Journal of Combinatorial Mathematics
and Combinatorial Computing
www.combinatorialpress.com/jcmcc

Combinatorial Press

Cloud Data Integrity Verification Algorithm for Smart Accounting Informatization in Cloud Computing

Yuhua Chen[1,2,✉], Hasri Mustafa[2], Asna Atqa Abdullah[2], Ziqin Feng[3]

[1] *School of Finance and Taxation, Zhengzhou Technology and Business University, Zhengzhou, Henan, 450000, China*
[2] *School of Business and Economics, University Putra Malaysia, Serdang, 43400, Malaysia*
[3] *School of Management and Economics, North China University of Water Resources and Electric Power, Zhengzhou, Henan, 450000, China*

ABSTRACT
In the process of sharing accounting information using cloud computing technology, the integrity of the data is related to the security of the transmission and utilization of accounting information. For this reason, this paper studies the algorithm optimization based on the multi-branch path tree LBT. Multi-branch path tree LBT adopts distributed data storage method to reduce the number of hash operations. The data integrity auditing scheme is designed for different phases of cloud auditing, and the dynamic update process of cloud data is optimized to improve the data integrity verification effect. This algorithm can still maintain a high challenge success rate after more than 300 challenge data blocks, and the total overhead of the experimental computation does not exceed 8 ms, and the verification efficiency is also better than the comparison algorithm. Therefore, the research idea of this paper has validity and has improved effect on data integrity verification in the process of cloud computing smart accounting informatization.

Keywords: data integrity verification, multi-branch path tree LBT, cloud computing, smart accounting informatization

1. Introduction

With the popularization of the Internet and technological progress, enterprise informationization has become an inevitable trend of enterprise development. In enterprise informatization, accounting informatization is the fastest growing and the most important part [1-3]. Cloud accounting is an emerging technology in accounting informatization, which is based on cloud computing, accounting information processing, storage, analysis and other functions are all moved to the cloud, which greatly improves the efficiency and accuracy of enterprise accounting information processing, but also brings more advantages and convenience [4-7].

✉ Corresponding author.
E-mail address: chenyuhua25218@163.com (Y. Chen).
Received 05 March 2024; Revised 25 May 2024; Accepted 05 December 2024; Published Online 16 April 2025.
DOI: 10.61091/jcmcc127b-500

With the development of cloud computing technology, cloud storage technology has become the data storage method chosen by more and more enterprises and individuals. However, the ensuing data security issues are also of great concern. Among them, data integrity verification, as an important part of data security, is particularly important for cloud storage technology [8-11]. In traditional data storage methods, data integrity verification is relatively easy to realize. However, in cloud storage technology, data integrity verification faces greater challenges. Since the data is stored in the cloud, users cannot directly access the physical storage device, so they cannot directly confirm whether the data has been tampered with [12-15]. Once the data is tampered with, it may lead to serious consequences, such as enterprise data leakage, personal privacy leakage and other problems. Therefore, data integrity verification is particularly important in cloud storage technology [16-18].

Accounting informatization is a kind of accounting processing based on modern technology and information system. It realizes the automation and efficiency of accounting and financial management by combining accounting management with information technology. Its significance lies in improving the efficiency and quality of accounting work, reducing the occurrence of human errors, and improving the accuracy and reliability of accounting data. Literature [19] examined the enterprise financial accounting informatization management. By introducing the concept and significance of big data, the necessity of enterprise financial accounting informatization construction, as well as the challenges faced by enterprises in promoting financial accounting informatization management, and proposed effective measures. Literature [20] launched a review of cloud accounting and accounting informatization, and discussed the application pathways by analyzing the advantages of cloud accounting applied in enterprise accounting informatization. Literature [21] introduced the evaluation method of accounting informatization. The evaluation index system is built from the aspects of accounting informatization implementation and application efficiency. Entropy and cross-entropy methods based on intuitionistic fuzzy sets are proposed to reduce errors. The evaluation results are obtained by combining the weights of the indicators with weighted ratings, and the effective feasibility of this method is described. Literature [22] reveals the deficiencies in the development and application of accounting informatization in numerous enterprises, and proposes countermeasures to cope with the challenges by analyzing the challenges of accounting computerization in financial enterprises. Literature [23] emphasizes the importance of accounting informatization, and by applying cloud computing to the enterprise accounting management system, the results show that small and medium-sized enterprises realize that accounting information management can effectively improve economic settlement. It also introduces the coping strategies for enterprises to face the risk of accounting informatization. Literature [24] combines machine learning algorithms to construct an intelligent management accounting information system and designs experiments to verify the performance of the model. The results reveal that the constructed system is able to meet the demand for intelligent accounting information. Literature [25] reveals that the manufacturing accounting informatization has undergone significant changes in the context of big data. It is pointed out that the development of big data brings many opportunities and challenges to enterprises. It also indicated that measures such as improving data application ability and perfecting data security system are important strategies for enterprises to cope with the challenges.

In modern information systems, data plays a crucial role. Incompleteness of data may lead to serious consequences such as information errors, decision-making mistakes and even system paralysis. Therefore, ensuring data integrity is a prerequisite for the safe and stable operation of the system. Literature [26] introduces the research on data integrity verification (DIV) based on literature review, and classifies its advanced solutions, analyzes the main topics and technical means of DIV solutions that satisfy different requirements as well as the possible challenges in the future, and puts forward the future research direction. Literature [27] aims to propose an effective data integrity verification technique and the building block of this technique is the Paillier Homomorphic Cryptography (PHC) system. The effectiveness of the proposed method is verified by building an application based on

Hadoop and MapReduce framework. Literature [28] proposes auditing schemes that enable security detection of user OLs. By grouping the OLs, the vt of the OLs is computed based on the association operation relationship.The results of the study show that the scheme achieves stronger privacy protection and efficient processing of data dynamics with security and effectiveness. Literature [29] proposes a data integrity verification method based on short signature algorithm, which supports privacy protection and public auditing by introducing TPA. Based on experiments it is shown that this scheme is relatively efficient and secure. Literature [30] proposes SIBAS as a data integrity checking scheme, which utilizes TEE to check outsourced data, which can check the integrity of outsourced data and implements secure key management in TEE through Shamir's threshold scheme. Experiments have proved that the scheme is feasible and efficient in practice. Literature [31] proposes a public data integrity verification scheme based on algebraic signatures and elliptic curve cryptography, and constructs partitioned hash tables to perform operations such as deletion and modification. Based on the security analysis and performance evaluation of the scheme related to the literature, it is pointed out that the scheme has advantages in integrity verification and dynamic update. Literature [32] describes a blockchain-based distributed data integrity verification protocol for multi-cloud environments, which uses multiple verifiers to verify data. The performance analysis shows that the protocol reduces the verification time as well as the computational and communication cost of the verifiers than using a single verifier.

For the data integrity verification problem of accounting cloud storage, this paper designs a test data integrity checking algorithm. Each node in the improved LBT stores data and retains the hash operation to reduce the number of operations. The data integrity audit scheme is designed from the phases of initialization, uploading, challenge-answer-audit, and the dynamic update process of cloud data is improved for user operations such as data modification, data insertion and data deletion. The cloud data update strategy for facing batch of data blocks is also discussed to enhance the data integrity verification. The overhead of this paper's algorithm is theoretically analyzed, and then experiments on time-consumption, overhead, challenge success rate, and verification efficiency are designed separately to test the effectiveness of this paper's algorithm.

2. Integrity validation issues for accounting cloud data

2.1. Cloud Computing Theory and Related Technologies

Cloud computing is currently a research hotspot in computer science and technology, which has gained the attention of many enterprises and institutions and relevant Internet experts, and is an important trend in the future development of computer network technology. A typical cloud computing platform needs to have: gridded data storage matrix network, firewall equipment, computing resources equipment, and allows users to use a scalable cloud storage space remotely through leasing to achieve cloud application services. The complete cloud computing architecture should include: access layer, core layer, resource aggregation layer, API interface layer and application layer [33].

