Secure Sharing Data Integrity Audit Scheme with Dynamic User Groups in Cloud Storage

Jia-Xian Liu, Hui Huang*, Qun-Shan Chen

School of Computer Science
Minnan Normal University, Zhangzhou 363000, China
jia1598149973@163.com, hhui323@163.com, xianmensam@163.com

Zhen-Jie Huang

Fujian Key Laboratory of Granular Computing and Application
Minnan Normal University, Zhangzhou 363000, China
zjhuang@mnnu.edu.cn

*Corresponding author: Hui Huang

ABSTRACT. *With the wide application of cloud storage, users with limited resources send outsourced data to the cloud service provider. Shared data storage has become a vital application form in some cloud storage scenarios. Many existing shared data integrity audit schemes don't consider the user's access rights. The revoked user can still access the data in the group. Moreover, they cannot support the dynamism of the data, nor can they support data privacy and user identity protection from infringement by the validator. We propose a safe and feasible shared data audit scheme to solve several of the problems above. In our scheme, once the group user is revoked, he has no permission to access and download data. We also design a dynamic multi-tree data storage structure to realize the increase, deletion, and search of dynamic user groups and data. Not only that, but our scheme also supports identity privacy protection for group users and data privacy protection. Experimental data show that our scheme is more efficient and safer than previous schemes. Especially in calculating tag generation, we reduces the computing overhead by 50 %.*
Keywords: Sharing data integrity audit, Data privacy protection, Identity privacy protection, Dynamic user groups

1. **Introduction.** As one of the most popular modes of online storage, cloud storage provides users with massive storage space and robust computing power [1–3]. Users with limited resources upload data to the Cloud Service Provider (CSP) for storage. Still, it means that the user needs to hand over its data management rights to the CSP, an untrusted entity in real life [4]. Since the CSP may corrupt data for personal gain, verifying the integrity of the data stored on the CSP is important.

In recent years, personal data integrity audit does not meet the needs of real life, and shared data integrity audit has become a new hotspot [5–7]. Unlike personal data audit schemes, group users can share their data in the cloud through data-sharing services, reducing the burden on users for local data storage. In the shared data scenario, the group user uploads data to the CSP, and other users within the group can access and download the data. When the data is shared with multiple users in the group, some issues must be addressed. For example, group users need to perform joining and exiting operations

in shared data, how to achieve group user dynamics is an essential problem. Once the group user is revoked, he cannot access and download data, and his identity information becomes invalid. Many researchers have proposed different schemes to solve the above problems. Some schemes only support data privacy protection but do not support other attributes [8–17]. Some schemes do not support dynamic shared data updates [18–20], which is crucial when users frequently update data to meet various application needs. In addition, validators may be curious about data identity information [21, 22], and we also need to consider identity privacy protection. So, building a secure and feasible shared data integrity audit scheme that supports data privacy, identity privacy, data dynamics, and the group user dynamic update remains an important challenge.

Contribution. In this paper, we propose a new integrity audit scheme for cloud storage shared data with dynamic user updates, supporting data privacy and user identity privacy protection. The contribution of our paper can be summed up as the following three points:

(1) We propose a new dynamic multi-tree storage structure, which the CSP maintains to achieve dynamic data and dynamic group management, effectively improving data space utilization.

(2) The group administrator maintains an anonymous identity table named IAT in our scheme. Only the group administrator knows the real identity information of group users, effectively improving the user's identity privacy protection.

(3) To protect user data privacy, we encrypt the data-proof information generated by the CSP to prevent the verifier from obtaining valid user information in the audit stage. Our scheme has stronger security than constructing linearly correlated data-proof information.

