Hindawi
Security and Communication Networks
Volume 2022, Article ID 8508755, 21 pages
https://doi.org/10.1155/2022/8508755

Research Article

Ldasip: A Lightweight Dynamic Audit Approach for Sensitive Information Protection in Cloud Storage

Li Lin,[1,2] WenTing Tan ,[1] and ZhenXing Chu[1]

[1]*College of Computer Science Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China*
[2]*Beijing Key Laboratory of Trusted Computing, Beijing 100124, China*

Correspondence should be addressed to WenTing Tan; 344842529@qq.com

Received 11 November 2021; Revised 15 April 2022; Accepted 12 May 2022; Published 20 June 2022

Academic Editor: Wenxiu Ding

How to audit the integrity of data stored on the cloud with incomplete trust is an important problem that restricts the development of cloud storage. Although there are several data integrity audit schemes in cloud storage, the increased need to protect sensitive information and support large-scale data storage and dynamic update will result in a significant increase in audit cost, which seriously affects the efficiency of existing cloud audit systems. To solve this problem, we propose Ldasip, a lightweight dynamic auditing method that supports sensitive information protection in cloud storage. Exploiting identity-based data integrity audit, a data masking technology is introduced into to protect user's sensitive information. At the same time, an improved multibranch tree structure is proposed to realize dynamic audit and reduce communication overhead in the verification process. Theoretical analysis and comprehensive experiments have been conducted, which demonstrate the effectiveness of Ldasip. The results show that Ldasip can ensure the correctness of the audit, protect the sensitive information in the user's stored content, and support the dynamic update of data with less audit time and communication overhead.

1. Introduction

With the rapid development of information technology and network technology, user data are growing explosively. The emergence of cloud services addresses the limitations of computation and storing large amounts of data locally. However, when the data are outsourced and stored in the cloud, users lose absolute control over the data, and the cloud data may be tampered or destroyed by attackers or cloud service providers intentionally or unintentionally. For example, in 2020, about 400 GB of data was downloaded from a UN cloud server in Europe by an intruder. The personal information of more than 4,000 UN staff members was compromised [1]. In 2021, more than 200 million pieces of user data stored on Sina servers, including the personal data of 7.3 million Chinese citizens, were stolen and made public by hackers [2]. In 2022, data from a cloud server of a Croatian telephone operator was downloaded by an intruder. The personal information of more than 200,000 people was compromised [3]. These events greatly reduce users' trust in cloud services and restrict the development and promotion of cloud storage. Therefore, how to maintain the security of cloud storage data is one of the important problems to be solved in cloud storage.

Security audit is an important approach to ensure data integrity. The existing cloud environment data integrity audit schemes are generally divided into two categories: private audit and third-party audit. Private audit can only be performed by users, which is more efficient, but requires users to be responsible for data signing, audit verification, and other calculation and maintenance of a large amount of information [4]. In third-party audit [5], data integrity verification is completed by a trusted third party and the verification report is sent to the user, which greatly reduces the computation cost of the user. Currently, more and more experts and scholars at home and abroad are paying attention to third-party audit. The existing work has put forward different solutions from three aspects: protection of sensitive information, reduction of audit cost, and dynamic update of data.

There are two main types of threats concerned in the existing methods of protecting sensitive information. First, third-party auditors are usually honest and curious. When they perform data integrity verification on behalf of users, they may push back the original data through the audit proof returned by the cloud service provider, resulting in user data leakage [6]. Second, cloud service provider (CSP) is not trustworthy, and he may leak user data to malicious users for his own interests. Due to the introduction of third-party auditors, there is communication overhead between the user and the third-party auditor. Meanwhile, in order to facilitate the validation, the auditor needs to maintain a large amount of validation metadata, state, secret key information, etc., which itself incurs additional computation and storage overhead.

In addition, for third-party audit, most data integrity verification structures rely on the public key infrastructure, where users need to generate and manage public key certificates and auditors need to validate them. In order to simplify certificate management, Wang et al. [7] proposed an identity-based integrity verification scheme based on bilinear pairings. In this scheme, a trusted key generator was introduced to generate the private key that can sign the user data, and the user's ID number, e-mail address, or name was directly regarded as the public key, which could eliminate the cost of generating and managing the user's certificate and the cost of verifying the proof by the third-party auditor. However, most current identity-based audit schemes did not take into account the scenario when a user updates data in the cloud. Supporting dynamic data manipulation means allowing users to insert, delete, and modify data files during the integrity audit of user data. Guo et al. [8] put forward an identity-based dynamic integrity audit scheme. However, this scheme adopted the Merkle Hash Tree authentication structure, with the increase in data blocks, the authentication process, and authentication data need too much auxiliary information, so it took a long time for the cloud service provider to query the data blocks in the verification and update processes, which brought extra computation and communication overhead to the cloud service provider.

To solve the problem, we proposed Ldasip, a lightweight dynamic audit method that supports sensitive information protection in cloud storage. Exploiting identity-based audit model, this method can avoid the complex certificate management work such as the issue, management, and revocation of public key certificate. At the same time, a data masking technology is introduced into the proof generation algorithm executed by the cloud service provider to prevent the third party from pushing out the original data through the audit proof and to realize the protection of sensitive information of user data. In addition, an improved multibranch tree structure is proposed. By selecting the deputy root node to store information, the length of authentication path is shortened. The locality principle is adopted to reduce the data block query time, thus improving the efficiency of third-party audit, and reducing the communication overhead in the verification process while realizing dynamic audit.

Compared with existing work, our contributions are summarized as follows.

(1) We proposed a new cloud service node proof generation algorithm by introducing data masking technology, which can prevent third parties from inferring the user's original data based on the audit proof after multiple challenges. A generation and verification mechanism of third-party legal authority is proposed to ensure that only the legitimate third-party authorization by the user can verify the files on behalf of the user and reduce the security threat brought by the third-party auditor.

(2) We proposed an improved multibranch authentication tree structure. The node utilization rate is improved by storing data block information on the nonleaf nodes of the authentication tree, and the deputy root node is selected to describe the integrity of the node and all its descendants. In this way, the cloud service provider does not need to traverse the whole authentication tree when conducting data block query, which shortens the authentication path length and improves the efficiency of user signature while supporting the dynamic update of data.

(3) We conducted theoretical analysis and experimental evaluation of the proposed scheme. The results show that Ldasip supports dynamic audit and sensitive information protection, and has lightweight challenge response and integrity verification cost, compared with the existing classical schemes such as literature [9, 10].

The rest of this paper is organized as follows. This section presents related work, describes the system model, introduces Ldasip in detail, provides security theoretical analysis of Ldasip, presents our experimental results, concludes this paper, and shows some possible future work.

2. Related Work

In cloud storage, data outsourced in the cloud by users are faced with external attacks and internal threats, and data integrity audit is an important solution [11]. There are several typical approaches to data integrity audit. The following analysis and comparison are made from three aspects: audit performance optimization, support sensitive information protection, and support data dynamic operation.

In the process of data integrity audit, the protection of sensitive information of user data is mainly threatened by the untrustworthiness of cloud service providers and third-party auditors. Ateniese et al. [12] put forward the concept of provable data possessing verification scheme based on homomorphic verification tag and random sampling strategy for the first time. However, the computation and communication overhead of this scheme were high, and the protection of sensitive information of user data was not considered. Shah et al. [13] proposed to encrypt users' data and calculate hash value based on symmetric encryption and send them to auditors. In this scheme, auditors need to verify whether the server had the previously promised decryption key, and the scheme was only applicable to encrypt files and

users need to redownload their data from the cloud to the local area, which increased the computation cost of the audit process. Wang et al. [14] proposed a privacy-protected remote data integrity audit scheme using random masking technology, but this scheme was not applicable to identity-based integrity audit scenarios, and there was no way to prove the authenticity of audit proofs sent by cloud service providers. In order to resist the attack of quantum computer, Tan et al. [15] proposed an audit scheme based on lattice to construct a random mask to cover up the audit proof, so that user data could be protected from the attack of curious third parties. Wang [16] et al. proposed a data integrity audit method based on Hash Message Authentication Code (HMAC) and indistinguishability confusion, which supported the protection of sensitive information of user data. However, this method required the user to manage the certificate, which had a large computation and storage overhead. Therefore, Han et al. [17] put forward a distributed data integrity audit scheme based on blockchain, which could resist various attacks in the integrity audit process by using blockchain system with decentralized, tamper-proof, and traceability characteristics. However, this method did not support data update.

Early data integrity verification schemes focus on ensuring the integrity of static data. When users perform dynamic operations such as adding, deleting, and modifying files, they need to download the files to the local area for update and then upload them to the cloud. This process will incur a lot of computation and communication overhead. Erway et al. [18] proposed the authentication dictionary based on Grade-based authentication dictionary (DPDP-I) and RSA tree-based authentication dictionary (DPDP-II) to construct the dynamic data audit scheme; however, when the cloud service provider updated the data, the update of the underlying nodes in the skip table would lead to a lot of computation overhead, and this structure would have a larger length when the number of data blocks was large, which would lead to an increase in the amount of auxiliary authentication information. Wang et al. [19] proposed a dynamic audit method based on the classic Merck Hash Tree Block Label Authentication and introduced bilinear aggregate signature technology to support batch audit, but the scheme did not consider the protection of sensitive user data. Daniel et al. [20] proposed to build a data structure based on file hash values to realize dynamic audit. Jian Shen et al. [21] put forward a new dynamic structure composed of double link information table and position array to realize dynamic audit. Sookhak et al. [22] put forward a dynamic audit scheme based on divide-and-conquer table, which divided the data structure into k numbers to reduce its size and reduced the computation cost of users when updating. However, the scheme did not consider the protection of sensitive information of user data. T. Shang et al. [23] proposed a data structure of Merkle hash tree for block tag authentication, allowing users to insert data after each data block, which could effectively improve the efficiency of dynamic integrity audit. Yuan et al. [24] proposed a modified index hash table (MIHT) structure, which can effectively realize data dynamics.

Existing performance optimization efforts focus on reducing the computation and storage overhead and communication complexity of users, third-party auditors, and cloud service providers in the data integrity audit process. Ateniese [25] et al. proposed a Proof of Data Possession Protocol (E-PDP) to reduce the computation and communication cost of the auditor through random sampling of data blocks. However, this scheme did not consider the huge computation cost of the cloud server. Sookhak et al. [22] proposed a data integrity audit method based on file compression and improved algebraic signature, which could reduce the computation and communication overhead between the user and the cloud service provider. Third party-based data integrity audit structures mostly rely on public key infrastructure. Users need to generate and manage public key certificates, and third-party auditors need to maintain a large number of verified metadata, state, key information, and verify certificates. Therefore, Wang et al. [7] proposed an identity-based integrity audit scheme based on bilinear pairs to simplify key and certificate management. A trusted key generator was introduced into the original tripartite interaction and was responsible for generating the private key for the user to sign the data. The user's ID number, e-mail address, or name and other information were treated as the public key, so there was no need for the user to generate and manage the certificate, and third-party auditors to verify the certificate, which reduced the computation overhead of the user and third-party auditors. However, this method was only suitable for small-scale users. For this reason, Zhang et al. [26] proposed an identity-based audit scheme by introducing a hierarchical private key generator suitable for large-scale user groups, which improved the audit efficiency of large-scale users. Shen et al. [27] proposed a data integrity audit scheme without storing private keys in which biometric data (such as iris scan and fingerprint) were used as the fuzzy private key of the user to avoid using hardware token, which improved the security and efficiency of audit. However, these schemes did not support the protection of sensitive information of user data and dynamic update.

3. System Model

In identity-based cloud data integrity audit system, the public and private keys are generated based on the user's identity, and there is a third party responsible for verifying the integrity and availability of the data stored in the cloud [7]. As shown in Figure 1, there are four entities in the identity-based data integrity audit system: user, third-party auditor (TPA), cloud service provider, and private key generator (PKG). Among them, users are entities that outsource data stored in a cloud server. Cloud service providers are entities that have powerful storage resources and provide storage services for users. A third-party auditor is a trusted entity with professional data audit capability that performs data integrity verification on behalf of users. PKG is a trusted entity that generates parameters for the system and private keys for users.