The combination of accounting informatization and cloud computing can effectively enhance financial shared management, which can greatly improve work efficiency and reduce hardware investment costs. For example, the financial management (accounting, finance) service system constructed by using cloud computing technology and SOA mode can realize the enterprise access to shared resources and reduce the cost of enterprise informatization [34].

2.2. Information security issues under the financial sharing model

Financial sharing breaks the limitations of geographic location and time, unifies the management of enterprise finance and puts relevant financial information on the network to realize data sharing. This requires advanced accounting informatization construction as a support, especially the accounting information security issues under the financial sharing mode, which needs special attention.

Compared with the traditional financial management mode, financial sharing mainly has the following three kinds of changes: the financial organizational structure mode has changed, the financial sharing based on the network environment has changed the traditional way of financial work, and the security risks of accounting information have become more prominent.

2.3. Data Integrity Validation Issues in Accounting Cloud Storage

The process of data integrity verification in cloud storage is also known as cloud auditing, and since public cloud auditing schemes are more practical in real life, only the publicly verifiable cloud auditing model and its algorithm description are detailed in this section.

2.3.1. System model. The public cloud auditing system model generally consists of three independent individuals, each of which is defined in detail as follows:

User (U): the user consists of two parts, the first part is the data owner, who has limited space for local storage resources but has a huge amount of data to be stored to the cloud server, and they can customize their choice of cloud storage service. The second part is the cloud users who are legally authorized by the data owner and they can access the data of that data owner stored in the cloud servers.

Cloud Servers (CSs): managed by Cloud Service Providers (CSPs), they have significant advantages in terms of data storage, storage space, computing power, etc. When users enjoy cloud storage services, the cloud servers will also directly control their outsourced data.

Third Party Auditor (TPA): possesses rich auditing experience and capability that the user does not have, he is also honest and curious, and his main responsibility is to verify the integrity of his data instead of the user, i.e., perform auditing tasks.

User U relies on cloud service provider CSPs to store and maintain large amounts of data, and they dynamically interact with each other as needed to form a secure data flow. Because users no longer have local copies of their data, ensuring the integrity of the data stored in the cloud is critical. To reduce the computational burden on users, they rely on a third-party auditor, TPA, to verify the integrity of their outsourced data. First, the user sends an audit request to the TPA: the TPA responds to the user's request by sending an audit challenge to the CSP, which generates an audit proof based on the TPA's challenge and returns it to the TPA: the TPA validates the audit proof and returns an audit report to the user. In addition, during the auditing process, it is important to prevent data privacy from being stolen by the TPA.

2.3.2. Requirements. An effective cloud storage data integrity verification program must meet the following basic conditions:

1) High probability of integrity verification: when data stored in the cloud is tampered with or deleted, the TPA can audit the current state of the data (intact/corrupted) with a very high probability without accessing the entire intact outsourced data.

2) Unlimited verification: Integrity verification schemes cannot limit the number of integrity verification of cloud-stored data, i.e., the verifier can perform countless integrity verification operations on data stored in the cloud.

3) Computing and communication overhead: the computing and communication overhead of the data storage phase and the auditing phase should be as low as possible, i.e., the computing and communication efficiency should be as high as possible in order to facilitate the enjoyment of the cloud storage service by mobile devices with weak computing power.

4) Cloud storage overhead: User-generated metadata should be as small as possible to minimize storage overhead in the cloud.

5) Non-interactivity: In the auditing phase, users can entrust the TPA to verify the integrity of their outsourced data without interacting with the GSP.

2.3.3. Basic framework. The basic framework of a fundamental cloud storage integrity verification scheme can be divided into the following two main phases:

1) Data storage phase. The user processes (chunks) the data file and generates a label for the data file, generates a verifiable metadata for each data block, and defines a data authentication metaset, and then outsources the data file, file label, and data authentication metaset together to be stored to the remote CSP, while the user locally only needs to save his private key and the label of the data file, where the label generally contains the file name, the file being divided into data blocks, the number of blocks and some necessary public parameters.

2) Data Audit Phase. TPA sends a random audit challenge to the CSP, then the CSP generates an audit proof based on the audit challenge using its saved data files and data authentication metasets and returns it to TPA, and finally, TPA verifies the proof to determine the integrity of the user's outsourced data.

2.3.4. General process of validating an algorithm. The integrity verification scheme for cloud stored data consists of two phases, data storage and data auditing, by employing a sampling strategy to verify the integrity of data files stored in the cloud. The specific implementation consists of the following six polynomial-time algorithms:

1) Data storage phase. $Setup(\lambda) \to (cp)$: Executed by the system, input security parameters in, output public parameters cp.

$keyGen(\lambda, cp) \to (pk, sk)$: Executed by the user, input system security parameter in, system public parameter cp, output user's public-private key pair (pk, sk).

$SigGen(\lambda, cp, sk, F) \to (Tag_F, \Phi)$: Executed by the user, input system security parameters into, system public parameters cp, user private key sk, data file F, output data file labels Tag_F, authentication metadata collection Φ, and send (Tag_F, Φ, F) to the CSP, which provides appropriate storage services to the user.

2) Data auditing stage. $Challenge(Tag_F, pk) \to (Q)$: Executed by TPA, inputs data tag Tag_F and user public key pk, outputs audit challenge Q, and sends to CSP.

$\Pr oofGen(Q, F, pk) \to (P)$: Executed by CSP, inputs challenge request Q, data file F, user public key pk, outputs audit proof P, and sends it to TPA.

$Verify(Q, Tag_F, pk, P) \to 0/1$: Executed by TPA, input challenge request Q, input data tag Tag_F, user public key pk, audit proof P, output $0/1$ and return corresponding audit result to user.

3. Cloud Data Integrity Verification Algorithm

3.1. BLS Signature Algorithm

In cryptography, the BLS signature scheme allows the user to verify that the signer is genuine. This signature type is verified using bilinear pairs and BLS signatures are a type of elliptic curve signatures. Signatures generated using BLS signature schemes are often referred to as short signatures, BLS short signatures, or simply BLS signatures.The security of BLS signature schemes is predicated on the existence of randomized predicate machines and the difficulty of the computational Diffie-Hellman problem in the GDH group. The use of BLS signatures is effective against indexed algorithmic attacks and they are shorter than normal signatures at the same level of security [35].

The BLS signature algorithm is based on bilinear mapping defined as:

$$e: G \times G \to G_T \tag{1}$$

where G_1 and G_2 are two GDH groups and G_T is another multiplicative cyclic group of prime p.

The bilinear mapping e has the following properties:

Computability: there exists an efficient algorithm to compute the bilinear mapping e.

Bilinear: for any $h_1 \in G$, $h_2 \in G$, $a, b \in Z_p$. e is satisfied if G_1 and G_2 are groups:

$$e\left(h_1^a, h_2^b\right) = e\left(h_1^b, h_2^a\right) = e\left(h_1, h_2\right)^{ab} \tag{2}$$

Non-degeneracy: $e(g, g) \neq 1$, and g is a generating element of group $G_l\left(G_2\right)$.

The BLS signature scheme contains the following three functions:

$KeyGen(\cdot)$ Key generation: the key generation algorithm chooses a random integer x in the interval $[0, r-1]$ to get the private key $sk = x$ and the public key $pk = g^x$.

$SigGen(\cdot)$ Signature: given a private key x and a message m, compute the signature by hashing $h = H(m)$ the bit string m and output the signature value $sig / \sigma = h^x$.