1.1. Realted work.
How to achieve data integrity auditing has always been a hot issue [23]. The most primitive auditing scheme is that the data owner (DO) downloads the complete data from the CSP and verifies the integrity of the blocks in turn, which brings enormous communication costs to the DO. The Provable Data Possession (PDP) [24] scheme only needs to perform sampling verification on blocks, decreasing the computational overhead of the DO. In the PDP scheme, the DO first divides the data file that needs to be uploaded into blocks, generates data block tags for each, and stores the data blocks and data block tags together in the CSP. To verify data integrity, the DO usually entrustes the third-party auditor (TPA) for validation. The TPA randomly selects some data block indices to challenge the CSP, and the CSP generates the correct proof information and sends it to the TPA. The TPA returns the verification result to the DO. Subsequently, many effective schemes are proposed based on the PDP. Due to the low computational power of the DO, entrusting a reliable TPA to perform verification tasks is commonly used in most audit schemes [25–30].

In the group data shared scenarios, group users preprocess and upload their data to the CSP. Other users in the group can also download and access data from the CSP. However, the data file for shared data is composed of data from each group of users, and different block tags are generated by other group users, which makes data integrity verification under shared data more complex. The existing schemes are focused on verifying personal data's integrity and are unsuitable for auditing user-group shared data. Recently, researchers have designed many new schemes for shared data scenarios. However, there are still some security issues.

Since the DO does not allow their private data to be leaked to the TPA, the privacy protection attribute requires them not to obtain user data information. Many schemes for data privacy protection have been proposed. Initially, Wang et al. [8] selected a random number as a masking factor to encrypt data-proof information to protect data

privacy, preventing the TPA from being curious about and obtaining the data. Ji et al. [9] proposed a secure and efficient identity-based cloud storage scheme. The scheme supports data privacy protection. Shen et al. [10] designed a remote shared data integrity audit model, which uses a purification program to conceal sensitive data in the shared data. Still, it requires establishing a safe passage between the user and the purification program. Xu et al. [11] proposed a privacy-protected shared data integrity audit scheme (PP-CSA) to address the issue of potential data leakage. Only authorized users can access data. Although these schemes solve the problem of data privacy protection, they are not well taken into account in data dynamics schemes.

Yang et al. [18] proposed designing a new cloud data-sharing framework that supports data privacy protection, dynamic revocation, and user identity traceability but cannot achieve dynamic data management. Yan et al. [19] proposed an identity-based shared data integrity audit scheme to realize user identity traceability and prevent group users from maliciously tampering with data. They add a rights distribution center for effective user identity management and use data blinding technology to upload encrypted data. The shared audit scheme proposed by Yan and Gui [20] considers anonymous protection of the user's identity, preventing auditors from accessing the relationship between the data and the user and ensuring the privacy of user identities. However, this scheme cannot support dynamic operations of data and user groups. In addition, all three of the above schemes have security issues. Even if the CSP does not store data, it can still respond to the verification of the TPA.

Although some shared data integrity audit schemes have been proposed to achieve data privacy protection, they cannot achieve identity privacy protection. Since different users within the group generate the tags of shared data blocks, validators can determine the importance of data by looking at how often the data block is modified. Rao et al. [21] proposed a certificateless shared data audit scheme (CLPPPA), which can realize the data update and group user revocation. However, it cannot achieve identity privacy protection. Yang et al. [17] proposed a shared data scheme based on certificateless encryption. Although this scheme claims to protect user identity, the TPA can obtain the relationship between data and the user's public key during the audit process. Therefore, it has yet to achieve user privacy protection. Using the idea of proxy re-encryption for reference, Yu et al. [22] proposed to use key sharing among group users to realize dynamic user revocation. In this way, the auditor can verify the integrity and correctness of the data without knowing the user's identity information. However, this scheme requires users to have a strong computing power. It also does not support dynamic auditing of the data. Hu et al. [31] proposed a new network authentication scheme in the Internet of Things environment, which supports data privacy protection and significantly reduces the damage caused by key exposure. They also introduced a secure and efficient encryption technology to enable secure communication over insecure channels. To protect the communication security between the user and the server, Wu et al. [32] proposed a new enhanced authentication protocol and proved its security given the shortcomings of previous schemes. Thakur et al. [33] designed a secure identity authentication scheme, which uses a fuzzy extractor to protect the user's identity information, realizes the interaction between the user and the server, and reduces the calculation and communication overhead of the system.

