses (generated by the hash of the user's public key) for transactions, and the user’s real information can be hidden.

Auditability. Just as Bitcoin stores user balances via the Unspent Transaction Output (UTXO) model, each input is split from the last output\textsuperscript{11}. Each transaction in the blockchain references the previous transaction status, which is updated once the transaction is added to the blockchain. Therefore, any transaction record can be easily traced and reviewed.

A smart contract in blockchain is a program deployed on the blockchain that can be executed automatically once the trigger conditions are met. Because of the nature of blockchain, the execution of a smart contract cannot be interfered with manually and the result is trusted and untamperable.

3.2. RSA-based Homomorphic Verifiable Tags

Ateniese et al.\textsuperscript{12} introduced homomorphic verification tags (HVT) firstly. Homomorphic verification tags is a non-falsifiable metadata for data block integrity verification that has the following two properties:

1) Homomorphism: Given two operations ""+"" and ""×", a function H is said to be homomorphic if it satisfies \( H(d + d') = H(d) \times H(d') \). Homomorphic label of data block \( d + d' \) can be obtained by combining the homomorphic label of data block \( d \) and the homomorphic label of data block \( d' \).

2) Blockless verification: With HVT, users can verify the integrity of the data block on the cloud server without fetching it.

This paper uses the RSA-based homomorphic hash function proposed in reference [13] to construct HVT. RSA-based HVT can support arbitrary size blocks, so it is suitable for the scenario of multi-cloud data migration in this paper.

Suppose \( p \) and \( q \) are two sufficiently large (large enough to be sufficiently safe) prime numbers. \( n = pq \), Euler function value \( \phi(n) = (p - 1)(q - 1) \), \( b \) is a random integer. \( p, q, \phi(n) \) are secret parameters, \( n \) and \( b \) are common parameters. Then the RSA-based HVT of data block \( d \) is:

\[
H(d) = b^d \mod n \tag{1}
\]

The function satisfies homomorphism, because

\[
\begin{align*}
H(d + d') &= b^{d+d'} \mod n \\
&= b^d b^{d'} \mod n \\
&= \left(b^d \mod n\right) \times \left(b^{d'} \mod n\right) \\
&= H(d) \times H(d')
\end{align*}
\tag{2}
\]

This function is secure enough to be used for data integrity verification. Suppose \( d \neq d' \) and \( H(d) = H(d') \), then \( b^d \mod n = b^{d'} \mod n \), so \( b^{d-d'} \equiv 1 \mod n \). According to Euler's theorem, \( d - d' \) must be a multiple of \( \phi(n) \). Since \( \phi(n) \) is secret and \( p, q \) are two sufficiently large secret prime numbers, it is difficult to find a \( d' \) to make the equation \( H(d) = H(d') \) true.

4. System Model

In order to avoid the phenomenon of falsification of integrity verification results caused by dishonest TPA, this section designs a system model based on blockchain for integrity verification of multi-cloud data migration, which completely replaces TPA with blockchain network. The system mainly contains four entities: multi-cloud broker, blockchain network, multi-cloud storage system, and key generation center. Each entity is described as follows.

Multi-cloud Broker (MCB): The broker is responsible for developing data placement policies for users, automating the implementation of data placement strategy updated (data migration), and sending integrity verification requests. Assuming it is honest.

Blockchain Network (BCN): consists of many blockchain nodes, with a master node for communication with the outside world. The blockchain network runs smart contracts to verify data integrity and records all intermediate data.
Multi-cloud Storage System (MSS): consists of multiple CSPs, each of which can store user data and respond to integrity verification challenges. However, CSPs may maliciously compromise the integrity of data and conceal malicious behavior.

Key Generation Center (KGC): The authority that generates the security parameters needed for integrity verification.

Figure 3 shows the verification steps of the system when the data is in the CSPs. First KGC generates the security parameters, where the secret parameters are stored in MCB and the public parameters are stored in the blockchain. Then MCB generates the challenge message and sends both the integrity verification request and the challenge message to the blockchain master node and MSS. Then the CSPs sends the proof of data possession to the blockchain master node. After that, the integrity verification process of the smart contract is triggered. Finally, the blockchain master node returns the integrity verification result to MCB.