$Verification(\cdot)$ Verification: given a signature σ and public key g^x, verify that the following equation holds:

$$e\left(\sigma', g\right) = e\left(h, g^x\right) \tag{3}$$

If formula (3) holds, then the representative signature is legitimate:

$$e\left(g^x, h\right) = e\left(g, h^y\right) \tag{4}$$

Based on the property of bilinearity, $e(g,h)^x = e(g,h)^y$ can be calculated to get $x = y$, and further to get $\sigma' = h^x = \sigma$, which means that σ' is valid and is indeed signed by the private key x.

3.2. PDP Program

Another classic PDP scheme uses randomly extracted data blocks for authentication, which can greatly reduce the I/O communication, reduce the bandwidth occupied in message transmission, and improve the efficiency of authentication. The scheme consists of two main entities, the user and the cloud storage server. The process of performing verification is similar to the POR model, but PDP does not support the recovery of damaged data.

The PDP scheme verifies whether the cloud server retains a file intact.The processing of the PDP scheme is divided into two main steps: the Setup phase and the Challenge phase.

Setup phase: the user preprocesses the file F. Execute the key generation function $KeyGen(\cdot)$ to generate the public key pk and private key sk. Then, divide the file F into n blocks, noted as $F = \{m_1, m_2, \ldots, m_n\}$, and utilize the label generation function $TagBlock(\cdot)$ to generate the metadata T. Send the processed file F', to the cloud server, and then delete the locally stored copy of the file.

Challenge phase: Throughout the challenge-response process the user acts as the initiator of the validation and the cloud server stores the file F' and responds to the validation request sent by the user. The user randomly generates the subscripts of the data blocks to be verified and generates a verification request R, the cloud server receives this verification request R and then runs the $GenProof(\cdot)$ algorithm to compute the proof P and returns it to the verifier, who uses the $CheckProof(\cdot)$ to verify whether the returned proof P is correct or not, and outputs the verification result of 0/1 (0 means the data is corrupted, 1 means the data is intact).

The PDP classic scheme contains the following 4 main algorithms:

1) $KeyGen\left(1^k\right) \to \left(pk, sk\right)$. Executed locally by the user during system initialization. 1^k for the input security parameters, returning a pair of public and private keys $\left(pk, sk\right)$.

2) $TagBlock(sk, F') \to \{T\}$. $TagBlock(\cdot)$ The algorithm is executed by the user and generates metadata T, also called homomorphic signature tag set. The input parameters of this algorithm include the private key sk and the data file F', returning the authenticated metadata T.

3) $Gen\,\mathrm{Pr}oof(pk, F', T, chal) \to P$. The algorithm is run by the cloud storage server to generate the proof of integrity P. The input parameters include the file F', the public key pk, the set of authentication metadata T and the challenge request chal that returns the proof information P.

4) $Check\,\mathrm{Pr}oof(pk, chal, P) \to (1, 0)$. It is run by the user to validate the proof P returned by the cloud storage server. Input parameters are public key pk, challenge request $chal$ and proof P. The verification result is "1" or "0", where 1 indicates that the data is complete and "0" indicates that the data is corrupted.

As a static audit verification model, the computational cost of the PDP scheme is mainly concentrated in the tagged value generation function $TagBlock(\cdot)$. Generating labels for all data blocks and using modulo and power operations has a high time complexity, but for statically stored files, this time-complex algorithm is only used when preprocessing the data, which is only done once, so the impact is minimal. Only c randomly selected chunks of data are verified in the challenge-response phase, not all of them. This c-value is usually 460, so even though the audit phase also involves modulo power operations, it is not computationally expensive because the c-value is not large. So overall, when no dynamic operations are performed, the auditing computational complexity of the PDP scheme is $O(1)$ with storage complexity and communication complexity, and the computational and communication costs are low [36].

3.3. Improved Accounting Cloud Data Integrity Verification Algorithm

3.3.1. System improvements. This scheme stores data at each node in the improved LBT of multi-branch path tree and retains the unique inter-node hash operation feature in the traditional MHT. In the integrity auditing scheme of the improved LBT data structure, assuming that the cloud auditor requests to verify the integrity of data blocks m_3 and m_{14}, when the cloud auditor verifies data block m_3, it only needs to perform hash operation once:

$$f(m_1) = h\left(h(m_1) \| f(m_3) \| f(m_2) \| f(m_4) \| f(m_5)\right) \tag{5}$$

The root node R for integrity verification can be obtained, and similarly m_{14} can be examined with only two hash operations:

$$f(m_4) = h\left(h(m_4) \| f(m_{14}) \| f(m_{15}) \| f(m_{16})\right) \tag{6}$$

$$f(m_1) = h\left(h(m_1) \| f(m_4) \| f(m_2) \| f(m_3) \| f(m_5)\right) \tag{7}$$

In this auditing scheme, the cloud auditor performs a total of three hash operations, which is significantly less than the number of operations required in the traditional Merkle hash tree integrity auditing scheme. The improved LBT structure shortens the authentication path length to a certain extent, which improves the auditing efficiency of the cloud auditor.

3.3.2. Specific implementation programs. A bilinear mapping is defined as $e: G \times G \to G_T$, where G is a cyclic multiplicative group, which is also a GDH group, G_T is another cyclic multiplicative group, which are both of prime order $p \circ g$ is the generating element of group G, and $h(\cdot): \{0,1\}^* \to G$ is a hash function with password. The user file M is divided into n data blocks: $M = (m_1, m_2, \cdots, m_n)$.

The data integrity auditing scheme based on the improved multi-branching path LBT structure consists of an initialization phase, an uploading phase, and a challenge-answer-audit phase, and each phase consists of the following polynomial algorithm:

Initialization Phase $KeyGen\left(1^k\right)$: The cloud auditor randomly selects a number $\alpha \leftarrow \square_p$, $u_1, u_2, \cdots, u_s \leftarrow G$ and computes $v \leftarrow g^{\alpha}$. The private key of the cloud auditor is $sk = \alpha$ and the public key is $pk = \left(v, \left\{u_j\right\}_{1 \le j \le s'} g\right)$.

Upload phase $TagGen\left(M, sk\right)$: For each chunk $M = \left(m_1, m_2, \cdots, m_n\right)$ of file M, the user randomly selects an element $\delta \leftarrow G$, so that file M is uniquely identified as $\Lambda = name \| n \| \delta \| sig_{sk} name \| n \| \delta)$. Then the user end sends the identification Λ and the chunk data $m_i (i = 1, 2, \ldots, n)$ of file M to the TPA at the cloud auditor end. With the improved LBT authentication data structure, the root node R is computed and the root node R is signed $f(R)^{\alpha} \leftarrow sig_{sk}(f(R))$ with its own private key $sk = \alpha$. And sends the tag $t = sig_{sk}(f(R))$ to the client as a message acknowledgement.

After that the TPA signs each small data block $m_i = \left(m_{i1}, m_{i2}, \cdots, m_{is}\right)$, where $i = 1, 2, \ldots, n$, is signed with the signature algorithm:

$$\sigma_i \leftarrow \left(h\left(m_i\right) \cdot \prod_{j=1}^{s} u_j^{m_{ij}} \right)^{\alpha} \tag{8}$$

Obtain the data block signature set $\Phi = \left\{\sigma_i\right\}_{1 \le i \le n}$. The cloud auditing end TPA sends the initialization file $m^* = \left\{M, \Phi, \Lambda, t\right\}$ to the cloud storage end CSS, keeping only the label t.

Challenge phase $Challenge\left(\cdot\right)$: Authorize the trusted cloud auditor to complete the proxy auditing work and generate the proxy agreement, and the authorized auditor randomly selects a random sample of c elements from the data block index set $[1, n]$ to form the data block challenge subset $Q = \left\{\left(i, v_i\right)\right\}_{1 \le i \le c}$, where $v_i \leftarrow f(t, i, \tau)$, τ are timestamps, and then sends the generated challenge information pairs set to the cloud storage end periodically.