Users may join or exit shared groups anytime in real life, so supporting dynamic user operations is crucial. For security reasons, once a group user is revoked or leaves, they can no longer have rights to access shared data. However, most shared data audit schemes [34, 35] do not involve user revocation. Wang et al. [34] used group signatures to generate block tags, which prevented verifiers from obtaining user data information and achieved data privacy protection. However, this scheme does not support user revocation. Subsequently,

they proposed using a ring signature to protect data privacy, but this scheme still does not support group user revocation. Li et al. [35] proposed a secure data integrity audit scheme for dynamic group users to support truly effective user revocation. This scheme supports updating data files and revocation of group users, and it can also resist the collusion of malicious attackers and revocation users. In the scheme proposed by Yu et al. [36], only authorized personnel can obtain user identity information, and the public audit can be realized based on protecting privacy. On this basis, the scheme also uses an idea called asymmetric group key protocol to implement the challenge-audit mechanism. However, the scheme only discussed this point, and in-depth research on data privacy protection needs to be conducted. Zhang et al. [37] proposed a new key generation strategy that can achieve efficient user revocation. When a user is retracted from a group, all the users who have not been retracted can update their private keys through private key update technology without reorganizing the user's identity information, and the audit is still valid. Although the scheme designed by Zhang et al. achieved user revocation, their proposed shared data scheme cannot support dynamic data updates. Huang et al. [38] used the Arithmetic Span Program as the access policy to add indirect undo and ciphertext update to the scheme and proposed a revocation storage ciphertext policy attribute encryption algorithm, which can prevent the revoked users from accessing data without authorization. These schemes support dynamic groups, allowing users to join and leave the user group at any time, but some problems remain.

1.2. **Organization.** The rest of this paper is structured as follows. We describe preparations in section 2. In section 3, we introduce the system model, threat model, and security objectives. In section 4, we present the notations, the overview, and the specific scheme. The feasibility and safety of the scheme are analyzed in section 5. In section 6, we design experiments to evaluate the scheme's efficiency, and in section 7, we summarize this paper.

2. **Preliminaries.**

2.1. **Bilinear map.** Set G_1 and G_2 be two multiplicative cyclic groups with the same big prime order q. g_1 and g_2 are generators of the group G_1, defining $e : G_1 \times G_1 \to G_2$ is a bilinear map with the following three properties:
(1) Computability: For any $g_1, g_2 \in G_1$, there is an efficient algorithm that calculates the value of $e(g_1, g_2)$.
(2) Bilinearity: For all $g_1, g_2 \in G_1$ and $a, b \in_R Z_q^*$, $e(g_1{}^a, g_2{}^b) = e(g_1, g_2)^{ab}$.
(3) Nondegeneray: The existence of $g_1, g_2 \in G_1$, $e(g_1, g_2) \neq 1$.

2.2. **Difficult problem assumptions.** Discrete Logarithm problem(DL): Set G_1 is a multiplicative cyclic group, g as the generator of G_1. Given (g, g^a), If we don't know the value of $a \in Z_q^*$, it's hard to calculate the specific value of a. For any adversary A, the probability of A solving the DL problem is negligible and can be expressed by the following equation (1):

$$Pr\left[A_{DL}(g, g^a) = a : a \xrightarrow{R} Z_q^* \right] \leq \epsilon \tag{1}$$

2.3. **A new dynamic multi-tree storage structure.** The binary tree is an essential type of tree structure. Each non-leaf node can have at most two child nodes. Child nodes are divided into left and right nodes in a binary tree form. The complete binary tree is a special kind of binary tree. Researchers use linked lists and tree structures to store data, and binary tree is one of the storage modes. In data integrity audit schemes, we usually use the Merkle tree [39] as data storage structures, a complete binary tree type. However,

constructing binary trees does not maximize the use of storage space. In our research, we use multiple fork trees to store data. Not only that, but we also design the upgrade and downgrade strategy of multi-tree to improve the efficiency of data queries.