5. Design of integrity verification scheme

| Symbols | Meaning |
|---------|---------|
| MCB     | Multi-Cloud Broker |
| BCN     | Blockchain Network |
| MSS     | Multi-Cloud Storage System |
| KGC     | Key Generation Center |
| \(d\)   | validated data |
| \(p\)   | a sufficiently large secret prime number |
| \(q\)   | another sufficiently large secret prime number |
| \(n\)   | public parameters \(n = pq\) |
| \(\phi(n)\) | secret parameters \(\phi(n) = (p - 1)(q - 1)\) |
| \(h(d)\) | \(h(d) = d \mod \phi(n)\) |
| \(m\)   | minimum number of blocks of reconstructed data |
Table 1 Symbols and meanings of Blockchain-based integrity verification scheme

| Symbols | Meaning |
|---------|---------|
| $d_i$  | the $i$-th data block |
| $CSP_i$ | CSP storing the $i$-th data block |
| $Tag(d_i)$ | HVT of the $i$-th data block generated by the $CSP_i$ |
| $Tag(d)$ | HVT obtained by smart contract merging all $Tag(d_i)$ |
| $Tag'(d)$ | HVT of data $d$ generated by multi-cloud broker |
| challenge | random integer used for verification challenge |
| Req     | integrity verification request |

The integrity verification scheme contains an initialization phase and three verification phases. In the initialization phase, KGC generates verification parameters. In the verification phase, MCB, MSS and BCN interact to complete the integrity verification of data. The flow of each phase is designed in detail below. Table 1 shows all the symbols in the scheme and their meanings.

5.1 Initialization phase

During the initialization phase, the first step is to ensure that the blockchain network has been started and that the smart contracts have been deployed. After the blockchain network is running, the smart contract can automatically store the various types of data received in a defined data structure and automatically trigger the integrity verification process when specific data is received. In addition, when data is stored for the first time, new integrity verification parameters need to be generated and saved so that the data can be validated in the future lifetime. The process of generating verification parameters is as follows.

1. MCB receives storage request of data $d$

   When MCB receives a new data storage request, it immediately informs KGC to generate the parameters used to validate the data.

2. KGC generates parameters $p, q, n, \phi(n)$

   KGC generates two sufficiently large prime numbers $p$ and $q$, and then calculate $n$ and the Euler function value $\phi(n)$ according to the formula in Subsection 3.2. Among these four parameters, only $n$ is a public parameter and the others are secret parameters, so KGC sends $p, q, n, \phi(n)$ to MCB and only sends $n$ to the master node of BCN.

3. MCB generates $h(d)$

   MCB calculates $h(d)$ based on the formula $h(d) = d \mod \phi(n)$, then stores it and all other parameters in the database as metadata for this data.

4. MCB sends data to MSS and then deletes it

   After calculating and saving all the verification parameters, MCB divides the data into a number of blocks, any $m$ of which can recover the complete data. The $m$ blocks are denoted as $\{d_i| i = 1, 2, ..., m\}$. Then MCB deletes the local data after sending all the blocks to CSPs.

   That is, if data $d$ is being stored in the MSS, the state of the entity holding the parameters in the system should be MCB holds the verification parameters $p, q, n, \phi(n), h(d)$, the BCN holds the common verification parameter $n$.

5.2 First verification

The first verification is after MCB receives data migration request and before the data blocks are retrieved. At this point all data blocks are stored in MSS and MCB does not own the data. In this case, the protocol in reference [13] is used, the method of comparing HVTs is adopted. $m$ data blocks are selected and the corresponding CSPs are required to calculate the HVT of these $m$ data blocks. Due to
homomorphism, the HVTs of the $m$ data blocks can be aggregated into the HVT of complete data. The specific steps are as follows.

1. MCB generates random integer $\text{challenge}$ and calculate $\text{Tag}'(d)$
   MCB generates random integer $\text{challenge}$ as the challenge number sent to CSPs. And calculate $\text{Tag}'(d)$ according to the formula $\text{Tag}'(d) = \text{challenge}^{h(d)} \mod n$ as a reference HVT for integrity verification.

2. MCB sends $\text{challenge}, \text{Tag}'(d), \text{Req}$ to BCN and sends $\text{challenge}, \text{Req}$ to MSS
   MCB sends $\text{challenge}, \text{Tag}'(d), \text{Req}$ to BCN master node. $\text{challenge}$ is recorded on the blockchain as intermediate data. $\text{Tag}'(d)$ is used as a reference HVT for the smart contract to verify integrity. $\text{Req}$ is used as the message that triggers the smart contract to perform integrity verification, making the smart contract ready to receive and integrate the HVTs of the data blocks. At the same time, MCB sends $\text{challenge}$ and $\text{Req}$ to the CSPs which has the data block $\{d_i|i=1,2,\ldots,m\}$.

3. $\text{CSP}_i(i=1,2,\ldots,m)$ calculates $\text{Tag}(d_i)$ and send it to BCN
   Having received $\text{Req}$. The $\text{CSP}_i$ storing data block $d_i(i=1,2,\ldots,m)$ reads parameter $n$ from BCN and calculates $\text{Tag}(d_i)$ according to the formula $\text{Tag}(d_i) = \text{challenge}^{d_i} \mod n$, and sends it to BCN master node.