As shown in Figure 1, the identity-based data integrity audit process is as follows:

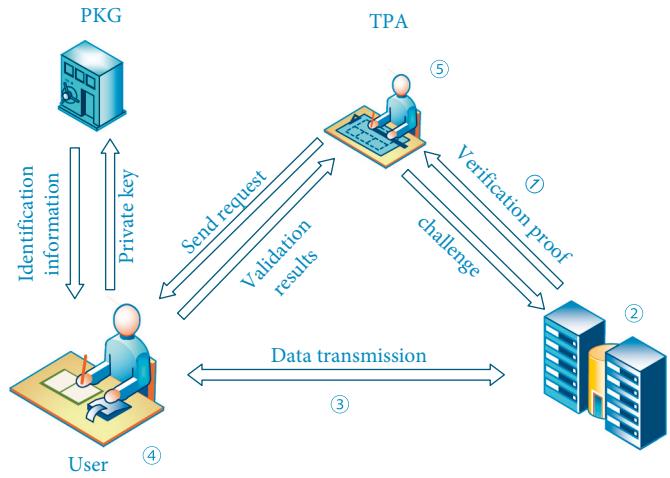

Figure 1: Identity-based data integrity audit model.

(1) The user sends his identity information (ID) to the PKG. The PKG calculates the private key for the user according to the user's ID and sends it to the user.

(2) The user uses his private key to generate data tags that will be used to verify the integrity of the data blocks, and uploads all data blocks and his corresponding tags to the cloud.

(3) To verify the integrity of data stored in the cloud, the user authorizes a TPA to send an audit challenge to the cloud.

(4) When CSP receives an audit challenge from TPA, CSP generates the audit proof based on the audit challenge and user data blocks stored in the cloud and sends it to TPA.

(5) When TPA gets the audit proof, it will judge whether the user's data stored in the cloud are complete based on the correctness verification of the audit proof, and sends the verification result to the user.

Identity-based data integrity audit still faces the following security risks and performance problems.

During the audit process, data privacy is mainly threatened by the following two aspects. As shown in Figure 1, firstly, the third-party auditor is not completely trustworthy, and it is possible to deduce the user's original data when verifying the audit certificate out of curiosity in step ①, thus causing the leakage of user data. In addition, cloud service providers lack credibility, and data stored by users may be damaged or missing due to external attacks or adverse effects of the cloud in step ②.

The audit process also faces performance problems in the dynamic update and integrity verification stage. When users update cloud data, they can only download the whole file locally to update the data, which leads to the calculation and communication overhead in step ③. In addition, data integrity audit meets the requirements of users to verify the integrity of cloud data at any time, but long-term integrity audit will bring bad user experience. For example, it takes too much time in the file signature generation process in step ④ and integrity verification process in step ⑤, which seriously affects the audit performance.

Under the framework mentioned above, this paper proposed a lightweight integrity dynamic audit approach that supported sensitive data protection. The approach has the following objectives:

(1) Ensure the correctness of the private key. When PKG sends the correct private key to the user, the private key must be verified by the user.

(2) Ensure the correctness of TPA audit authorization. Only the TPA authorized by the user can get a certified reply from CSP.

(3) Ensure the correctness of the verification process [28]. The valid proof produced by the proof generation algorithm passes the verification algorithm with overwhelming probability. In other words, Ldasip can ensure that in the verification process, if both TPA and CSP are trusted and the data files are stored correctly, then the audit proof generated by CSP based on the challenge information must pass the verification successfully.

(4) Support sensitive data protection. User identity and data content are not disclosed to TPA during the audit process.

(5) Ensure the integrity of storage [29]. A CSP without user data cannot provide a valid audit proof.

4. Design of Ldasip

In this section, we will give the design of the proposed Ldasip approach, including its working principles and detailed core functions.

4.1. Working Principle. In Ldasip, there are different function modules deployed in users, PKG, TPA, and CSP, respectively. The architecture of Ldasip is shown in Figure 2, where PKG is responsible for system initialization and key generation. Users perform third-party legal authority

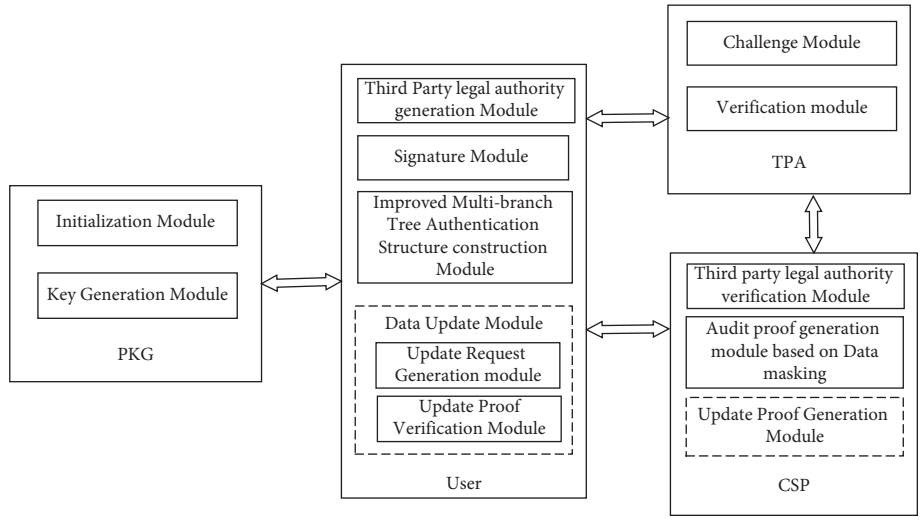

Figure 2: The architecture of Ldasip.

generation, signature generation, improved multibranch tree authentication structure construction, update request generation, and update proof verification. TPA performs challenge and verification model. CSP executes third-party legal authority verification, audit proof generation, and update proof generation module.

System initialization module is executed to generate relevant system parameters for initializing data integrity audit by PKG. It is realized by $Setup(1^{\lambda}) \longrightarrow (PP, msk)$. It inputs a security parameter and generates a public parameter and a master key according to bilinear mapping, which can be used for the subsequent data integrity verification.

Key generation module is executed on PKG [30], which generates the corresponding private key for the user according to the identity information provided by the user. It is realized by $KeyExtract\ (PP, msk, ID) \longrightarrow skID$. It inputs public parameters, master key, and user ID, and outputs the private key corresponding to the user ID. In this process, the user sends the ID to PKG, and PKG calculates the user's private key and sends it to the user through a secure channel. When the user receives the private key sent by PKG, it verifies whether the private key is valid through formula (1) (details are shown in Table 1), and receives the private key if the equation holds, otherwise, discards it.

Third-party legal authority generation module is executed by a user. The user authorizes TPA to perform data integrity audit instead of himself and generates legal authority for the authorized TPA and legal authority verification value for CSP according to the TPA's identity information and the user's identity information, so that the legitimacy of TPA can be verified during the audit process. The function is realized by $\text{Entrust}(ID, ID_{TPA}) \longrightarrow \text{Entrust}$. The inputs of this function are user ID and TPA's ID, and the output is the corresponding legal authority. To prevent malicious attackers from launching denial-of-service attacks on CSP, it is stipulated that only TPA authorized by users can launch integrity audit challenge. The user generates legal authority for TPA. In Ldasip, the user generates authorization, calculates legal authority verification value, sends the legal authority to TPA, and then sends the legal authority verification value to CSP.

Improved multibranch tree authentication structure construction module is executed on a user, who constructs the improved multibranch tree authentication structure, and sends authentication structure and signature together to the CSP. The function can be realized by $Construct(m_i) \longrightarrow C$. It inputs data blocks and outputs the authentication structure. The details of the algorithm are described in the following subsections.

Signature module is executed on the user, which generates signature of the data blocks and the signature of the root node and the deputy root nodes of the multibranch authentication tree. The function is realized by $Sign\ (m_i, skID) \longrightarrow (\sigma_i, \Gamma, \gamma)$. The inputs of this function are user's private key and data blocks, and the outputs are file authentication tag and authentication structure signatures. The user generates an authentication tag for each data block of the file by hash operation, calculates the root node and the deputy root node by user's private key, sends the data blocks and signatures to the cloud server, and then deletes the local data blocks.

Challenge module is executed on a TPA to generate audit challenge. The function is realized by $Challenge\ (PP, ID) \longrightarrow chal$, where inputs public parameters and user ID, and outputs challenge *chal*. TPA generates a challenge *chal* and sends it to the cloud. In this process, TPA randomly selects a set containing multiple elements to form a challenge *chal*. TPA sends the challenge *chal* and legal authority (user ID, TPA ID, and corresponding legal authority) to the cloud.

Third-party legal authority verification module is executed on CSP to verify the validity of TPA authorized by users. It is realized by $EntrustV(PP, Entrust, V, ID, ID_{TPA}) \longrightarrow \{0,1\}$. The inputs of the algorithm are public parameter, legal authority, legal authority verification value, user ID, and TPA ID, and the output is 0 or 1. CSP verifies whether the TPA is legal through formula (2). CSP considers the TPA legal and executes the audit proof generation algorithm if it holds; otherwise, it terminates the process.

Table 1: Symbol description table.

Symbol	Meaning
λ	A security parameter as input
G	Additive cyclic group whose two orders are big prime numbers $q > 2^k$
G_T	Multiplicative cyclic group whose two orders are big prime numbers $q > 2^k$
e: $G \times G \longrightarrow G_T$	Bilinear mapping
$H_1/H_2/H_3$	Hash functions: $H_1:\{0,1\}^* \longrightarrow G$, $H_2:\{0,1\}^* \longrightarrow Z_q^*$, $H_3: G \longrightarrow Z_q^*$
g,u	Generators of group G
p_{pub} (mpk)	PKG randomly selects $x_0 \in Z_q^*$ and calculates $P_{pub} = g^{x0}$
Msk	s selected randomly by PKG
skID	The corresponding private key to ID, $skID = b + x_0 H_2(ID,B) \bmod q$, where $b \in Z_q^*$, $B = g^b$
Pk	Public key:$pk = B \cdot P_{pub}^{H2(ID,B)\bmod q}$
PP	Public parameter $PP = \{G, G_T, e, q, g, u, H_1, H_2, H_3, p_{pub}\ (mpk)\}$
formula (1)	$g^{skID} = B \cdot P_{pub}^{H2(ID,B)\bmod q}$
Entrust	$Entrust = (H_1(ID,ID_{TPA}))^x$. $x \in Z_q^*$ as the secret key to generate authorization, and calculate $V = g^x$ as the legal authority verification value.
m_i	Data block of the file
C	The improved multibranch tree authentication structure
I	The index of the data block m_i
Name	The file identifier
V_n	The current version number
t_i	The timestamp
σ_i	File authentication tag $\sigma_i = (H_1(name \| V_n \| t_i) \cdot u^{H3(mi)})^{skID}$
Φ	The ordered set of σ_i
Γ	Deputy root nodes signed by skID, $\Gamma = (H_1(R^*))^{skID}$
γ	Root node signed by skID, $\gamma = (H_1(R))^{skID}$
Chal	Challenge $chal = \{1, v_i\}_{i \in I}$, where $v_i \in Z_q^*$ randomly generated by TPA
formula (2)	$e(Entrust, g) = e(H_1(ID,ID_{TPA}),V)$
formula (3)	$e(\gamma,g) = e(H_1(R),pk)$
formula (4)	$e(\Gamma,g) = e(H_1(R*),pk)$
formula (5)	$e(T,g) = e(\prod_{i \in I} H_1(name \| V_n \| t_i)^{v_i} \cdot u^{\mu}, B \cdot P_{pub}^{H_2(ID,B)\bmod q})$

Audit proof generation based on data masking module is executed on CSP, which generates audit proof according to the audit challenge sent by TPA. It is realized by $Proof(chal,\sigma_i,m) \longrightarrow P$, where inputs are the data block m_i, the authentication tag σ_i and the challenge *chal*, and the output is audit proof. Details will be introduced in the following subsection.