Assuming the stored data has reached 2^k, k refers to the number of tree layers. If we want to add another leaf node to the group, it means that the number of layers of the tree is increased by one, leading to binary tree storage space being wasted. The emergence of multi-trees solves the problems of low storage space utilization of binary trees. However, if the amount of data stored by the multi-tree reaches a certain amount, the query efficiency decreases.

We design a dynamic multi-tree storage structure to use storage space more and achieve efficient search. Its leaf nodes store the user's data, connecting the anonymous identifier UID of the group of users, and the non-leaf node stores the number of leaf nodes under this node. In addition, we propose a multi-tree upgrade and demotion strategy to achieve the dynamic data update. The revoked user changes the corresponding leaf node value, which will not affect other users' data distribution in the multi-tree. The following introduces inserting, querying, and deleting data in dynamic multi-trees.

Data insertion: We use the dynamic multi-tree to achieve data insertion. For example, in Figure 1, we insert new data n_5 into the multi-tree. The parent node corresponding to the leaf node value increases by 1, from 2 to 3, and the root node value from 4 to 5.

Upgrade strategy: When the data number reaches 2^k, where k denotes the number of
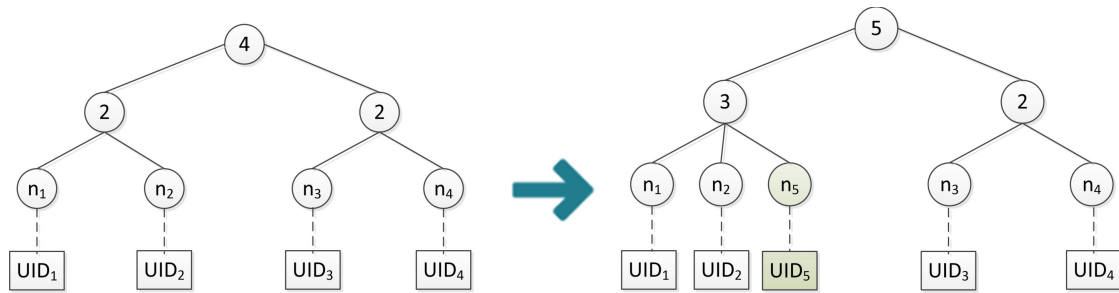

FIGURE 1. Data insertion structure diagram

layers in the dynamic multi-tree, the multi-tree needs to be upgraded, and the number of layers will increase accordingly. The specific structure is shown in Figure 2. For example, if the number of data is 8 and the number of layers of the multi-tree is 3, we need to upgrade the multi-tree, and the number of layers increases by 1 to become the 4-layer tree.
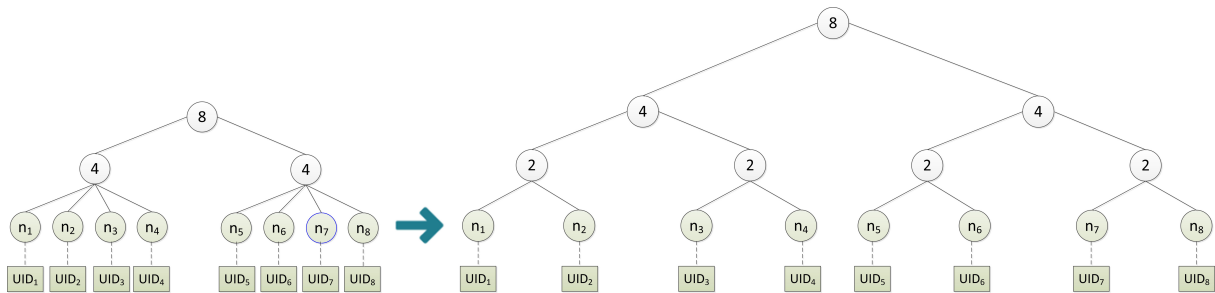