4. The smart contract computes $\text{Tag}(d)$, compares $\text{Tag}(d)$ and $\text{Tag}'(d)$
   Because of homomorphism, the smart contract can calculates $\text{Tag}(d)$ according to the formula $\text{Tag}(d) = \prod_{i=1}^{m} \text{Tag}(d_i)$, and compare it with $\text{Tag}'(d)$. If the two are consistent, data integrity is considered uncorrupted.

5. The smart contract returns the result to the MCB
   The smart contract returns the integrity verification result. Meanwhile, it saves all intermediate datas so that to the errant CSP could be traced back in case a data integrity compromise is detected.

Figure 4 summarizes the process of the first verification. Moreover, the Euler number $\phi(n)$ is the Order of the multiplicative group of integers modulo $n$ $(\mathbb{Z}/n\mathbb{Z})^*$ and $h(d) \equiv d \pmod{\phi(n)}$, so

$$\text{challenge}^{h(d)} = \text{challenge}^d \pmod{n} \quad (3)$$

$$\text{challenge}^{h(d)} \mod n = \text{challenge}^d \mod n \quad (4)$$

Therefore, the above scheme is correct [13].

Figure 4 The first verification process
5.3 Second and third verification
The second verification is after the data blocks are retrieved and before the data is rebuilt. At this point MCB has the data blocks with unknown integrity. Fortunately, the intermediate data \( \{Tag(d_i)|i = 1,2,\ldots,m\} \) of the first verification is already stored in BCN. MCB calculates the HVT of the retrieved data blocks and compares them with \( \{Tag(d_i)|i = 1,2,\ldots,m\} \) stored in BCN in turn to verify the integrity.

The third verification is after the distribution of the data block and before the deletion of the data block. At this point, MCB has the correct data blocks, but the integrity verification method is the same as the first verification due to the limited computing resources of MCB.

6. Security Analysis
1. In the scheme proposed in this paper, each entity acts in its own interest.

   MCB is honest because MCB does not store data, it has no need to reduce the storage burden, and has to consider user satisfaction and be responsible for users’ data, so there is no malicious behavior like leaking secret parameters, maliciously destroying data integrity or colluding with or harboring CSPs. Therefore, the behavior of MCB, including generating \( h(d) \), saving the secret verification parameters of the data, generating the challenge number, calculating \( Tag'(d) \), and sending the verification request to BCN and CSP, are all true and reliable.

   The CSPs in MSS are not always honest; they may lose data but hide their fault, or deliberately delete infrequently used data to reduce the storage burden. However, the tags \( \{Tag(d_i)|i = 1,2,\ldots,m\} \) provided by CSPs in this scheme should be true, because CSPs should be penalized for not passing integrity verification.

   The nodes in the BCN are unreliable, but a few malicious nodes cannot have any impact on the correct functioning of the blockchain. The decentralized nature makes the data stored in the blockchain real and trustworthy, and the smart contracts running on the blockchain cannot be interfered with by humans.

   KGC is an authority and does not act in bad faith.

2. The scheme proposed in this paper is able to avoid the malicious effects of the untrustworthiness of CSPs and TPA in traditional schemes.

   In this scheme, the CSPs need to provide the correct tags \( \{Tag(d_i)|i = 1,2,\ldots,m\} \) in order to pass the integrity verification. The tags are generated from the data block and the challenge number. Since MCB generates a new challenge number at each verification, the CSP cannot forge the correct tags in advance, which means that the CSP must hold the complete data at the time of integrity verification in order to pass the verification. Integrity verification is performed before MCB retrieves the data blocks from CSPs and after sending the data blocks to the CSPs, so dishonest behavior of CSPs can be detected in this scheme.

   What’s more, BCN replaces the TPA in the traditional scheme to complete the integrity verification work, completely eliminating the malicious impact of untrustworthy TPAs. A malicious TPA may conspire with a malicious CSP to falsify evidence of data integrity or send a forged message to the user that data integrity is intact without performing integrity verification in order to save computational resources. However, because the data in the blockchain is not alterable and the execution of smart contracts is not subject to human intervention, the above-mentioned malicious behaviour can be avoided by executing the integrity verification process with smart contracts and storing all the intermediate data in the blockchain.