Verification module is executed on TPA, which verifies the audit proof returned by CSP and judges whether the CSP stores the user data completely, then sends the verification result to the user. It is realized by $Verify(chal, PP,ID,P) \longrightarrow \{0,1\}$, where inputs are public parameters, challenge *chal*, user ID, and audit proof, and outputs the audit result 0 or 1 to indicate whether the file stored in the cloud has been tampered. The TPA verifies whether formula (3) and (4) are hold. If they do not hold, it means the integrity of the file cannot be guaranteed and then outputs fail. If they are hold, TPA judges whether the proof is correct by checking the following formula (5). If the equation is true, it means the data stored in the cloud are integrated, and TPA outputs 1, otherwise outputs 0.

Supporting dynamic update means users can update cloud data without downloading files from the cloud. The user data are updated through the interaction between the user and CSP. The module consists of two parts: one part is update request generation and update proof verification run by the user, and the other part is update proof generation run by the cloud service provider. A user generates the update request information as *update* and sends it to the CSP by executing the update request generation module. After the CSP receives the update request, it runs the update proof generation module and sends the update audit proof P_{update} to the user. Then, the user verifies P_{update} provided by the CSP. If the verification is successful, it means the update operation is performed correctly. The user can delete the locally stored data information; otherwise, the verification fails. Specific agreements will be described in the following sections.

For the convenience of subsequent introduction, a unified symbol description table is given in Table 1.

4.2. Construction of Improved Multibranch Tree Authentication Structure. The traditional multibranch tree authentication [31] only stores the hash value of the data block in the leaf nodes, the data structure is huge, and the effective utilization of nodes is low. In Ldasip, the multibranch authentication tree is reformed as shown in Figure 3. First, data

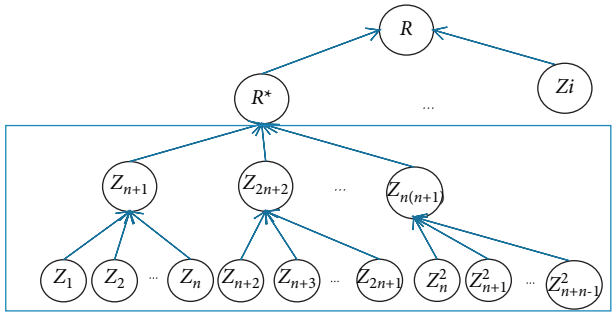

Figure 3: Improved multibranch certification tree structure.

block information is stored on nonleaf nodes of multibranch tree such as z_{n+1}. Second, the deputy root node is set to shorten the length of the authentication path. Third, the principle of locality is used to add access frequency to the node to record the frequency of the data block being accessed.

Definition 1. Deputy root node R^* In a multibranch tree, a node is selected from all the tree nodes in a region of the tree to describe the integrity of the information stored by this node and all the descendant nodes below, and it is called the deputy root node of this region, denoted by R^*.

In Figure 3, the node R^* is the deputy root node of all nodes in the rectangular box area, and Q is set as the identifier of the deputy root node, which is expressed as follows.

Root node R is a special deputy root node used to describe the integrity of the whole file. All R^* are signed separately to obtain the unique signature set Γ of the deputy root nodes of file F. These deputy root nodes and signatures are used for data integrity verification and user data update.

In the multibranch tree structure, there is an n-branch tree below the deputy root node; that is, each leaf node of the tree has n child nodes, while each leaf node can only have one parent node. The depth of the tree is d, and each node in the tree is a data container, used to store the node identification information and the hash information of the data block corresponding to the data block.

According to the principle of locality given by [32], researchers find that data access is characterized by stages and aggregation when analyzing programs in which aggregation is usually reflected in temporal locality and spatial locality. Locality of time refers to the fact that after data have been accessed, it is likely to be accessed again shortly after. Spatial locality means that after one datum is accessed, data with adjacent addresses may also be accessed shortly thereafter. Therefore, this paper designs the information stored by the node of multibranch tree with the help of locality principle, aiming to realize the efficient search of multibranch tree. Specifically, the node storage information is as follows.

Definition 2. Node storage information multibranch tree storage information is denoted as $z_i = (\Psi, h(z_i), F)$, where $\Psi = (Q, A_i)$ is the identifier, Q is the identifier of the deputy root nodes, and A_i is the node version number identification to ensure the freshness of the node, $h(z_i)$ is the hash value of the node, which is obtained by hashing the hash value $h(m_i)$ of its corresponding data block after linking with the hash value of the child node, F is the recent access frequency; that is, if there are q requests for access in the most recent time interval t, then $F = q/t$.

Definition 3. Authentication path. The authentication path refers to the set of all parent nodes on the path of the i-th node from bottom to top, starting from the user request authentication node to the deputy root node, recorded as $path_i = (r_{1,i}, r_{2,i}, \ldots, r_{j,root'})$, where $path_l = d$ is the authentication path length, $r_{1,i}$ indicates the i-th node that needs to be verified, and $r_{d,root*}$ represents the deputy root node.

The construction process of the improved multibranch tree authentication structure includes initialization and the construction of multibranch tree, which are shown in Algorithm 1 and Algorithm 2.

Compared with the traditional audit scheme based on multibranch tree, Ldasip stores the data information in the nonleaf nodes of multibranch tree. For the same level and the branches of the tree structure, Ldasip stores more data blocks, so the file will be divided into smaller blocks, and smaller blocks will shorten the time of computing the hash value, thus increasing the operation efficiency of the whole tree structure. In addition, by adding deputy root nodes, Ldasip enables decentralization. In the integrity of the audit process, the CSP traverses the tree structure when retrieving data blocks based on the audit challenge sent by the TPA, and Ldasip can query the recent access frequency of the data block from the node storage information of the deputy root node. It starts to traverse from the deputy root node with high recent access frequency to search for data blocks and quickly find the area where the target data block is located. Thus, it does not need to traverse the entire tree structure to leaf nodes like the traditional scheme, and Ldasip shortens the retrieval path and reduces the file retrieval time of CSP and the computation cost of cloud service providers.

In the same way, when users dynamically update the file, CSP can quickly find the corresponding data block by traversing from the deputy root node with high access frequency, and then, CSP updates the file. Ldasip only needs to update the part of the deputy root node hash value and shortens the update levels of the hash value, and it also shortens the path when it calculates the hash from the bottom up and reduces the computation overhead of CSP.

4.3. Generation of Audit Proof Based on Data Masking. In Ldasip, a data masking technology is introduced into the proof generation algorithm executed by CSP in order to prevent the sensitive information of users from deriving by curious third parties. The details are as follows.

The CSP is responsible for executing the proof generation algorithm. First, CSP calculates $T = \prod i \varepsilon I \sigma_i^{vi}$ and $\mu = \sum_{i \varepsilon I} v_i m_i$ and then sends them to TPA as an audit proof for verification after the CSP receives the audit challenge *chal* sent by the third-party auditor. However, if μ is directly sent

```
Inputs: file information M, n blocks, tree height d, hash function H, traversal variable i
Outputs: {Ψ(Q, Ai), HASH, F}, List<TreeNode>
TREENODE//Set the node structure
{
{Ψ(Q, Ai), HASH, F}//Information set
List<TreeNode> Child;//Child linked list
TreeNode Father;//Father node
}
FILE INITIALIZATION (M)
List<TreeNode>up//x-1 layer node linked list
List<TreeNode>m//x layer node linked list
CUR_DEPTH = 0//Height of current tree
List<TreeNode>R* = {R}//At present, R* includes the root node R, so it is only necessary to find all R*
FOR i = 1 TO n̂d//if the n-tree of d-layer is formed, the file M needs to be divided into nd copies
{
m.add(new TreeNode(block_i))
}
```

Algorithm 1: Node storage information and file initialization algorithm.

to TPA, with the increase of TPA verification times, it is very likely that the data block m can be easily obtained through solving linear equation $\mu = \sum_{i\varepsilon I} v_i m_i$. To solve the above problem, we learn from Cong Wang's idea of random concealment [19], but instead of setting random concealment factor, we directly encrypt the hash value of the user data blocks. The CSP uses hash function H_3 to calculate the user data block, so that $\mu = \sum_{i\varepsilon I} v_i H_3(m_i)$ can hide the user's original data block and prevent the user data and sensitive information from being deduced and leaked by curious third parties. Finally, $P = \{T, \mu, \{H_1(z_i), \Omega_i\}_{i \in I}, \gamma, \Gamma_{1 \le i \le I}\}$ will be sent to TPA as the audit proof. The process of audit proof generation includes two parts: block searching algorithm and proof generation algorithm, which are shown in Algorithm 3 and Algorithm 4.

4.4. Authentication Protocol Supporting User Data Update. The common update of data by users mainly includes inserting, deleting, and modifying data blocks [33]. In the process of data update, the user first sends an update request to CSP, and then, CSP updates the data block; generates an update proof, the new root node, and the new deputy root node; updates the authentication structure; and sends the update proof to the user. The user needs to verify the validity of the improved multibranch tree authentication structure before verifies the update data block. If the verification passes, it continues; otherwise, it terminates. The root and the deputy root node are then recalculated and compared with the value returned from CSP. If they are consistent, it means CSP updates the file correctly; otherwise, it does not. After verification, the user signs the new root and the deputy root node, and sends them to the cloud service provider for updating.

Since the deputy root node $R*$ is added to multibranch tree structure and stores the hash value of the corresponding data block for each node in the tree with it as the root, when updating the data block, CSP does not need to retrieve the bottom leaf node when retrieving the data block, which shortens the file retrieval path and the update level of hash value. This scheme reduces the computation overhead of CSP and reduces the auxiliary information and the communication overhead between CSP and users. In addition, CSP only updates the deputy root node nearest to the data block to be updated, which reduces the auxiliary information required during the generation of audit proof and improves the audit efficiency of the whole system.

4.4.1. Verification Protocol When Data Are Modified. The user's data modification operation is essentially a replacement process. Generally speaking, it is the process of finding the target data block to be modified first and then replacing it and modifying the data block m_i to m_i'. The modification process is shown in Figure 4.

(1) The user generates the update request through the authentication tag of the new data block, the authentication tag is $\sigma_i{}' = (H_1(\text{name} \| V_n{}' \| t_i{}') \cdot u^{H_3(m_i)'})^{skID}$, and the update request is update=(M, i, m'_i,σ'_i).

(2) The user sends the update request to CSP, where M represents the modification operation.

(3) After the CSP receives the user's update request, CSP modifies the files. First, CSP replaces the original data block m_i with the new data block m_i' to generate a new file F'. Then, CSP replaces the old authentication tag σ_i with σ_i' and calculates the hash value $H_1(m_i')$ of the replaced data block and replaces the hash value on the i-th node, calculates, and updates the hash values $H_1(z_i')$ of all relevant nodes on the authentication path to form a new tree structure and outputs the new root node value R' and the deputy root node value $R*$'. Finally, the CSP generates the update proof $P_{\text{update}} = \{\{H_1(z_i), \Omega_i\}_{i \in I}, \gamma, R', \Gamma_{1 \le i \le I}, R*'\}$.