Response phase $Gen\,\mathrm{Pr}oof\left(M, T, chal, pk\right)$: After receiving the set of challenge information pairs, the cloud storage end runs the evidence generation algorithm and computes them respectively:

$$\mu_j = \sum_{\{(i, v_i)\} \in Q} v_i m_{ij} \in \square_p \tag{9}$$

Of which $j = 1, 2, \cdots, s$, and:

$$\sigma = \prod_{\{(i, v_i)\} \in Q} \sigma_i^{v_i} \in G \tag{10}$$

This will be followed by Data Integrity Exhibit P:

$$P = \left\{ \left\{\mu_j\right\}_{1 \le j \le s}, \sigma, \left\{h\left(m_i\right), \Omega_i\right\}_{1 \le i \le c}, sig_{sk}(f(R)) \right\} \tag{11}$$

Send it back to the Cloud Audit Terminal TPA.

Audit Phase $VerifyProof\left(P, pk\right)$: The cloud audit TPA receives evidence P and runs the audit algorithm. It first returns $\left\{h\left(m_i\right), \Omega_i\right\}_{1 \le i \le c}$ from evidence P, calculates the root hash value $f(R)$, and then verifies $e(t, g) = e\left(f(R), v\right)$. If the equation does not hold, the verification fails and outputs "reject". If and only if $f(R)$ passes, then continue to validate the equation:

$$e(\sigma, g) = e\left(\prod_{\{(i,v_i)\}\in Q} h(m_i)^{v_i} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, v \right) \tag{12}$$

If or not it is valid. If it is valid, output "accept" to prove that the data is complete. If it is not complete, output "reject", it means that the data has been destroyed, the occurrence of additions, deletions and other operations.

3.3.3. Dynamic updating of data. The improved LBT structure can support not only the update of a single data block, but also the update of batch data.

1) Data Modification. Suppose the Client wants to modify the i st data block m_i to m_i', then perform the following steps:

1. The Client generates an update request message $updateInfo = (M, i, m_i')$ and sends it to the TPA of the cloud auditing terminal, where M means the modification operation.

2. After receiving the update request, the cloud auditing TPA generates the corresponding signature by Equation (13):

$$\sigma_i' \leftarrow \left(h(m_i') \cdot \prod_{j=1}^{s} u_j^{m_{ij}'} \right)^{\alpha} \tag{13}$$

Then send an update request $update' = (update, \sigma_i')$ to the CSS on the cloud storage side.

3. After receiving the update request $update'$, the CSS on the cloud storage side runs the $Update(\cdot)$ algorithm and performs the following operations:

i. Modify the originally stored data block m_i, to m_i', and output a new file M'.

ii. Change the originally stored data block signature σ_i, to σ_i', and output the new set of signatures Φ'.

iii. Replace $h(m_i)$ with $h(m_i')$ and recalculate the new node hash $f(m_i')$, the authentication auxiliary information Ω_i of the data block need not be changed. Compute the new root node R' using the number type authentication structure.

iv. For the above update operation, the CSS on the cloud storage side returns an evidence of the update to the TPA on the cloud auditing side $P_{update} = (\Omega_i, h(m_i), R')$.

4. After receiving the evidence P_{update} from the CSS, the TPA at the cloud auditing end runs the $VerifyUpdate(\cdot)$ algorithm and performs the following operations:

i. Calculate the root node R using $\{\Omega_i, h(m_i)\}$.

ii. Verifies root node R' and verifies whether Equation $e(t, g) = e(h(R), g^{\alpha})$ holds, where α is the private key of the auditor. If it holds, then the validation passes. If the equation does not hold, the validation fails.

iii. Recompute root node R'' using $\{\Omega_i, h(m_i')\}$ and compare whether $R' = R''$ holds.

iv. If the validation passes, send the new label $t' = sig_{sk}(h(R'))$ to the CSS at the cloud storage end for updating.

The CSS at the cloud storage end does not need to reconstruct the whole tree structure during the update operation, therefore, compared with the traditional MHT structure, this scheme simplifies the process of updating the data and thus reduces the computational overhead.

2) Data insertion. Assuming that the data owner wishes to insert data block m^* after the ind data block m_i, the operation steps are similar to the process of data modification, so this operation also does not change the original tree structure.

1. After receiving the evidence of insertion from the client, the cloud auditor TPA generates a new node hash $f(m_i') = f(m_i \| m^*)$ and a new signature:

$$\sigma_i' \leftarrow \left(h\left(h(m_i') \| h(m^*)\right) \cdot \prod_{j=1}^{s} u_j^{m_{ij}'} \right)^{\alpha} \tag{14}$$

Then send an insertion request $insert = (i, m^*)$ to the CSS on the cloud storage side.

2. After receiving the insertion request, the CSS on the cloud storage side runs $Update(\cdot)$ the algorithm. The newly inserted data block m^* is stored in the server and then the tree structure is updated. It only needs to recalculate the hash value of the node corresponding to the modified data block m_i', as well as the parent node it passes through in the authentication path to the root node, in order to reduce the amount of computation and improve the efficiency. Finally, the CSS on the cloud storage side returns the evidence of the operation that generated the data insertion to the TPA on the cloud auditing side:

$$P_{insert} = (\Omega_i, h(m_i), R') \tag{15}$$

3. After receiving the evidence P_{update} from the CSS, the TPA at the cloud auditing end runs the $VerifyUpdate(\cdot)$ algorithm and performs the following operations:

i. Calculate the root node R using $\{\Omega_i, h(m_i)\}$.

ii. Verifies root node R' and verifies whether Equation $e(t, g) = e(h(R), g^{\alpha})$ holds, where α is the private key of the auditor. If it holds, then the validation passes. If the equation does not hold, the validation fails.

iii. Recompute root node R'' using $\{\Omega_i, h(m_i')\}$ and compare whether $R' = R''$ holds.

iv. If the validation passes, send the new label $t' = sig_{sk}(h(R'))$ to the CSS update on the cloud storage side.

3) Data Deletion. Assuming that the data block that the user wishes to delete is m_i, the cloud storage end first deletes that data block m_i from the server, and then removes the hash value $h(m_i)$ of its own data block from the hash value of that node in the authentication structure tree as in equation (16):

$$f(m_i) = h\left(f(m_{i1}) \| f(m_{i2}) \| \cdots \| f(m_{ip})\right) \tag{16}$$

Then update it to the root node in the authentication path through the hash value of all the parent nodes to complete the data deletion operation. The subsequent authentication process of data deletion and data modification operation is also largely the same, and will not be repeated here. As a result, the data authentication structure tree does not need to be reconstructed, and the computational overhead of the CSS on the cloud storage side can be reduced to a certain extent.

3.3.4. Bulk data block updates. The data authentication structure of this paper's scheme adopts the multi-branch path tree LBT structure, a parent node can connect multiple child nodes, so it can update the nodes corresponding to multiple data blocks at the same time, with the same complexity as a single data block update, the specific process of insertion update is as follows:

Suppose the user wants to insert a file F after data block m_i, and assume that file F can be divided into c small data blocks, $c < p$ where p is the out-degree of the multibranch path tree. If the file to be inserted is large and the number of chunks is greater than p, it can be divided into multiple sets containing p data chunks, and the insertion operation is performed sequentially.

1. After receiving the evidence of insertion from the Client on the user side, the TPA on the cloud auditing side will F chunk the file $F = (m_1', m_2', \ldots, m_c')$ and then will calculate the homomorphic signature of each chunk:

$$\sigma_i' \leftarrow \left(h(m_i') \cdot \prod_{j=1}^{s} u_j^{m_{ij}'} \right)^{\alpha} \tag{17}$$

After that, the update information $Update = (I, i, F)$ is sent to the CSS on the cloud storage side.