FIGURE 2. Upgrade strategy structure diagram

Data queries: When inserting, deleting, and modifying data, we need to find the location of the data. First, we need to determine whether the value of nodes in layer 2

is less than 4. The new data can be inserted if the value is less than 4. We can divide it into two cases when looking up data. If the data index $i \leq 2^{k-1}$, we need to find the $\lceil i/2 \rceil$ (Roundup) node in layer 2. For example, if we want to find n_3 in Figure 1, we query the second node in layer 2 and then the leaf node. If the data index $i > 2^{k-1}$, we only need to find the $\lceil (i - 2^{k-1})/2 \rceil$ (Roundup) node in layer 2 and then the leaf node. For example, if we want to find n_5 in Figure 1, we query the first node in layer 2 and then sequentially from the leaf nodes.

Data deletion: We use the dynamic multi-tree to delete data. For example, in Figure 3, we delete the data n_2 in the multi-tree, and the parent node value corresponding to the leaf node decreases by 1, from 2 to 1, and the root node's value from 4 to 3.
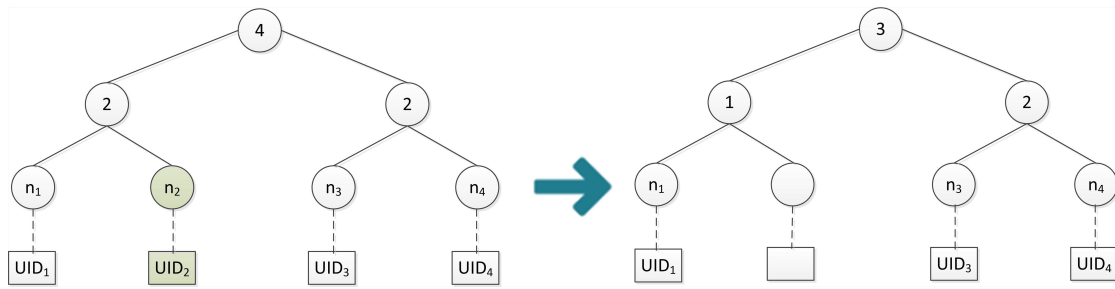

FIGURE 3. Data deletion structure diagram

Demotion strategy: When the data number is reduced to $2^{k-2} - 1$, the dynamic multi-tree needs to be downgraded, and the number of layers is reduced accordingly. The specific structure is shown in Figure 4. For example, if the number of data is 8, and the number of layers of the multi-tree is 4, we want to delete the data n_1, n_3, n_6, n_7, n_8, the number of layers is reduced to 3.
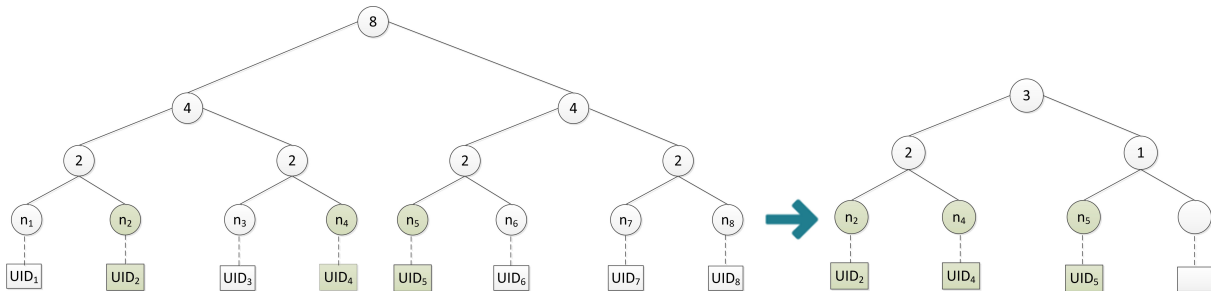