(4) CSP sends P_{update} to the user.

```
Inputs: {Ψ(Q, Ai), HASH, F}, List<TreeNode>
Outputs: deputy root node R∗
BUILDTREE(M,n,d,H,i)//Achievements: file information M, n blocks, tree height d,
hash function H, traversal variable i.
 FILE INITIALIZATION(M)
WHILE(CUR_DEPTH < d)//Build n-tree layer by layer until d-layer.
{
 TreeNode tmp_father = new TreeNode ()
 HASH = 0//Current hash value
 List<TreeNode> tmp_up//Build temporary parent node
 FOR i = 1 TO m.size
{
 IF (tmp_up.size() < n)//Have not formed a block, continue to traverse the nodes
{
  tmp_up.add(m[i])
  hash = H(hash, m[i]- > hash)//Keep taking hash values
  m[i]- > Father = tmp_father//Set the father of the current node as "temporary father node"
  tmp_father - > Child.add(m[i])//Add a child to the "temporary father node"
}
ELSE
{
  tmp_father- > hash = hash
  IF (CUR_DEPTH = = d/2)//Mark the deputy root node and add it to the result
  {
  tmp_father- > Q = 1
  R∗.add(tmp_father)
}
  ELSE
{
  tmp_father- > Q = 0
  }
  up.add(tmp_father)
  tmp_up.clear()
  tmp_father = new TreeNode()
  i i-1
}
  m = up;
  up.clear();
++CUR_DEPTH;
}
return R∗
```

Algorithm 2: Improved multibranch tree authentication structure construction algorithm.

```
Inputs: user file block m, and deputy root nodes signature set Γ
Output: {path, {tp.Ψ, tp.HASH, tp.F}
SEARCHING (Γ, m, i)
SORT(R∗, cmp_F)//Order R∗ according to access frequency F.
FOR i = 1 TO R∗.size()//Traverse all R∗
{
  QUEUE<TreeNode>QUE
  QUE.push(R∗[i])
  R∗[i].F + = 1
  WHILE(!Q.empty())
{
  TreeNode tp = QUE.top()
  QUE.pop()
  IF(tp.Child.empty() andand tp.HASH = = m.HASH)
```

Algorithm 3: Continued.

```
{
   List<TreeNode> path;
   path.add(tp)
   WHILE(path.back().Q = = 0)
{
    path.add(path.back().Father)
}
   REVERSE(path)
   return {path, {tp.Ψ, tp.HASH, tp.F}}//Return path, information
}
ELSE
{
    FOR j = 1 TO tp.Child.size() DO
{
   QUE.push(tp.Child[j])
}
}
}
}
```

Algorithm 3: Block searching algorithm.

Inputs: audit challenge Chal = $(i, v[i])$, file authentication label $\sigma[i]$, user file block m, system parameter PP, root node signature γ, and deputy root nodes signature set Γ
Output: Audit proof P
AUDIT PROOF GENERATION(T,μ)
$T = 1$;$\mu = 0$;P=NULL;
FOR $i = 1$ TO TPAAuditElement.length I {
$T = T^{*}\,\mathrm{Pow}(\sigma[i], v[i])$;
$\mu = \mu +$ (Element from Hash3 $(m[i])) \cdot * v[i]$;
}
$P = \{T,\mu,L,H(z_i),path,\gamma,\Gamma_{1\le i\le l}\}$;
Return P;

Algorithm 4: Audit proof generation based on data masking algorithm.

(5) The user verifies the information based on the proof P_{update} provided by the CSP and uses $\{H_1(z_i), \Omega_i\}$ to generate the original root R and deputy root R^{*}.

(6) The user judges whether formulas (3) and (4) are true or not. If they are not true, the user outputs fail. If they are true, the user continues to verify whether CSP performs the data modification operation correctly.

(7) The user generates root R'' and $R*''$ with $\{H_1(z'_i),\Omega_i\}$ and compares with R' and $R^{*\prime}$ returned from CSP if the two values are equal. It means that CSP performed the modification operation correctly.

(8) The user computes the signatures of the root node γ' and the deputy root node Γ'.

(9) The user sends the new signatures to the CSP.

4.4.2. Verification Protocol When Data Are Deleted. As shown in Figure 5, verification interaction during data deletion is as follows:

(1) In the process of data deletion, the user first sends a delete request update=(D,i) to CSP, i represents the sequence number of the data block to be deleted, and D represents the deletion operation.

(2) CSP updates the file after receiving the request message. First, CSP retrieves the data structure, finds and deletes the specified data block mi, updates the hash values of the root node R′ and the deputy root node R∗′, and then sends the updated proof Pupdate = {{H1(zi),Ωi}i ∈ I,γ,R′,Γ1≤i ≤ I,R∗′ } to the user.

(3) The user verifies the proof and judges whether CSP updated honestly like the above modification operation.

(4) If CSP updated the file correctly, the user sends the new signatures to CSP.

4.4.3. Verification Protocol When Data Are Inserted. Assuming that $m_{\text{i}}*$ is to be inserted after the data block m_{i}. The process during data insertion is shown in Figure 6.

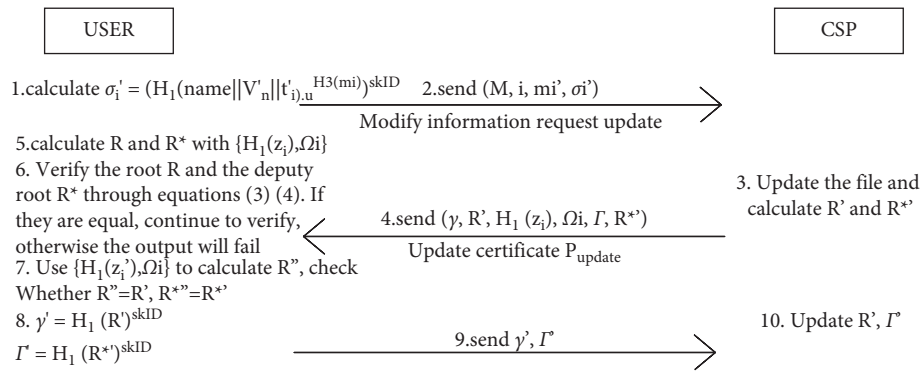

Figure 4: Verification interaction during data modification.

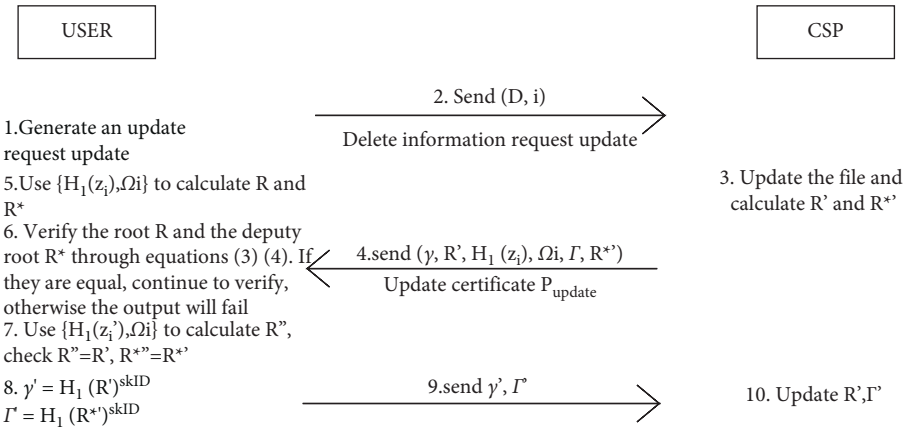

Figure 5: Verification interaction during data deletion.

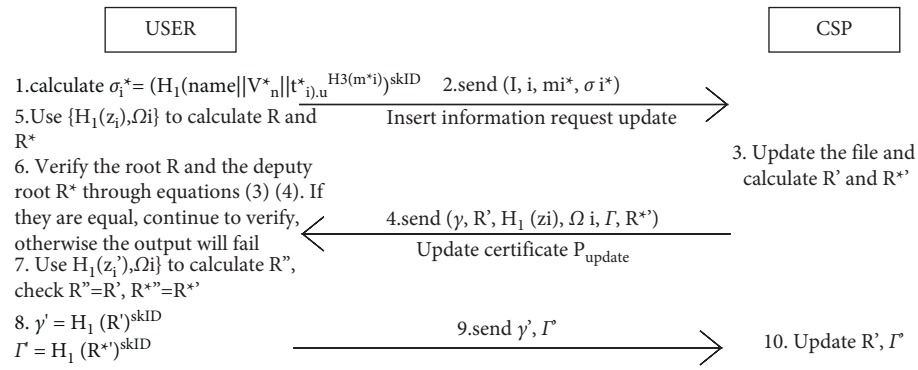

Figure 6: Verification interaction during data insertion.

(1) The user first generates an authentication tag $\sigma_i^* = (H_1(\text{name}\|V_n^*\|t_i^*) \cdot u^{H_3(m^*_i)})^{skID}$ for the data block to be inserted and sends an update request update=$(I,i, m_i^*\sigma_i^*,\sigma*_i)$ to CSP. I represents the insertion operation.

(2) CSP inserts the data block at the specified position. After the CSP receives the update request, it updates the file. First, CSP retrieves the location of data block m_i and inserts $m*$ behind it, then updates the authentication tag set $T' = \{\sigma_1, \sigma_2, \ldots, \sigma_i, \sigma_*, \ldots, \sigma_I\}$. The node hash value of the inserted data block is updated to the hash value of the original data block m_i and the inserted data block $m*$, and then, CSP updates the hash value of the deputy root node and sends the update proof $P_{\text{update}} = \{\{H_1(z_i),\Omega_i\}_{i \in I}, \gamma,R',\Gamma_{1\le i \le I},R*'\}$ to the user.

(3) The user verifies the proof and judges whether CSP updated honestly like the above modification operation.

(4) If CSP updated the file correctly, the user sends the new signatures to CSP.

Table 2: Function comparison of related schemes.

Scheme	Public verification	Simplify certificate management	Support data privacy	Data dynamic	Batch verification
Scheme [9]	√	√	×	×	×
Scheme [10]	√	√	√	×	×
Ldasip	√	√	√	√	√

5. Theoretical Analysis

In this section, the Ldasip method is analyzed theoretically, including functional, security analysis, and communication cost comparison with existing schemes.

5.1. Functional Comparison. There are several identity-based data integrity audit schemes, in which scheme [9] and scheme [10] are classic schemes. As shown in Table 2, both Ldasip and the scheme [9] adopt identity-based integrity audit, while the scheme [10] adopts fuzzy identity-based data integrity audit. All three schemes can simplify certificate management. The scheme [9] does not support data privacy protection and neither scheme [9] nor scheme [10] supports data dynamic and batch verification. Ldasip supports data privacy protection, data dynamic, and batch verification at the same time.

5.2. Security Analysis. This section mainly analyzes the correctness, integrity, and sensitive information protection ability of Ldasip.

5.2.1. Correctness Analysis. The correctness of a cloud audit approach is that the information generated by the private key generation algorithm KeyExtract(), the authorization algorithm Entrust(), and the proof generation algorithm Proof() can be equation-verified with overwhelming probability. The following analysis results are presented in the form of propositions. See appendix for the proof process.

Proposition 1. *Let the user's private key be* $skID = b + x_0 H_2(ID,B) \bmod q$ *and public key* $pk = B \cdot P_{pub}^{H2(ID,B) \bmod q}$, *the equation* $g^{skID} = B \cdot P_{pub}^{H2(ID,B) \bmod q}$ *can be proved to be true.*

Proposition 2. *indicates that the private key will definitely pass the user's verification when PKG sends the correct private key to the user in Ldasip.*

Proposition 3. *Given the legal authority verification value* $V = g^x$ *and the legal authority Entrust=*$(H_1(ID, ID_{TPA}))^x$, *the equation* $e(Entrust, g) = e(H_1(ID, ID_{TPA}), V)$ *can be proved to be true according to the properties of bilinear mapping [23].*

Proposition 4. *shows that only TPA with the legal authority can get the proof provided by the cloud and audit instead of the user in Ldasip.*

Proposition 5. *Let the root node signature be* $\gamma = H_1(R)^{skID}$, *the user's public key be* $pk = g^{skID}$, *and the equation* $e(\gamma, g) = e(H_1(R), pk)$ *can be proved to be true according to the bilinear mapping property. Knowing* $T = \prod i \varepsilon I \sigma_i^{vi}$ *and* $\mu = \sum_{i \varepsilon I} v_i H_3(m_i)$, *equation (5)* $e(T, g) = e(\prod_{i \in I} H_1(name \| V_n \| t_i)^{v_i} \cdot u^{\mu}, B \cdot P_{pub}^{H_2(ID,B) \bmod q})$ *can be proved to be true.*

Proposition 6. *shows that the audit proof based on the challenge information must pass the validation successfully if both the TPA and the CSP are trusted, and the data files are stored correctly in Ldasip.*

5.2.2. Soundness Analysis. The following analysis will ensure that any CSP that can generate valid proofs and pass validation algorithms is in fact storing complete files. See appendix for the proof process.