2. After receiving the insertion update request, the CSS on the cloud storage side runs the $Insert(\cdot)$ algorithm:

i. Saves the newly inserted data block and outputs a new file M'.

ii. Saves the signature σ_i' corresponding to the newly inserted data block and outputs a new collection of signatures Φ'.

iii. Adjust the nodes of the data authentication tree structure by storing the hash value $h(m_i')$ corresponding to the new data block $m_i'(i \in [1, c])$ as a child node under the node corresponding to data block m_i.

iv. Compute $f(m_i) = h\left(h(m_i) \| h(m_1') \| h(m_2') \| \cdots \| h(m_c')\right)$ and calculate and update the hash values of all the parent nodes it passes through in the authentication path to the root node and finally generate the new root node R'.

v. For the above update operation, the CSS on the cloud storage side returns to the TPA on the cloud auditing side an evidence $P_{update} = (\Omega_i, h(m_i), R')$ of the insertion operation.

3. After receiving evidence P_{update} from the CSS, the cloud audit side TPA runs the $VerifyUpdate(\cdot)$ algorithm and performs the following operations:

v. Calculate the root node R using $\{\Omega_i, h(m_i)\}$.

vi. Verify the root node R' and verify that Equation $e(t, g) = e(h(R), g^{\alpha})$ holds, where α is the private key of the auditor. If it holds, the validation passes. If the equation does not hold, the validation fails.

vii. Recompute the root node R'' using $\{\Omega_i, h(m_i), h(m_1'), h(m_2'), \cdots, h(m_c')\}$ and compare whether $R' = R''$ holds.

If the validation passes, send the new label $t' = sig_{sk}(h(R'))$ to the CSS update on the cloud storage side.

4. Theoretical and experimental analysis

4.1. Theoretical analysis

This paper defines the following notations to denote the computational operations and parameter sizes in the scheme: E_G denotes the power operation of a group element, M_G denotes the multiplication operation of a group element, $M_{Z_q^*}$ denotes the multiplication operation of a finite domain element, $A_{Z_q^*}$ denotes the addition operation of a finite domain element, $Pair$ denotes the bilinear mapping operation, $Rand$ denotes the generation of a random number operation, n denotes the total number of blocks of data, k denotes the capacity of a challenge set, l_1 denotes the number of bits required for a random number pair , l_2 denotes the number of bits required to

represent a time node in binary, l_3 denotes the number of bits required for a cluster element, $BitCoinReq$ denotes the number of bits required to get the block hash value generated by bitcoin at the current time, ft denotes the number of bits required for a file identifier, and $Param$ denotes the number of bits required for an elliptic curve parameter. The comparison of the theoretical overhead of the algorithm in this paper, THT-IVA algorithm and CP-IVA algorithm is shown in Table 1.

It can be seen that the computational overhead of the algorithm proposed in this paper is slightly larger than the control algorithm in the generation of the challenge phase and the evidence response phase, which is due to the fact that the algorithm proposed in this paper does privacy protection as well as control of the key time nodes in the verification process. In terms of communication overhead, this paper's algorithm is slightly larger than THT-IVA and CP-IVA algorithms. And in terms of storage overhead, this paper algorithm is similar to the control algorithm.

Table 1. Overhead Comparison

		Ours	THT-IVA	CP-IVA
Theoretical calculation overhead	Generate challenge stage	$2\cdot log_2^{l_2}+6\cdot E_G$ $+2\cdot M_G+c\cdot Rand$	$c\cdot Rand$	$BitCoinReq$ $+c\cdot Rand$
	Response phase	$(c+2)\cdot E_G$ $+c\cdot(M_{Z_q^*}+M_G)$ $+(c-1)\cdot A_{Z_q^*}$	$c\cdot(E_G+M_{Z_q^*})$ $+(c-1)\cdot(M_G+A_{Z_q^*})$	$c\cdot(E_G+M_{Z_q^*})$ $+(c-1)\cdot(M_G+A_{Z_q^*})$
Communication overhead		$c\cdot l_1+2\cdot l_2$ $+2\cdot log_2^{l_2}+3\cdot l_3$	$c\cdot l_1+2\cdot l_3+ft$	$c\cdot l_1+2\cdot l_3$
Storage overhead		$Param+l_3$	$Param+2\cdot l_3$	$Param+l_3$

4.2. Time Consumption and Challenge Success Rate Experiments

For the experiment, a host computer with 8-core CPU (3.2GHz) and 16GB RAM is selected as DO, an AliCloud ECS with dual-core CPU and 8GB RAM is selected as CSP, and a smartphone with Snapdragon 835 processor (up to 2.45GHz) is selected as DC.The overall algorithm, which is written in Java and using JPBC library, preprocesses a 120MB file into a number of 1KB data blocks, the total number of data blocks is 10^4 to 10^5, and the security parameter is 512 bit.If the corruption rate of data is at least 1%, the probability that the DC can detect CSP misbehavior by randomly selecting 450 data blocks is more than 99%, so the experiments in this paper only extract 450 data blocks each time for verification. The average of the test data from 20 experiments is taken as the final data. A-PDP as well as OPOR are compared and analyzed with the algorithm proposed in this paper.

4.2.1. Time of evidence calculation. The evidence computation time refers to the time required by the CSP to generate evidence in the evidence response step, and its experimental results are shown in Fig. 1.The evidence computation time of A-PDP and OPOR is basically the same, while the evidence computation time of this paper's algorithm is slightly higher than that of A-PDP and OPOR.This is because A-PDP and OPOR only need to generate the labeled evidence and the data evidence in this phase, and this paper's algorithm needs to additionally perform the secret comparison to ensure the validity of the verification request. Considering the powerful computing capability of cloud service providers in real cloud storage environments, the difference in evidence computation time can be compensated by distributed computing.

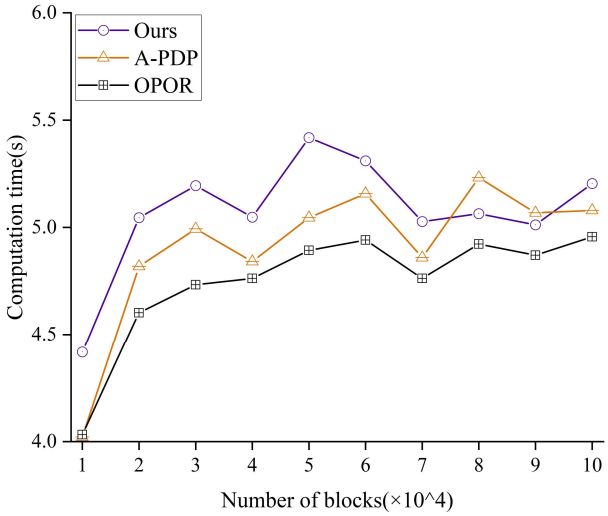

Fig. 1. Proof computation time

4.2.2. Challenge transmission time and evidence transmission time. The challenge transmission time is the time required for the challenge set sent by the DC to be transmitted in the network, and the evidence transmission time is the time required for the evidence sent by the CSP to be transmitted in the network, and the experimental results are shown in Fig. 2. The experimentally measured result is between 9ms-12ms, while the calculated result is 3.3ms by the challenge time limit formula $T = t_c + \alpha \cdot \Delta t_c + C_p + \beta \cdot \Delta t_c + C_v$. In order to solve this kind of network transmission problem, this paper adds a certain amount of redundancy to the transmission time in the challenge time limit. According to the gap between the experimental results and the calculation of the above formula, this paper sets α and β between 5-6 to ensure that the validation process can be carried out effectively.