7. Simulation experiments
This section conducts simulation experiments on the blockchain-based migration data integrity verification scheme across clouds. The experiments are conducted on a private chain constructed by Ethereum, with one virtual machine simulating MCB, one virtual machine simulating MSS, and one virtual machine simulating BCN master node. The configuration of each virtual machine is shown in Table 2.
Table 2  environment configuration

| Hardware          | Software          |
|-------------------|-------------------|
| CPU: 1 core       | OS: Ubuntu 18.04  |
| RAM: 4GB          | Truffle: v5.0.2   |
| Disk: 50GB        | Testrpc: v6.0.3   |
| GPU: Intel(R) UHD Graphics | Geth: v1.8.21   |

Two sets of experiments are done to measure the execution time of the CSPs’ tags generation phase and the smart contract’s verification phase. The first set of experiments does not change the data block size (each block is set to 1M) and changes the number of data blocks. The second set of experiments does not change the number of data blocks (set to 6) and changes the size of each data block. Figure 5 and Figure 6 show the results of the two sets of experiments respectively.

![Figure 5](image1.png)  The time consumption of two phases changes with the size of data blocks

![Figure 6](image2.png)  The time consumption of two phases changes with the number of data blocks

The experimental results illustrate that the time used by the smart contract to verify integrity stays low regardless of the number or size of data blocks. This is because the smart contract only needs to aggregate the homomorphic tags of the data blocks and then compare it with the tag provided by the MCB, without the need for computationally heavy work. The computationally intensive work of
generating homomorphic tags is done by CSPs. Because of the limited throughput of blockchain and the high computing power of cloud servers, the scheme proposed in this paper is fully feasible.

8. Conclusion
In this paper, to address the problem that data integrity is vulnerable to corruption in the process of multi-cloud data migration, a blockchain-based data integrity verification scheme is proposed. In this scheme, a blockchain network is used instead of TPA, and for each migration, a multi-cloud broker will send integrity verification requests to the blockchain at three different times. A smart contract will verify the data integrity according to the RSA-based homomorphic verification tags. Then, the security of the scheme is analyzed. Finally, simulation experiments and tests are conducted on Ethereum, and the results prove the feasibility of the scheme.

Acknowledgments
This paper is one of the phased achievements of the National Natural Science Foundation of China (61402518) and Jiangsu Natural Science Foundation of China (BK20191327).

References
[1] ZIKRATOV I, KUZMIN A, AKIMENKO V, et al. (2017) Ensuring data integrity using blockchain technology. In: 2017 20th Conference of Open Innovations Association (FRUCT). Russia. pp. 534-539.
[2] WEI P C, WANG D, ZHAO Y, et al. (2020) Blockchain data-based cloud data integrity protection mechanism. Future Generation Computer Systems, 102: 902-911.
[3] XUE J, XU C, ZHAO J, et al. (2019) Identity-based public auditing for cloud storage systems against malicious auditors via blockchain. Science China Information Sciences, 62(3): 32104.
[4] ZHANG G, YANG Z, XIE H, et al. (2021) A secure authorized deduplication scheme for cloud data based on blockchain. Information Processing & Management, 58(3): 102510.
[5] LI S, LIU J, YANG G, et al. (2020) A Blockchain-Based Public Auditing Scheme for Cloud Storage Environment without Trusted Auditors. Wireless Communications and Mobile Computing, 2020: 8841711.
[6] LI J, WU J, JIANG G, et al. (2020) Blockchain-based public auditing for big data in cloud storage. Information Processing & Management, 57(6): 102382.
[7] PINHEIRO A, CANEDO E D, DE SOUSA R T, et al. (2020) Monitoring File Integrity Using Blockchain and Smart Contracts. IEEE Access, 8: 198548-198579.
[8] ZHANG C, XU Y, HU Y, et al. (2021) A blockchain-based multi-cloud storage data auditing scheme to locate faults. IEEE Transactions on Cloud Computing, 2021: 305771.
[9] Shin S, Kwon T. (2015) A Survey of Public Provable Data Possession Schemes with Batch Verification in Cloud Storage. Journal of Internet Services and Information Security, 5(3): 37-47.
[10] YANG C. (2020) Research on Blockchain-based Cloud Storage Data Integrity Detection. School of Computer Science and Engineering.
[11] GAETANI E, ANIELLO L, BALDONI R, et al. (2017) Blockchain-based database to ensure data integrity in cloud computing environments. In: the First Italian Conference on Cybersecurity (ITASEC17) Venice. pp. 146-155.
[12] Ateniese G, Burns R, Curtmola R, et al. (2007) Provable data possession at untrusted stores. In: Proceedings of the 14th ACM conference on Computer and communications security. Alexandria pp. 598-609.
[13] Gazzoni Filho D L, Barreto P S L M. (2006) Demonstrating data possession and uncheatable data transfer. Cryptology ePrint Archive, 1: 150–159.