Proposition 7. *Suppose* $P = \{T, \mu, \{H_1(z_i), \Omega_i\}_{i \in I}, \gamma, \Gamma_{1 \le i \le l}\}$, *the equation* $e(T, g) = e(\prod_{i \in I} H_1(name \| V_n \| t_i)^{v_i} \cdot u^{\mu}, B \cdot P_{pub}^{H_2(ID,B) \bmod q})$ *cannot be verified when* m_i *is replaced by* m'_i.

Proposition 7 indicates that an adversary may not provide a valid audit proof if he does not store or does not store files fully. In other words, if data stored outsourced in the cloud have been compromised, it is computationally infeasible for the CSP to fabricate data to obtain the verifiable audit proof.

5.2.3. Ability to Protect Sensitive Information. The following part analyzes whether Ldasip can protect sensitive information.

Proposition 8. *Suppose that TPA has the audit proof* $P = \{T, \mu, \{H_1(z_i), \Omega_i\}_{i \in I}, \gamma, \Gamma_{1 \le i \le l}\}$, *but he cannot infer data block* m_i.

The proof process is shown in the appendix.

Proposition 8 shows that the TPA cannot obtain the user's original data from the audit proof.

5.3. Performance Analysis and Comparison

5.3.1. Computation Overhead. The computation cost of Ldasip is analyzed, and Proposition 9–Proposition 13 are obtained. See the appendix for specific proofs.

Proposition 9. *If* $Hash_G$ *represents a hash operation in G,* Exp_G *represents a power operation in G, and Pair represents a pairing operation in e:GxG* $\longrightarrow G_T$. *The computation cost of the integrity audit scheme in the third-party legal authority generation and verification phase is* $2Hash_G + 2Exp_G + 2Pair$.

Proposition 10. *If* $Hash_G$ *represents a hash operation in G,* Exp_G *represents a power operation in G, and* Mul_G *represents a multiplication operation in G. The computation overhead of*

Table 3: Qualitative analysis of the calculation cost of this scheme and existing schemes.

scheme	Signature generation stage	Challenge response phase	Verification phase	Dynamic operation phase
[9]	$nHash_G + nMul_G + 2Exp_G$	(c-1) $Mul_G + cExp_G + cMul_{z*q} + (c-2)\ add_{z*q}$	(c+2) $Exp_G + cHash_G + (c+1)\ Mul_G + Hash_{z*q} + 6Pair$	Not supported
[10]	$nHash_G + 3nMul_G + (s+3)\ nExp_G$	(c-1) $Mul_G + cExp_G + cMul_{z*q} + (c-1)\ add_{z*q}$	$(3c + cs + 1)\ Exp_G + nHash_G + (5c + sc-1)Mul_G + (2c+1)Pair$	Not supported
Ldasip	$(n+2)\ Hash_G + nMul_G + (2n+2)\ Exp_G + nHash_{z*q}$	(c-1) $Mul_G + cExp_G + cMul_{z*q} + (c-1)\ add_{z*q}$	$(c+2)Exp_G + (c+2)\ Hash_G + cMul_G + Hash_{z*q} + 6Pair$	$4Exp_G + 7Hash_G + Mul_G + 4Pair + Hash_{z*q}$

Note. n represents the number of divided blocks of the file, s represents the number of sectors divided by each data block, and c represents the number of data blocks challenged by the auditor.

the signature generation stage is $(n+2)Hash_G + nMul_G + (2n+2)Exp_G + nHash_{z*q}$.

Proposition 11. *If* $Hash_{z*q}$ *represents a hash operation in* Z_q^*, Exp_G *represents a power operation in G,* Mul_G *represents a multiplication operation in G,* Mul_{z*q} *and* Add_{z*q} *represent one multiplication operation and one addition operation in* Z_q^*, *and c is the number of data blocks being challenged. The computation overhead of the challenge response phase is (c-1)* $Mul_G + cExp_G + cMul_{z*q} + (c-1)Add_{z*q}$.

Proposition 12. *If* $Hash_G$ *represents a hash operation in G,* $Hash_{z*q}$ *represents a hash operation inz*$_q^*$, Exp_G *represents a power operation in G, Pair represents a pairing operation in e:* $GxG \longrightarrow G_T$, Mul_G *represents a multiplication operation in G, and c is the number of data blocks being challenged. The computation overhead of the verification phase is (c+2)* $Exp_G(c+2)Hash_G + cMul_G + Hash_{z*q} + 2Pair$.

Proposition 13. *If* $Hash_G$ *represents a hash operation in G,* Exp_G *represents a power operation in G, Pair represents a pairing operation in e:* $GxG \longrightarrow G_T$, Mul_G *represents a multiplication operation in G, and c is the number of data blocks being challenged. The computation overhead of the dynamic operation phase is* $7Hash_G + 4Exp_G + 4Pair + Mul_G \cdot Hash_{z*q}$.

According to the above Proposition 9–Proposition 13, it can be compared with scheme [9] and scheme [10] in the computation overhead of signature generation phase, challenge response phase, verification phase, and dynamic operation phase. As shown in Table 3, although the computation cost of Ldasip in the signature process is slightly higher than that in the literature [9], because Ldasip supports dynamic update, the process needs to calculate the signatures of root nodes and deputy root nodes. Compared with the large amount of computation caused by users downloading files locally and updating them in static audit, the cost of Ldasip in this stage is far less than the above situation, and Ldasip can support sensitive information protection, but the literature [9] does not. Compared with the literature [10], the computational cost of our scheme in tag generation and verification stage is obviously less than that in the literature [10]. Ldasip uses the improved multibranch tree as the authentication structure in the audit process, which realizes the low-cost construction of the tree. In addition, this structure can shorten the authentication path in the audit process, effectively reduce the computational burden of users and third-party auditors, and improve the efficiency of integrity audit. Moreover, the scheme in this paper can meet the needs of users' sensitive information protection and dynamic update at the same time. Ldasip is more applicable in the cloud storage environment, since it reduces the computation burden of TPA and the user, and it can support sensitive data protection and dynamic operation.

5.3.2. Communication Overhead. Communication overhead mainly comes from the transmission of legal authority, audit challenge, audit proof, and data update process. Proposition 14 analyzes the communication cost at each stage, and see the appendix for specific proofs.

Proposition 14. *If* $|p|$ *represents the size of an element in G, and according to the safety and practical experience, we choose random numbers from the big prime number* Z_q^*, *which* $|q|$ *represents the size of an element in* Z_q^*, $|m|$ *is the size of the data block, n is the number of data blocks, and* $|n|$ *represents the size of an element in set [1,n], and l represents the number of deputy root nodes included in the retrieval process of authentication structure. The communication overhead of the authorization process is* $2|p|+|q|$. *The communication overhead of the data tag value and signature upload process is* $(n+2)|p|+n|m|$. *The communication overhead of the audit challenge response process is* $(c+3)|p|+(c+1)|q|+c|n|$, *and the communication overhead of the data update process is* $(l+7)|p|+|m|$.

As shown in Table 4, the communication cost of Ldasip is equivalent with one of [9] in the process of uploading signatures and audit proof. Compared with [9], Ldasip supports sensitive data protection and dynamic data update. The communication cost of Ldasip is lower than that in [10]. Compared with [10], Ldasip supports dynamic data update and achieves high efficiency of data integrity audit.

We further analyzed and compared the communication complexity with schemes [9, 10]. As seen from Table 5, the communication complexity of each entity in the audit process is reduced in Ldasip. Where n represents the number of data blocks, and m represents the number of

Table 4: Compared with the communication cost of existing solutions.

Scheme	Data tag value, signature upload	Audit challenge	Audit proof	Data update
Scheme [9]	$n(\|p\|+\|m\|)$	$c(\|n\|+\|q\|)$	$\|p\|+\|q\|$	-
Scheme [10]	$(3n+1)\|p\|+n\|m\|$	$c(\|n\|+\|q\|)$	$s\|q\|+(2c+1)\|p\|$	-
Ldasip	$(n+2)\|p\|+n\|m\|$	$c(\|n\|+\|q\|)$	$(c+3)\|p\|+\|q\|$	$(l+7)\|p\|+\|m\|$

Table 5: Comparison of communication complexity of each scheme.

Scheme	Communication complexity		
	DO	TPA	CSP
Scheme [9]	O ($\log_2 n$)	O ($\log_2 n$)	O ($\log_2 n$)
Scheme [10]	O ($\log_2 n$)	O ($\log_2 n$)	O ($\log_2 n$)
Ldasip	O ($\log_2 n$)	O ($\log_2 m$)	O ($\log_2 m$)

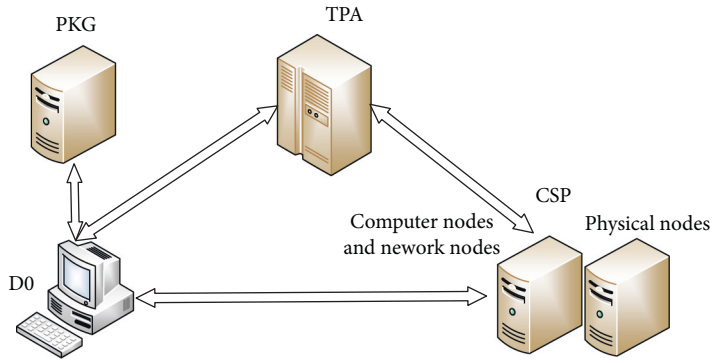

Figure 7: Experimental topography.

subnodes of the deputy root node of the multibranch tree, which n is much bigger than m.

6. Experimental Evaluation

We have implemented Ldasip in an OpenStack-based cloud computing platform. Comprehensive experiments have been conducted to compare with the existing schemes [9, 10] at public verification, simplified certificate management, support for data protection, data dynamics, and batch verification.

6.1. Experimental Evaluation. The experiment topology is shown in Figure 7. Five physical machines are used in the experiment. One physical machine serves as the user, one physical machine serves as PKG, and one physical machine serves as TPA. The cloud service provider provides cloud platform services. We deployed the control node to a single physical machine, and the compute node and the network node to a single physical machine. Figure 8 is a sequence diagram between entities. In our experiment, we set the base field size to 512 bits, and the size of Z_q^* ($|p|$) is 160 bits.

6.2. Authentication Structure Effect. In the experiment, we compared CSP computation time and the authentication tree construction time of Ldasip with traditional MHT authentication structure under different data block conditions. The file with the size of 200M is divided into blocks according to each data block of 1 KB from which different numbers of data blocks are extracted each time and record the computing time.

As shown in Figure 9 and 10, the horizontal axis is the number of data blocks, and the vertical axis is CSP computing time and the authentication construction time, respectively. The construction time of Ldasip is greatly reduced compared with that of MHT, and the audit efficiency of Ldasip is significantly improved compared with MHT structure. Therefore, Ldasip can reduce the computational burden of users and cloud service providers, thus improving the performance of audit methods.

6.3. Computation Overhead. Table 6 shows the specific algorithms of Ldasip and scheme [9, 10] in the signature generation stage, challenge response stage, and verification stage.

In the experiment, we compared the signature generation computational overhead of scheme [9, 10] with Ldasip under different data block conditions. Let the file size be 20 MB, each file is divided into 1000000 data blocks, with an interval of 100, select different data blocks from 0 to 1000 for experiment, and record the computation overhead.