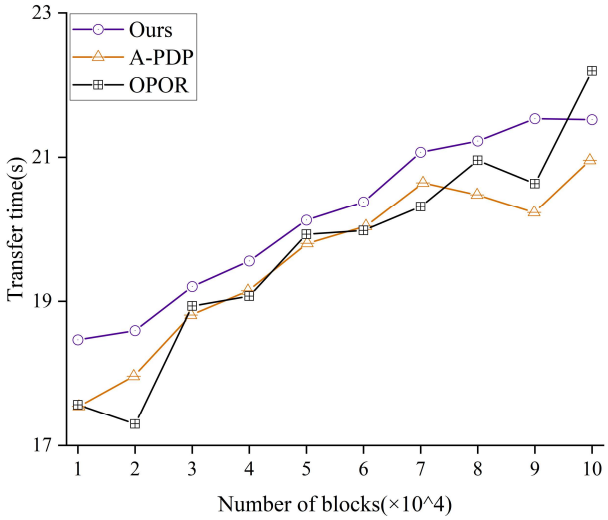

Fig. 2. Challenge and proof transmission time

4.2.3. Time to judge results. The result judgment time refers to the time required for DC to execute the result judgment step, and the experimental results are shown in Fig. 3. The result judgment times of A-PDP and OPOR are roughly the same, and both are slightly lower than that of this paper's algorithm. This is because A-PDP and OPOR only need to validate the evidence at this stage, while this paper's algorithm needs to additionally generate the result judgment moment t_v and the commitment value for validation result res to prevent CSP tampering.

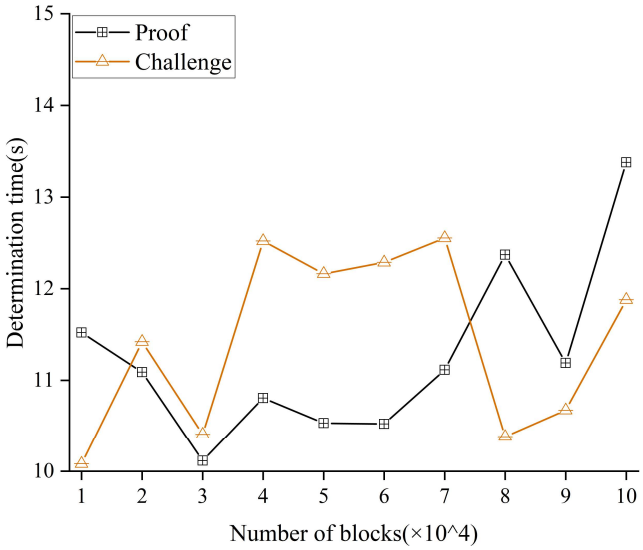

Fig. 3. Result determination time

4.2.4. Challenging implementation time. The challenge execution time is the time from the DC initiating the verification challenge to the DC executing and completing the result judgment, and the experimental results are shown in Fig. 4. It can be seen that the challenge execution time of A-PDP and OPOR is basically the same, while the challenge execution time of this paper's algorithm is slightly larger than that of A-PDP.This is because this paper's algorithm adds a series of commitment values as well as secret comparisons during the verification process to ensure the timeliness of verification.

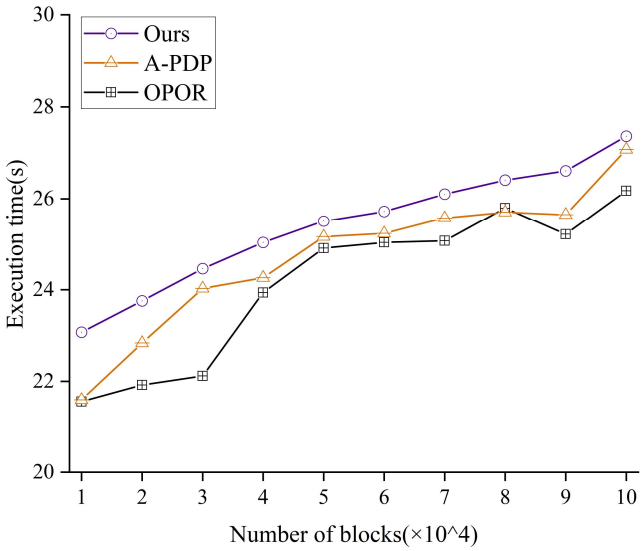

Fig. 4. Challenge execution time

4.2.5. Challenge success rate. The challenge success rate is the ratio of the number of successfully executed challenges to the total number of initiated challenges. The maximum request response tolerance time for mobile devices is set to 3.5 seconds, and the experimental results are shown in Fig. 5. It can be seen that the challenge success rate of A-PDP and OPOR is the same, while the challenge success rate of this paper's algorithm is significantly higher than that of A-PDP and OPOR after the number of challenged data blocks is more than 300.The challenge failure of A-PDP and OPOR is due to the fact that the evidential response time of the challenges with more than 300 data blocks exceeds the maximum request response tolerance time of the mobile device. The algorithm in this paper

introduces a challenge time limit and a critical time node to control the data validation and intercepts the meaningless challenges, thus ensuring the successful execution of the validation.

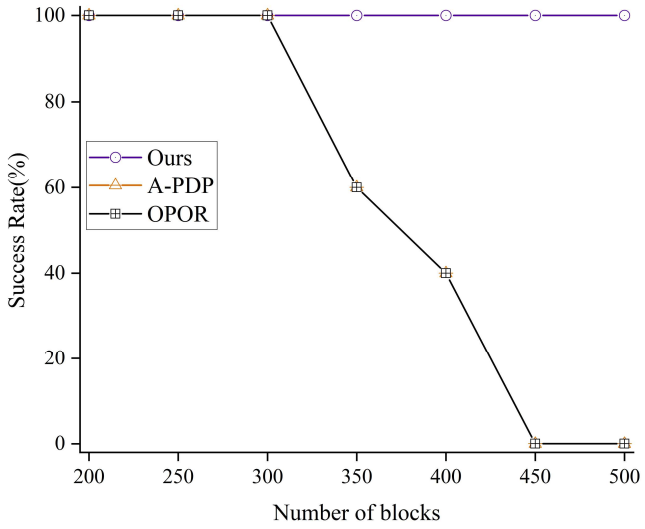

Fig. 5. Challenge success rate

4.3. Overhead and verification efficiency experiments

4.3.1. Computational overhead. The computational overhead is the time required for TPA to return the validation results from the beginning of sampling, and the experimental results are shown in Fig. 6. Fig. 6 shows the total computational overhead required to complete a validation, and according to the experimental results, it can be seen that the computational overhead of this paper's algorithm is higher than that of G-PDP and is linearly related to the total number of data blocks. This is due to the fact that this paper's algorithm needs to calculate the integrity probability speculative value of each data block through convolutional computation before establishing the hierarchical model, and then establishes the hierarchical model of the overall data according to the speculative value, thus increasing the corresponding computational overhead. While G-PDP adopts random sampling, the computational overhead mainly depends on the overhead of generating random numbers, and the computational overhead is independent of the total number of data blocks. However, the total computational overhead of both the algorithm in this paper and G-PDP is less than 0.8 seconds, so it is negligible. In addition, when the total number of data blocks is large, both Conv and Stratification phases can be executed in parallel, so distributed techniques or parallel computing can be used to reduce the waiting time of the corresponding phases.

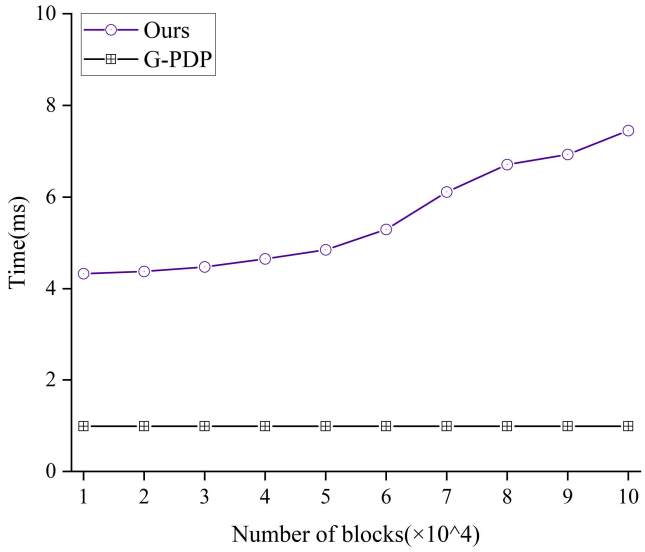

Fig. 6. Total overhead

4.3.2. Communication overhead. The communication overhead is the total amount of data that needs to be transmitted for the TPA to interact with the CSP during the verification process, and the experimental results are shown in Fig. 7. The curves of this paper's algorithm overlap with those of G-PDP, i.e., the two algorithms require equal communication overhead. This is due to the fact that the hierarchical model produced by this paper's algorithm only needs to be stored in the TPA, and only the set of challenges required for validation, i.e., the set of binary groups consisting of the index number of the data block as well as the corresponding random number, needs to be transmitted during the communication process. Also the communication overhead is the same as in G-PDP since the number of data blocks per challenge is equal.