FIGURE 4. Demotion strategy structure diagram

2.4. **Identity anonymity table.** To realize the privacy protection of group user identity, the group administrator stores the real user identifier RID and the anonymous user identifier UID for each authorized group user in the Identity Anonymity Table (IAT), where $UID = H(RID)$. The RID is used to apply for user private keys from the KGC. The CSP uses the UID to store user information related to data, which can prevent the CSP and the TPA from obtaining valid user information in the audit process. In addition, the group administrator also stores the private key sk of group users in IAT, which is convenient for later computing data tags. The specific structure of IAT is shown in Table 1.

3. **System model and security goals.**

TABLE 1. Identity anonymity table

No	Real user identifier	Anonymous user identifier	The private key of the group user
1	RID_1	UID_1	sk_1
2	RID_2	UID_2	sk_2
3	RID_3	UID_3	sk_3
$\dots$	$\dots$	$\dots$	$\dots$
j	RID_j	UID_j	sk_j

3.1. **System model.** The system comprises five entities: Key Generation Center, Cloud Service Provider, Group Administrator, Group User, and Third-Party Auditor. The system model diagram is shown in Figure 5.

- Key Generation Center (KGC): The KGC is a trusted authority responsible for generating security parameters and the user's private key for the system. The user's private key is distributed to the corresponding group user and administrator through the secure channel.
- Cloud Service Provider (CSP): The CSP is untrusted. He is responsible for storing data uploaded by group users. He has vast data storage space and computing power to generate proof information. However, he may have needed to store complete data and be curious about users' identities.
- Group Administrator (GM): The GM is a trusted entity that manages access rights for group users and maintains the anonymous table of group user identities.
- Group User: Group users are all users who share data within a group. They are trusted entities. Each group user can be a data uploader. They are responsible for uploading the data information to the CSP for storage, and they also are authorized to download the data information of other group users from the CSP.
- Third-Party Auditor (TPA): The TPA is a semi-trusted entity. It may be an institution or an independent individual. The GM commissions him to conduct an audit task, and he honestly verifies the correctness of the data-proof information. However, he may be curious about the identity information of the group users. In addition, he may attempt to obtain valid information about the data from the proof information.

3.2. **Threat model.** The CSP is not credible, and he may lose or damage data, which human operations or natural disasters cause. Infection of a system or device with a virus may also cause data corruption. To conceal the fact that the user's data is corrupted, the CSP can launch the following attacks:

- Forgery attacks. The CSP may use other data blocks and tags to forge proof information to pass the verifier's verification.
- Replace attacks. The CSP may use previously generated proof information to pass the verification.
- Collusive attacks. The CSP may attempt to collude with the TPA to pass the verification. The CSP may also attempt to obtain user identity information.