As shown in Figure 11, where the horizontal axis is the number of data blocks that generate the signature, and the vertical axis is the signature computation time. The experimental results show that the computation overhead of the

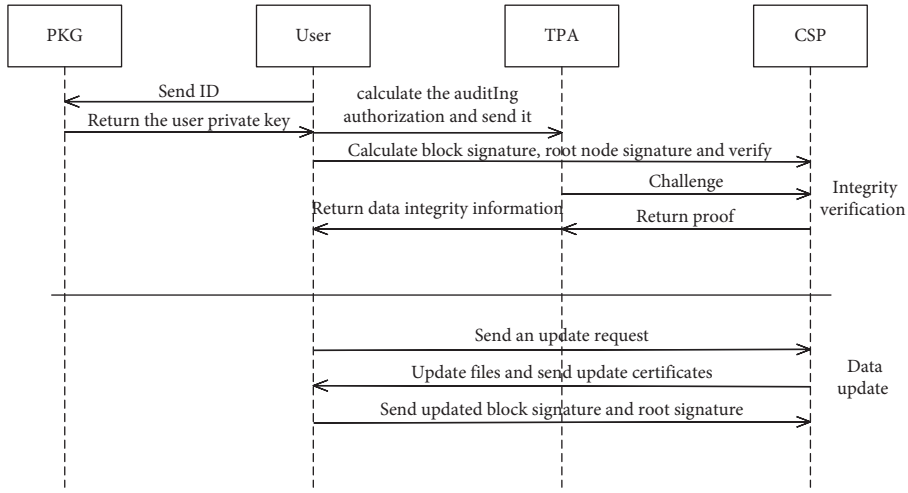

Figure 8: Sequence diagram of data integrity audit.

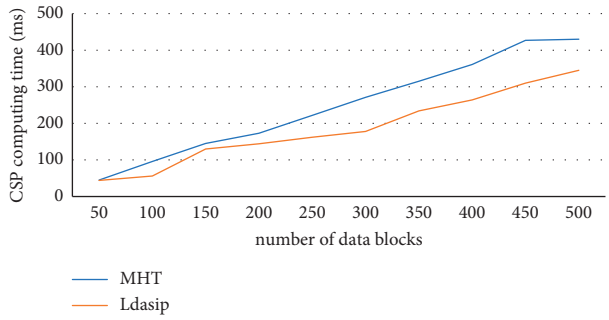

Figure 9: Different data block numbers with a step size of 50. The overhead of CSP computing time under different authentication structures.

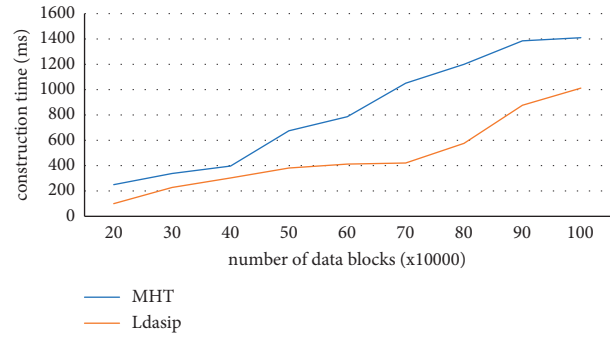

Figure 10: Different data block numbers with a step size of 10^5. The overhead of construction time under different authentication structures.

signature process increases linearly with the increase of the number of data blocks for all methods. The cost of Ldasip in signature generation stage is less than that in scheme [10] but slightly higher than that in scheme [9]. This is because Ldasip supports dynamic update, and the process needs to calculate the signatures of root nodes and deputy root nodes. Compared with the large amount of calculation caused by users downloading files locally and updating them in static audit, the cost of Ldasip at this stage is far less than the above situation.

In the experiment, we compared the challenge response computational overhead of scheme [9, 10] with Ldasip under different data block conditions. Let the file size be 20 MB, each file is divided into 1000000 data blocks, with an interval of 100, select different data blocks from 0 to 1000 for experiment, and record the computation overhead.

As shown in Figure 12, the horizontal axis is the number of challenged data blocks, and the vertical axis is the challenge response overhead. Experimental results show that the computation overhead of the challenge response process increases linearly with the increase of the number of challenged data blocks for all methods. However, the overhead of Ldasip is lower than that of the scheme [9, 10]. The reason is that Ldasip introduces an improved multitree, which makes CSP spend less time on searching and calculating data blocks. In addition, Ldasip can also realize the protection of user sensitive data.

In the experiment, we compared the verification computational overhead of scheme [9, 10] with Ldasip under different data block conditions. Let the file size be 20 MB; divide each file into 1000000 data blocks, with an interval of 100; select different data blocks from 0 to 1000 for experiment; and record the computation overhead.

As shown in Figure 13, the horizontal axis is the number of challenged data blocks, and the vertical axis is the verification overhead. The experimental results show that the computing overhead of verification increases with the increase of the number of challenged data blocks. The calculation process and parameter size of Ldasip and Scheme [9] in the verification stage are nearly the same, and the simulation results change linearly. Because the length of a single challenge chal is constant, with the linear increase of the number of challenge data blocks, the computational overhead is also linear. Although Ldasip adds the root node integrity verification, the amount of calculation is very small, which is practically negligible. Therefore, the cost of Ldasip is basically the same as that in the scheme [9], but significantly lower than

Table 6: Algorithms of different audit schemes in signature generation stage.

Stage	Scheme [9]	Scheme [10]	Ldasip
Signature generation	$\sigma_i = (H(i)\mu^{mi})^{skID}$	$\sigma_{1i}^{(k)} = \left\{ skID \cdot (H(\text{name}\|i) \cdot v' \cdot \prod_{j=1}^{s} u_j^{mij})^{s_k} \right\}_{k\epsilon\omega}$ $\sigma_{2i}^{(k)} = \{g^{-s_k}\}_{k\epsilon\omega}$ $\sigma_{3i}^{(k)} = \{g^{-r_k}\}_{k\epsilon\ \omega}$	$\sigma_i = (H_i(\text{name}\|V_n\|t_i) \cdot u^{H3(mi)})^{sKID}$ $\gamma = (H_1(R))^{skid}$ $\Gamma = (H_1(R*))^{skID}$
Challenge response	$T = (\prod_{j=1}^{c} \sigma i_{ij}^{vj})$, $\widehat{m} = \sum_{j=1}^{c} v_j m_{ij}$	$\mu_j = \sum_{(i,vi)\epsilon I} v_i m_{ij}$ $\sigma_1^{(k)} = \left\{ \prod_{(i,vi)\epsilon I} \sigma_{1i}^{(k)^{vi}} \right\}_{k\epsilon\omega}$ $\sigma_2^{(k)} = \{\sigma_{2i}^{(k)}\}_{i\epsilon I, k\epsilon\omega}$ $\sigma_3^{(k)} = \{\sigma_{3i}^{(k)}\}_{i\epsilon I, k\epsilon\omega}$	$T = \prod i\varepsilon I \sigma_i^{vi}$, $\mu = \sum_{i\varepsilon I} v_i H_3(m_i)$
Verification	$e(T, g) = e(\prod_{j=1}^{c} (H(i_j))^{vi} \mu^{\widehat{m}}, R \cdot Y^{H(ID,R)})$	$\prod_{(i,vi)\epsilon I} A^{vi} = \prod_{k\epsilon S} ((e(\sigma_1^{(k)}, g) \cdot \prod_{(i,vi)\epsilon I} e(T(k), \sigma_3^{(k)^{vi}}) \cdot e((H(name\|\|i)v')^{vi} \prod_{j=1}^{s} u_j^{\mu j}, \sigma_2^{(k)}))^{\Delta k, S(0)}$	$e(\gamma, g) = e(\prod_{i\epsilon I} H_1(\text{name}\|V_n\|t_i)^{v_i} \cdot u^{\mu}, B \cdot P_{\text{pub}} H_2(ID, B) \text{mod} q) e(T, g) = e(\prod_{i\epsilon I} H_1(\text{name}\|V_n\|t_i)^{v_i} \cdot u^{\mu}, B \cdot P_{\text{pub}}{}^{H_2(ID,B)\text{mod}q})$

Note. ω represents user identity, and s_k, rk represents a random for each user ID.

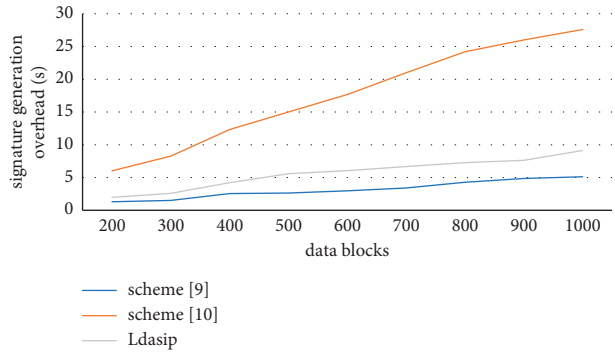

Figure 11: Different data block numbers with a step size of 100. The overhead of signature generation under different audit schemes.

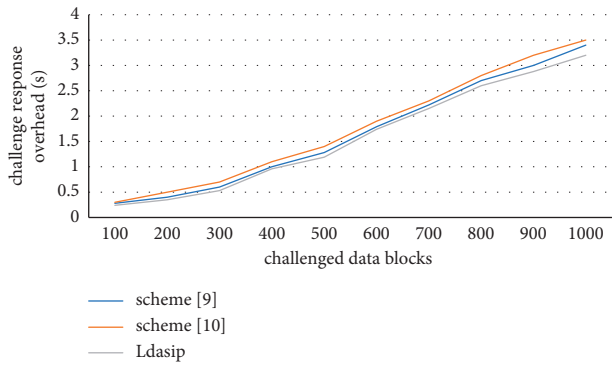

Figure 12: Different challenged data block numbers with a step size of 100. The overhead of challenged response under different audit schemes.

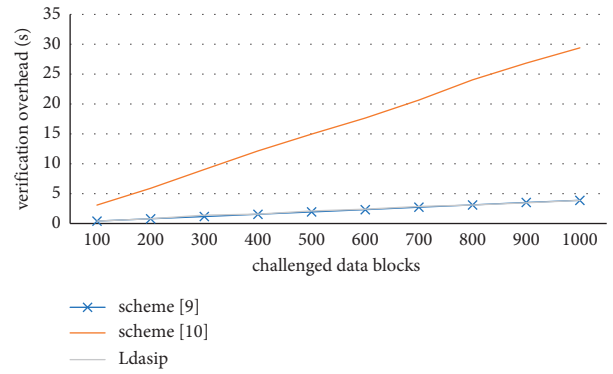

Figure 13: Different challenged data block numbers with a step size of 100. The overhead of verification under different audit schemes.

that in the scheme [10], indicating that Ldasip can guarantee good audit performance on the premise of supporting data sensitivity protection and dynamic update of data.

6.4. Communication Overhead. In the experiment, we compared the communication overhead of schemes [9, 10] with Ldasip under different data block conditions. Let the file size be 20 MB, divide each file into 1000000 data blocks every 100 intervals, select different data blocks from 0 to 1000 for experiment, and record the communication overhead.

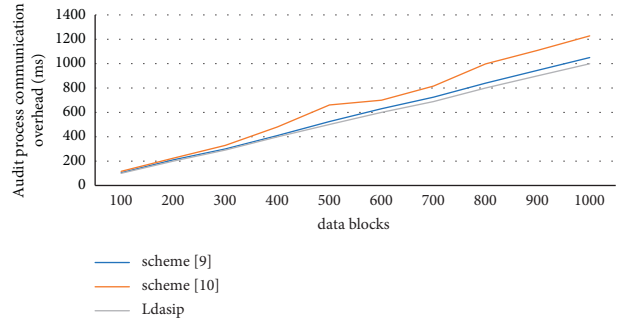

Figure 14: Different data block numbers with a step size of 100. The overhead of communication is under different audit schemes.

As shown in Figure 14, the horizontal axis is the number of challenged data blocks, and the vertical axis is the communication overhead. Experimental results show that communication overhead increases with the increase in the number of challenged data blocks. Because we introduces an improved multibranch tree structure supporting lightweight authentication, which shortens the authentication path and auxiliary information, the communication overhead of Ldasip is less than that of scheme [9, 10].