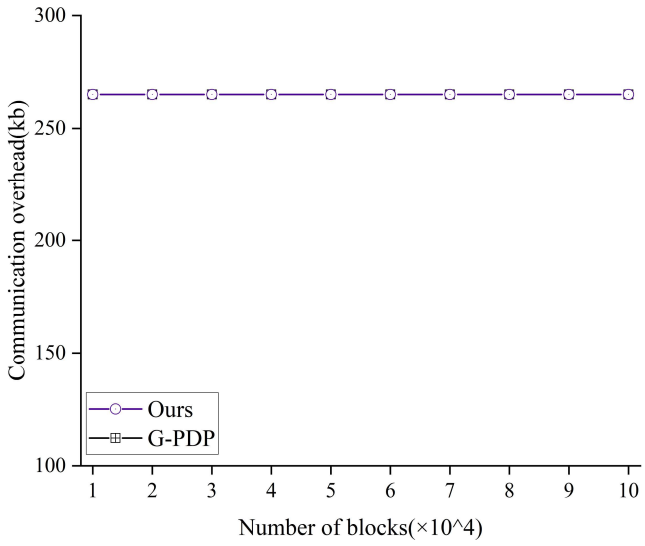

Fig. 7. Communication overhead

4.3.3. Storage overhead. The storage overhead is the storage space occupied by the verification auxiliary information that needs to be stored by the TPA and CSP during the verification process. The experimental results are shown in Fig. 8, the storage overhead of this paper's algorithm is higher than that of G-PDP. This is due to the fact that the TPA in GPDP only needs to store the set of challenges required for validation and the validation evidence returned by the CSP, and the size of the set of challenges is only linearly related to the number of data blocks per validation. Since only 450 data

blocks are extracted for each validation, the storage overhead is constant. In contrast, the algorithm in this paper needs to store the hierarchical model on top of that, and the storage overhead is higher than that of G-PDP because the size of the hierarchical model is linearly related to the total amount of data. However, each data in the hierarchical model only represents the integrity probability speculation value of a data block, which can be stored using a certain byte-length integer when higher accuracy is not required, thus avoiding higher storage overhead due to the use of floating-point storage. In addition, the size of each data block is 1024bit, which requires 32bit when using floating point numbers to store the integrity probability speculative values, increasing the storage overhead by only 3.225%. When an 8bit integer is used to store the corresponding speculative value, the storage overhead increases by only 0.78485% and is thus negligible.

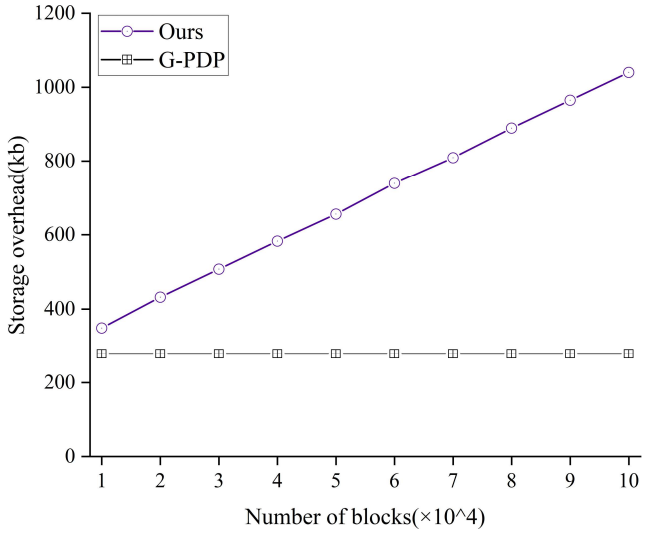

Fig. 8. Storage overhead

4.3.4. Validation efficiency. The verification efficiency is the average number of verification times required to detect all corrupted data blocks when other conditions are the same and the total number of data blocks increases from 10^4 to 10^5. The experimental results are shown in Fig. 9, the verification ability of this paper's algorithm is better than that of G-PDP. The reason for this is that during the multiple verification process this paper's algorithm will stratify the currently unverified data blocks according to the integrity probability speculative value, and at the same time, it will give different weight values according to the speculative value from low to high. In the next round of sampling process, the probability of data blocks with lower presumptive values being extracted is higher, so that each verification has a higher probability of verifying corrupted data blocks, thus reducing the number of verifications required to verify all corrupted data blocks and improving the verification efficiency. In contrast, G-PDP uses random sampling for each round of sampling, and the probability of a corrupted data block and an intact data block being sampled is the same, so that multiple rounds of verification process will generate a large number of repetitive verifications, which reduces the verification efficiency.

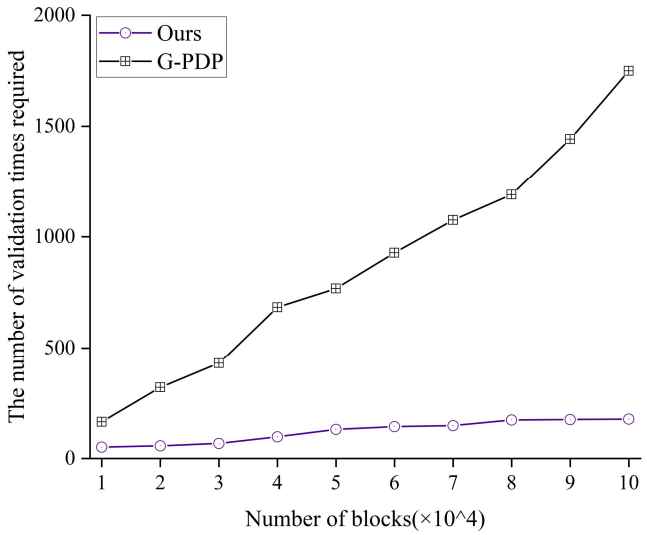

Fig. 9. Validation efficiency results

In summary, the algorithm proposed in this paper constructs the hierarchical model required for the current verification through the verification results of the previous round in the stage of generating the challenge set and dynamically adjusts the sampling ratio of different integrity probabilities, so as to improve the probability that the corrupted data blocks are extracted. Although the algorithm increases the computational overhead and storage overhead in a single verification process, it greatly reduces the number of verifications required to detect corrupted data blocks and improves the verification efficiency.

5. Conclusion

In this paper, we design an optimized cloud data storage integrity verification algorithm scheme based on multi-branch path tree LBT to improve the efficiency and security of accounting cloud data transmission. The theoretical computation overhead and communication overhead of this paper's algorithm are slightly larger than those of THT-IVA and CP-IVA algorithms, while the theoretical storage overhead is similar to that of existing algorithms. In the designed experiments, the evidence computation, challenge transmission, evidence transmission and challenge execution times of this paper's algorithm are slightly higher than those of the comparison algorithms, but the difference in the time consumed is not significant, and more reliable security is guaranteed. And the challenge success rate of this algorithm can still be maintained at a high level after more than 300 challenge data blocks. The total computational overhead of this paper's algorithm is below the 0.8 second level, while the verification efficiency is better than that of the G-PDP algorithm. In summary, this paper's algorithm achieves higher cloud data integrity verification efficiency with a slight increase in computational overhead, and is a reasonable data integrity verification algorithm for accounting information.