7. Conclusion

In recent years, cloud storage services have become an increasingly important part of the information technology industry. It is critical to ensure the integrity of data outsourced to the cloud. Therefore, we proposed Ldasip, a lightweight dynamic audit method that supports sensitive information protection in cloud storage. Exploiting identity-based data integrity audit, a data masking technique is introduced to protect users' sensitive information. An improved multibranch tree structure is proposed to realize dynamic audit and reduce the communication overhead during verification. Third-party legal authority verification mechanism is introduced to ensure that only the legitimate third-party authorization by the user can handle the files on behalf of the user and reduce the security threat brought by the third-party auditor. Finally, the theoretical analysis and experimental evaluation results of Ldasip are given.

However, some issues are not covered in this paper. First, more efforts should be made to support data integrity audit in more complex cloud service scenarios such as the cloud service composition. Second, this paper mainly studies the integrity audit of cloud storage data by trusted third-party auditors and supports the integrity audit of dynamic operation of data by users, without focusing on the security and performance issues in data sharing scenarios. In the future, the security and performance issues of integrity audit will be further considered when data are shared by entities other than users.

APPENDIX

Proofs of Proposition 1. Given the correct private key $skID = b + x_0 H_2(ID,B) \bmod q$ generated by PKG, the verification equation (A.1) in the KeyExtract algorithm will hold.

Based on the properties of bilinear mapping, the equation (A 1) can be proved to be correct by deriving the left-hand side from the right-hand side.

$$\begin{aligned} g^{skID} &= g^{b+x_0 H_2(ID,B) \bmod q} \\ &= g^{b} \cdot g^{x_0 H_2(ID,B) \bmod q} \\ &= B \cdot P_{\text{pub}}{}^{H_2(ID,B) \bmod q}. \end{aligned} \tag{A.1}$$

□

Proofs of Proposition 2. Given the legal authority verification value $V = g^x$ and legal authority *Entrust*= $(H_1(ID, ID_{TPA}))^x$ generated by the legal authority generation algorithm, the verification of the equation (A.2) in the Entrust algorithm will hold. Based on the properties of bilinear mapping, the equation (A.2) can be proved to be correct by deriving the left-hand side from the right-hand side:

$$\begin{aligned} e(\text{Entrust}, g) &= \left(\left(H_1(ID, ID_{TPA})\right)^x, g\right) \\ &= \left(\left(H_1(ID, ID_{TPA})\right), g^x\right) \\ &= e\left(H_1(ID, ID_{TPA}), V\right). \end{aligned} \tag{A.2}$$

□

Proofsof Proposition 3. A valid proof $P = \{T,\mu,\{H_1(z_i),\Omega_i\}_{i\in I},\gamma,\Gamma_{1\le i \le I}\}$ is given from the cloud. TPA is responsible for executing the verification algorithm. First, it verifies whether the equations (A.3) and (A.4) will hold to check the correctness of the authentication structure. If yes, it judges whether the data blocks and authentication tags stored by CSP are correct by checking whether the equation (A.5) holds. According to the properties of bilinear mapping, the left side of the equation can be deduced from the right side, which can prove that the equations (A.3) and (A.5) are correct. The equation (A.4) can also be proved.

$$\begin{aligned} e(\gamma, g) &= e\left(\left(H_1(R)\right)^{skID}, g\right) \\ &= e\left(H_1(R), g^{skID}\right) \\ &= e\left(H_1(R), pk\right). \end{aligned} \tag{A.3}$$

$$\begin{aligned} e(T, g) &= e\left(\prod_{i\in I} \sigma_i{}^{v_i}, g\right) \\ &= e\left(\prod_{i\in I} \left(\left(H_1\left(\text{name}\|V_n\|t_i\right) \cdot u^{H_3(m_i)}\right)^{skID}\right)^{v_i}, g\right) \\ &= e\left(\prod_{i\in I} \left(\left(H_1\left(\text{name}\|V_n\|t_i\right) \cdot u^{H_3(m_i)}\right)^{v_i}\right)^{skID}, g\right) \\ &= e\left(\prod_{i\in I} \left(H_1\left(\text{name}\|V_n\|t_i\right) \cdot u^{H_3(m_i)}\right)^{v_i}, g^{skID}\right) \\ &= e\left(\prod_{i\in I} \left(H_1\left(\text{name}\|V_n\|t_i\right) \cdot u^{H_3(m_i)}\right)^{v_i}, pk\right) \\ &= e\left(\prod_{i\in I} \left(H_1\left(\text{name}\|V_n\|t_i\right) \cdot u^{H_3(m_i)}\right)^{v_i}, pk\right) \cdot e\left(u^{rH_3(L)}, pk\right) \\ &= e\left(\prod_{i\in I} H_1\left(\text{name}\|V_n\|t_i\right)^{v_i} \cdot u^{\sum_{i\in I} H_3(m_i)v_i}, pk\right) \\ &= e\left(\prod_{i\in I} H_1\left(\text{name}\|V_n\|t_i\right)^{v_i} \cdot u^{\mu}, B \cdot P_{\text{pub}}{}^{H_2(ID,B) \bmod q}\right). \end{aligned} \tag{A.4}$$

□

Proofs of Proposition 4. If the file has been damaged, and a malicious cloud forged an audit proof about the damaged data block and won the following security game, then we will be able to solve the discrete logarithm problem (DL) on G. The safe games are as follows [34].

When TPA sends an audit challenge Chal = $\{1, v[i]\}_{i\varepsilon I}$ to the cloud, the audit proof of the correct data block m_i should be $P = \{T,\ \mu,\{H_1(z_i),\Omega_i\}_{i\in I},\gamma,\Gamma_{1\le i\le I}\}$. However, a malicious cloud wants to generate an audit proof of the wrong data block m'_i. The wrong audit proof $P = \{T,\ \mu',\{H_1(z_i),\Omega_i\}_{i\in I},\gamma,\Gamma_{1\le i\le I}\}$. Let us define $\Delta\mu = \mu' - \mu(\Delta\mu \neq 0)$. If this wrong proof can be successfully verified by TPA, then the malicious cloud is considered to have won the game; otherwise, it is considered to have lost the game.

It is assumed that the malicious cloud won the above game. $e(T,g) = e(\prod_{i\varepsilon I} H_1(\text{name}\|V_n\|t_i)^{v_i} \cdot u^{\mu}, B \cdot P_{\text{pub}}{}^{H_2(ID,B)\bmod q})$ can be obtained by the equation (A.5). It is assumed that $P = \{T,\ \mu', \{H_1(z_i), \Omega_i\}_{i\in I}, \gamma, \Gamma_{1\le i\le I}\}$ can be verified by TPA; then, the authors have the equation $e(T,g) = e(\prod_{i\varepsilon I} H_2(\text{name}\|V_n\|t_i)^{v_i} \cdot u^{\mu'}, B \cdot P_{\text{pub}}{}^{H_2(I\ D,B)\bmod q})$. Through these two equations and using the characteristics of bilinear mapping, the authors can get the following.

$$\begin{aligned} & e\left(\prod_{i\in I} H_1\left(\text{name}\|V_n\|t_i\right)^{v_i} \cdot u^{\mu}, B \cdot P_{\text{pub}}{}^{H_2(ID,B)\bmod q}\right) \\ & = e(T,g) \\ & = e(T',g) \\ & = e\left(\prod_{i\in I} H_1\left(\text{name}\|V_n\|t_i\right)^{v_i} \cdot u^{\mu'}, B \cdot P_{\text{pub}}{}^{H_2(ID,B)\bmod q}\right). \end{aligned} \tag{A.5}$$

Therefore, the authors can get $u^{\mu} = u^{\mu'}, u^{\Delta\mu} = 1$. Given $(g,h)\varepsilon G$, because G is a cyclic group, so there is $x \in Z_q^*$ that makes $h = g^x$ does not loss of generality. Given g, h, let $u = g^a h^b \varepsilon G$, where a and b are two random elements in Z_q^*. The authors find a way to solve the discrete logarithm problem on G, that is, $1 = u^{\Delta\mu} = (g^a h^b)^{\Delta\mu} = g^{a\cdot\Delta\mu} \cdot h^{b\cdot\Delta\mu}$, the solution of the discrete logarithm problem as follows:

$$h = g^{(-a\Delta\mu/b\Delta\mu)} = g^{(-a/b)}, x = -\frac{a}{b}. \tag{A.6}$$

Note that the probability that b is zero is only $1/q$. Because q is a large prime number, the authors believe that the probability of b being zero is negligible. In this way, the probability that the authors can solve the discrete logarithm problem is 1-1/q. However, this is contrary to our assumption that solving the discrete logarithm problem is difficult. Therefore, for a malicious cloud, it is computationally infeasible to generate an audit proof about incorrect data that can be verified by TPA. □

Proofs of Proposition 5. The TPA cannot obtain the original data from the audit proof. This scheme uses data masking technology to disguise the original data, that is, $\mu = \sum_{i\varepsilon I} v_i H_3(m_i)$. Because $H_3(m_i)$ is a hash value, the input data of the hash function cannot be obtained by reverse deduction; even if the third-party auditor obtains the value of $H_3(m_i)$ through multiple verifications, it cannot deduce the value of mi according to the equation $\mu = \sum_{i\varepsilon I} v_i H_3(m_i)$, so the third-party auditor cannot deduce the user's original data from the audit proof, thus protecting the user's data privacy.

The TPA cannot obtain the user's original data from the audit proof. Because in$\{\{H_1(z_i), \Omega_i\}_{i\in I}, \gamma, \Gamma_{1\le i\le I}\}$, $H_1(z_i)$ is the link between the hash value $H_1(m_i)$ of the data block of this node and the hash value of its child nodes, Ω_i is the auxiliary verification path information, which is also the hash value, and γ and Γ are the signature the of root node and deputy root nodes. According to the one-way nature of the hash function, it is difficult to find the input data through the hash value, γ and Γ are the signatures of the IBS, and the scheme is secure, so the third party cannot obtain the original user data. □

Proofs of Proposition 6. The legal authority stage is divided into two stages, namely, the third-party legal authority generation stage of the user and the third-party legal authority verification stage of CSP. In the authority generation stage, the computation cost for the user to calculate the legal authority verification value $V = g^x$ and the legal authority $Entrust = (H_1(ID,\ IDTPA))^x$ is $\text{Hash}_G + 2\text{Exp}_G$. In the authorization verification stage, the cloud service provider verifies the legitimacy of authority through equation $\text{e}(\text{Entrust, g}) = e(H_1(ID, ID_{TPA}), V)$, and the required computation overhead is $Hash_G$+2$Pair$. To sum up, the cost of auditing authorization stage is 2Hash_G+ $2\text{Exp}_G + 2\text{Pair}$. □

Proofs of Proposition 7. The computation overhead of the signature generation process is generated by the users. First, users need to calculate tags $\sigma_i = (H_1(\text{name}\|V_n\|t_i) \cdot u^{H_3(m_i)})^{sk_{ID}}$ for n data blocks, and the computation overhead is $n\text{Hash}_G + n\text{Mul}_G + 2n\text{Exp}_G + n\text{Hash}_{z*q}$. After that, the root node signature $\gamma = (H_1(R))^{skID}$ and the deputy root node signature set $\Gamma = (H_1(R*))^{skID}$ are signed by $skID$, and the computation cost is $2\text{Hash}_G + 2\text{Exp}_G$. To sum up, the computation overhead of the signature generation process is $(n+2)\text{Hash}_G + n\text{Mul}_G + (2n+n)\text{Exp}_G + n\text{Hash}_{z*q}$. □

Proofs of Proposition 8. The challenge response stage is divided into two parts; namely, the third-party auditor sends the audit challenge to CSP, and CSP provides the audit proof to TPA. In an audit task, TPA only spends a small amount of computation overhead to generate audit challenge, which is ignored here. The audit proof generation stage is performed by CSP. CSP first calculates $T = \prod i\varepsilon I \sigma_i^{vi}$. The required computation overhead is $(c-2)MulG + c\text{Exp}_G$, calculates $\mu = \sum_{i\varepsilon I} v_i H_3(m_i)$, and then sends the generated audit proof to TPA. The computation cost in this process is $cMul_{z*q}$+(c-1)Add_{z*q}. To sum up, the computation overhead of the of challenge is (c-1) $\text{Mul}_G + c\text{Exp}_G + c\text{Mul}_{z*q} + (c-1)\text{Add}_{z*q}$. □

Proofs of Proposition 9. The verification process is performed by the third-party auditor. When TPA receives the audit proof sent by CSP, it will judge whether the received proof is correct by verifying the equations (A.3) and (A.4). The computation cost of this process is $4\text{Pair} + 2\text{Hash}_G$. If the equations do not hold, TPA terminates verification. If the equations hold, TPA will judge whether the audit proof sent by the cloud is correct according to $e(T,g) = e(\prod_{i\in I} H_1(\text{name}\|V_n\|t_i)^{v_i} \cdot u^{\mu}, B \cdot P_{\text{pub}}{}^{H_2(I\ D,B)\bmod q})$. The computation cost of this process is $(c+2)\ \text{Exp}_G + c\text{Hash}_G + c\text{Mul}_G + \text{Hash}_{z*q} + 2\text{Pair}$. To sum up, the calculation overhead of the whole verification phase is $(c+2)\ \text{Exp}_G (c+2)\text{Hash}_G + c\text{Mul}_G + \text{Hash}_{z*q} + 6\text{Pair}$. □

Proofs of Proposition 10. In the dynamic operation stage, the main computation overhead is computation of the

corresponding tag, the update process of the root nodes, and the equation verification process.