Funding

This research is supported by National Social Science Foundation of China (Grant No. 21BGL040).

References

[1] GuQian, L. Y. (2021). Exploring the Focus of Enterprise Informatization During Digital Transformation. *In E3S Web of Conferences* (Vol. 251, p. 03070). EDP Sciences.

[2] Dai, Q. (2022). Designing an accounting information management system using big data and cloud technology. *Scientific Programming*, 2022(1), 7931328.

[3] Wu, X. (2021, August). Application and Thinking of Cloud Accounting in Accounting Informatization. *In Journal of Physics: Conference Series* (Vol. 1992, No. 3, p. 032109). IOP Publishing.

[4] Meng, L. (2022). The Promotion Effect of the Improved ISCA Model on the Application of Accounting Informatization in Small‐and Medium‐Sized Enterprises in the Cloud Computing Environment. *Mobile Information Systems*, 2022(1), 4228178.

[5] Li, F., & Fang, G. (2022). Process‐Aware Accounting Information System Based on Business Process Management. *Wireless Communications and Mobile Computing*, 2022(1), 7266164.

[6] Wu, Q. F. (2021). Distance teaching method of accounting informatization course based on big data mining. *In e-Learning, e-Education, and Online Training: 7th EAI International Conference, eLEOT 2021, Xinxiang, China, June 20-21, 2021, Proceedings Part I 7* (pp. 155-166). Springer International Publishing.

[7] Yang, X. (2024). Optimizing Accounting Informatization through Simultaneous Multi-Tasking across Edge and Cloud Devices using Hybrid Machine Learning Models. *Journal of Grid Computing*, 22(1), 12.

[8] Sharma, P., Jindal, R., & Borah, M. D. (2020). Blockchain technology for cloud storage: A systematic literature review. *ACM Computing Surveys* (CSUR), 53(4), 1-32.

[9] Wang, R. (2017). Research on data security technology based on cloud storage. *Procedia engineering*, 174, 1340-1355.

[10] Chen, Y., Li, L., & Chen, Z. (2017, December). An approach to verifying data integrity for cloud storage. *In 2017 13th International Conference on Computational Intelligence and Security* (CIS) (pp. 582-585). IEEE.

[11] Roslin, S. E. (2021). Data validation and integrity verification for trust based data aggregation protocol in WSN. *Microprocessors and Microsystems*, 80, 103354.

[12] Lin, C., Shen, Z., Chen, Q., & Sheldon, F. T. (2017). A data integrity verification scheme in mobile cloud computing. *journal of Network and Computer Applications*, 77, 146-151.

[13] Xie, G., Liu, Y., Xin, G., & Yang, Q. (2021). Blockchain‐Based Cloud Data Integrity Verification Scheme with High Efficiency. *Security and Communication Networks*, 2021(1), 9921209.

[14] Zhao, Y., Qu, Y., Xiang, Y., Uddin, M. P., Peng, D., & Gao, L. (2024). A comprehensive survey on edge data integrity verification: Fundamentals and future trends. *ACM Computing Surveys*, 57(1), 1-34.

[15] Zhang, Y., Geng, H., Su, L., & Lu, L. (2022). A blockchain-based efficient data integrity verification scheme in multi-cloud storage. *Ieee Access*, 10, 105920-105929.

[16] Cheng, L., Liu, F., & Yao, D. (2017). Enterprise data breach: causes, challenges, prevention, and future directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(5), e1211.

[17] Gaidarski, I., & Kutinchev, P. (2019, November). Using big data for data leak prevention. *In 2019 Big Data, Knowledge and Control Systems Engineering* (BdKCSE) (pp. 1-5). IEEE.

[18] Wang, H., & Zhang, J. (2019). Blockchain based data integrity verification for large-scale IoT data. *IEEE Access*, 7, 164996-165006.

[19] Gao, J. (2022). Analysis of enterprise financial accounting information management from the perspective of big data. *International Journal of Science and Research* (IJSR), 11(5), 1272-1276.

[20] Huang, N. (2016, August). Discussion on the Application of Cloud Accounting in Enterprise Accounting Informatization. *In 2016 International Conference on Economics, Social Science, Arts, Education and Management Engineering* (pp. 136-139). Atlantis Press.

[21] Li, M., Wei, W., Wang, J., & Qi, X. (2018). Approach to evaluating accounting informatization based on entropy in intuitionistic fuzzy environment. *Entropy*, 20(6), 476.

[22] Ma, Y. M., Huang, J. Y., Danarson, J. H., Huang, Y., & Wei, X. H. (2023, February). Problems and Countermeasures Facing Accounting Informatization in the Era of Big Data. *In Proceedings of the 2023 15th International Conference on Machine Learning and Computing* (pp. 217-220).

[23] Zhao, J., Zhang, L., & Zhao, Y. (2022). Informatization of Accounting Systems in Small‐and Medium‐Sized Enterprises Based on Artificial Intelligence‐Enabled Cloud Computing. *Computational Intelligence and Neuroscience*, 2022(1), 6089195.

[24] Zhang, X. (2021). Application of data mining and machine learning in management accounting information system. *Journal of Applied Science and Engineering*, 24(5), 813-820.

[25] Wang, J. (2022, March). Influence of Big Data on Manufacturing Accounting Informatization. *In The International Conference on Cyber Security Intelligence and Analytics* (pp. 695-700). Cham: Springer International Publishing.

[26] Zhou, L., Fu, A., Yu, S., Su, M., & Kuang, B. (2018). Data integrity verification of the outsourced big data in the cloud environment: A survey. *Journal of Network and Computer Applications*, 122, 1-15.

[27] Saxena, R., & Dey, S. (2016). Cloud audit: A data integrity verification approach for cloud computing. *Procedia Computer Science*, 89, 142-151.

[28] Tian, J., & Jing, X. (2020). Cloud data integrity verification scheme for associated tags. *Computers & Security*, 95, 101847.

[29] Zhu, H., Yuan, Y., Chen, Y., Zha, Y., Xi, W., Jia, B., & Xin, Y. (2019). A secure and efficient data integrity verification scheme for cloud-IoT based on short signature. *IEEE Access*, 7, 90036-90044.

[30] Fan, Y., Lin, X., Tan, G., Zhang, Y., Dong, W., & Lei, J. (2019). One secure data integrity verification scheme for cloud storage. *Future Generation Computer Systems*, 96, 376-385.

[31] Ping, Y., Zhan, Y., Lu, K., & Wang, B. (2020). Public data integrity verification scheme for secure cloud storage. *Information*, 11(9), 409.

[32] Witanto, E. N., Stanley, B., & Lee, S. G. (2023). Distributed data integrity verification scheme in multi-cloud environment. *Sensors*, 23(3), 1623.

[33] Mais Nijim & Divya Ballampalli. (2022). The design of a novel smart home control system using a smart grid based on edge and cloud computing. International Journal of Smart Grid and Clean Energy (SGCE)(2),57-71.

[34] Yan Fang Niu. (2011). The Analysis of Accounting Information Activities Blending SOA. Advanced Materials Research(267-267),35-38.

[35] Haoran Shi,Zehua Chen,Yongqiang Cheng,Xiaofeng Liu & Qianqian Wang. (2024). PB-Raft: A Byzantine fault tolerance consensus algorithm based on weighted PageRank and BLS threshold signature. Peer-to-Peer Networking and Applications(1),1-16.

[36] Walker Ieuan,Hewage Chaminda & Jayal Ambikesh. (2021). Provable Data Possession (PDP) and Proofs of Retrievability (POR) of Current Big User Data.SN Computer Science(1).