First of all, the user computes the new authentication tag $\sigma_i' = (H_1(\text{name}\|V_n'\|t_i') \cdot u^{H_3(m_i')})^{skID}$ and sends update request to CSP. The computation overhead is $\text{Hash}_G + 2\text{Exp}_G + \text{Mul}_G + \text{Hash}_{z*q}$.

After receiving the update information, CSP calculates the hash value $H_1(z_i')$ of the updated data block and the hash values of all relevant nodes on the update authentication path. Finally, the new root node R' and the deputy root node $R*'$ are output, the computation cost of this process is 2Hash_G, and then, the update audit proof is sent to the user.

Finally, the user verifies the information provided by CSP, generates the original root R and the deputy root R∗ with $\{H_1(z_i),\Omega_i\}$, and judges whether the equation $e(\gamma, g) = e(H_1(R), pk)$ and (4) $e(\gamma, g) = e(H_1(R^*), pk)$ hold. If they does not hold, the user outputs fail; otherwise, it continues to verify whether CSP correctly executes the update operation, generates root R'' and $R*'$ with $\{H_1(z'_i),\Omega_i\}$, and compares them with the returned R' and $R*'$ If the two values are equal, it means that CSP correctly executed the update operation, then the user calculates $\gamma' = H_1(R')^{skID}$ and $\Gamma' = H_1(R*')^{skID}$, and sends them to CSP to complete the update. The computation overhead of this process is $4\text{Hash}_G + 4\text{Pair} + 2\text{Exp}_G$. Therefore, the total computation cost of the modification process is $7\text{Hash}_G + 4\text{Exp}_G + 4\text{Pair} + \text{Mul}_G \cdot \text{Hash}_{z*q}$, and the insertion process is the same. Since the deletion process does not need to calculate the block tag, the computation cost of the deletion process is $6\text{Hash}_G + 2\text{Exp}_G + 4\text{Pair}$. □

Proofs of Proposition 11. During the legal authority generation process, the user sends the audit authorization $Entrust=(H_1(ID,ID_{TPA}))^x$ to TPA and sends the legal authority verification value $V = g^x$ to CSP, where the cost of Entrust is $|p|$ and the cost of V is $|q|$. In the process of legal authority verification, the third-party auditor needs to send the legal authority to CSP, the communication overhead in this process is $|p|$ bits, so the communication cost in the authorization process is $2|p|+|q|$ bits.

Users upload data blocks, data block tags and signatures to CSP, where the total cost of data blocks is $n|\text{m}|$, and the size of data block label $\sigma_i' = (H_1(\text{name}\|V_n'\|t_i') \cdot u^{H_3(m_i')})^{skID}$ is $n|\text{p}|$. Because Ldasip uses the improved multibranch authentication tree to sign, it needs to transmit the signatures of root nodes γ=(H1(R))skID and deputy root nodes Γ=(H1(R∗))skID, and the communication cost of both is |p|. To sum up, the communication cost in this process is $(n+2)|\text{p}|+n|\text{m}|$.

The communication overhead of the audit challenge response process is divided into audit challenge and audit proof transmission communication overhead in which TPA sends audit challenge chal to CSP, and one audit challenge $\{1, vi\}_{i\in I}$ occupies $(|n|+|q|)$ bits. CSP calculates the audit proof and sends the audit proof to TPA, with one audit proof $P = \{T,\mu,\{H_1(z_i),\Omega_i\}_{i\in I},\gamma,\Gamma_{1\le i\le I}\}$ occupying $(c+3)|p|+|q|$ bits. To sum up, the total communication overhead of this process is $(c+3)|p|+(c+1)|q|+c|n|$ bits.

Data update is divided into three operations: data insertion, modification, and deletion.

In the process of data modification and addition, the main communication overhead includes data transmission process, update proof transmission process, and update signature transmission process. Transmission data update information includes updated signature $\sigma_i' = (H_1(\text{name}\|V_n'\|t_i') \cdot u^{H_3(m_i^*)})^{skID}$ and data block m_i, and the overhead required is $|p|+|m|$. CSP updates root node and authentication structure according to received update information, calculates the update proof $P_{\text{update}} = \{\{H_1(z'_i),\Omega_i\}_{i\in I},\gamma, R',\Gamma_{1\le i\le I},R*'\}$, and sends it to the user in which the communication overhead of P_{update} is $(l+4)|p|$. The user receives the update proof and verifies it. After verification, the signature values of the updated root node and deputy root node are transmitted to CSP, and the overhead of transmitting the updated signature is $2|p|$. To sum up, the communication overhead of data update process is $(l+7)|p|+|m|$. Since the user does not need to transmit updated signatures and the data block to CSP during the deletion operation, the communication overhead of the deletion operation is $(l+6)|p|$. □

Data Availability

All data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declared that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Science Foundation of China under Grant 61502017 and the Scientific Research Common Program of Beijing Municipal Commission of Education under Grant KM201710005024.

References

[1] Tencent.com, "Inventory of top ten cloud downtime accidents in 2018: No mainstream public cloud is spared!," 2020, https://xw.qq.com/cmsid/20181229A0W3UK00.

[2] "Take stock of domestic network security events in 2021," 2022, https://m.thepaper.cn/baijiahao_17013245.

[3] "2022 inventory of global major cyber security events," 2022, https://zhuanlan.zhihu.com/p/496014016.

[4] L. Bai, Y. Zhu, and B. Lu, "Research and progress of cloud data storage security audit," *Computer Science*, vol. 47, no. 10, pp. 290–300, 2020.

[5] Z. Qin, S. Wu, and X. Hu, "Overview of data integrity audit schemes in cloud storage services," *Information Network Security*, vol. 10, pp. 1–6, 2014.

[6] B. Shao, X. Li, and G. Bian, "Overview of research on cloud storage data integrity audit technology," *Information Network Security*, vol. 19, no. 6, pp. 28–36, 2019.

[7] H. Q. Wang, "Identity-based distributed provable data possession in multicloud storage," *IEEE Trans. Services Compute*, vol. 8, no. 2, pp. 328–340, 2015.

[8] Z. Lin, *Research on Identity-Based Data Integrity Verification in Cloud Storage*, Hebei University, Hebei, China, 2017.

[9] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based remote data possession checking in public clouds," *IET Information Security*, vol. 8, no. 2, pp. 114–121, 2014.

[10] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 1, pp. 72–83, 2019.

[11] L. Lin, W. Tan, and Z. Chu, "Summary of outsourcing data integrity audit," *Cyberspace Security*, vol. 11, no. 11, pp. 61–69, 2020.

[12] G. Ateniese, R. Burns, and R. Curtmola, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 598–609, ACM, New York, NY, USA, October 2007.

[13] M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents," *Pasos Revista De Turismo Y Patrimonio Cultural*, vol. 477-494, 2008.

[14] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[15] Y. Tan, W. Fan, and J. Wang, "A cloud data integrity verification scheme supporting privacy protection," *Small Micro Computer System*, vol. 38, no. 12, pp. 2736–2740, 2017.

[16] Y. Wang, *Research on Data Integrity Audit Method Supporting Privacy Protection*, Xi'an University of Architecture and Technology, Xi'an, China, 2018.

[17] H. Baofu, L. Hui, and W. Chuansi, "Blockchain-Based Distributed Data Integrity Auditing Scheme," in *Proceedings of the 2021 IEEE 6th International Conference on Big Data Analytics (ICBDA)*, pp. 143–149, IEEE, Xiamen, China, March 2021.

[18] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security*, pp. 213–222, ACM, Illinois, IL, USA, January 2009.

[19] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[20] E. Daniel and N. A. Vasanthi, "A cost effective dynamic auditing scheme for outsourced data storage in cloud environment," in *Proceedings of the 2017 International Conference on Innovations in Green Energy and Healthcare Technologies (IGEHT)*, pp. 1–5, IEEE, Coimbatore, India, March 2017.

[21] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2402–2415, 2017.

[22] M. Sookhak, F. R. Yu, and A. Y. Zomaya, "Auditing big data storage in cloud computing using divide and conquer tables," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 5, pp. 999–1012, 2018.

[23] T. Shang, F. Zhang, X. Chen, J. Liu, and X. Lu, "Identity-based dynamic data auditing for big data storage," *IEEE Transactions on Big Data*, vol. 7, no. 6, pp. 913–921, 2021.

[24] Y. Yuan, J. Zhang, W. Xu, and Z. Li, "Enable Data Privacy, Dynamics, and Batch in Public Auditing Scheme for Cloud Storage System," in *Proceedings of the 2021 2nd International Conference on Computer Communication and Network Security (CCNS)*, August 2021.

[25] G. Ateniese, R. C. Burns, R. Curtmola et al., "Remote data checking using provable data possession," *ACM Transactions on Information and System Security*, vol. 14, no. 1, pp. 12–34, 2011.

[26] Y. Zhang, H. Zhang, R. Hao, and J. Yu, "Authorized identity-based public cloud storage auditing scheme with hierarchical structure for large-scale user groups," *China Communications*, vol. 15, no. 11, pp. 111–121, 2018.

[27] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, and J. Ma, "Data integrity auditing without private key storage for secure cloud storage," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1408–1421, 2021.

[28] M. Zhao, Y. Ding, and Y. Wang, "Cloud data security storage scheme for designated auditors," *Information Network Security*, vol. 11, pp. 66–72, 2018.

[29] S. Hiremath and S. Kunte, "A Novel Data Auditing Approach to Achieve Data Privacy and Data Integrity in Cloud computing," in *Proceedings of the 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, pp. 306–310, Mysuru, India, December 2017.

[30] Y. Zhang, Yu Jia, H. Rong, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 608–619, 2020.

[31] Y. Zha, S. Luo, J. Bian, and W. Li, "Multi-user and multi-copy data possession proof scheme based on multi-branch authentication tree," *Journal of Communications*, vol. 36, no. 11, pp. 80–91, 2015.

[32] P. J. Denning, "The locality principle," *Communications of the ACM*, vol. 48, no. 7, pp. 19–24, 2005.

[33] Y. Zhao, S. Wang, S. Wu, and X. Hu, "A proxy remote data integrity audit protocol," *Journal of University of Electronic Science and Technology of China*, vol. 45, no. 01, pp. 80–85, 2016.

[34] W. Shen, Ye Su, and H. Rong, "Lightweight cloud storage auditing with deduplication supporting strong privacy protection," *IEEE Access*, vol. 8, pp. 44359–44372, 2020.