The TPA is not trustworthy. During the integrity audit process of third-party audits, the TPA may obtain valid information from the proof information and infer the identity information of the group users.
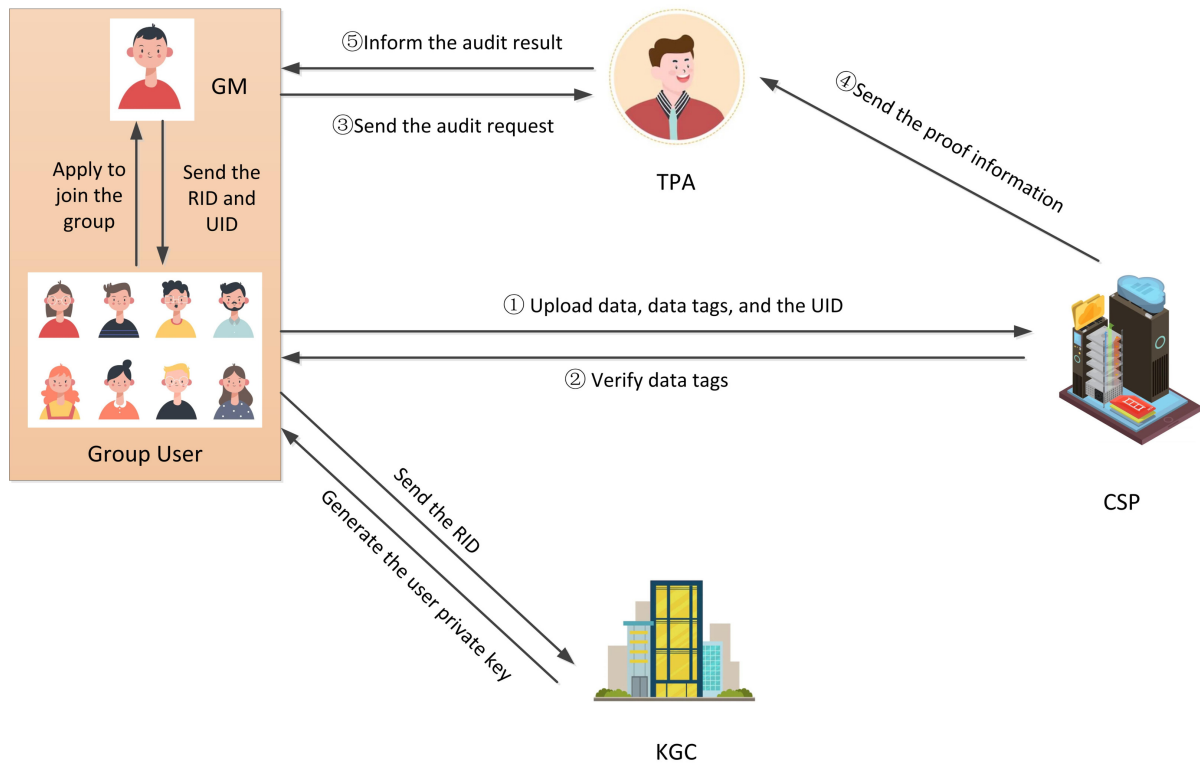
FIGURE 5. System model diagram

In cloud data sharing, unauthorized users may illegally obtain or obtain data beyond their authority.

3.3. Security goals.

- Data integrity: If the CSP does not store correct and complete data, the proof information he generates cannot pass the audit of the verifier.
- Data privacy protection: Unauthorized users and the TPA cannot obtain outsourced data from the CSP.
- Group user dynamics: Both group user registration and revocation should be implemented securely and effectively.
- Identity privacy protection: Except for the GM, nobody can obtain the real identity information of group users.
- Non-interactivity: Non-interactivity reduces the communication overhead between the TPA and the CSP. The CSP can autonomously generate proof information based on public information and the number of challenge blocks.
- Dynamic shared data: The shared data can be added, deleted, modified, and checked efficiently.

4. The proposed scheme.

4.1. Notation.
The notation description of this paper is detailed in Table 2.

4.2. Overview.
The system process diagram is shown in Figure 6. In the setup phase, group users first send an application to the GM to join the group, which generates the real user identifier RID and anonymous user identifier UID for authorized group users. The

TABLE 2. Notations description

Notation	Description
λ	A security parameter
q	A big prime order
G_1 and G_2	Two multiplicative cycilc groups of same order q
g	The generating element of group G_1
e	A bilinear mapping
H and h	Two different hash functions
ϕ	A pseudo-random function
RID	Real user identifier
UID	Anonymous user identifier
F_j	Group user u_j uploads the file
m_{ji}	One of the data blocks that make up the file F_j
σ'_{ji}	The GM generates the partial block tag of the data block m_{ji}
σ_{ji}	The group user generates the entire block tag of the data block m_{ji}
θ_j	Aggregation of data tags for data file F_j
c	The number of challenge blocks
τ	Public information for generating challenge information
P	Proof information

GM sends the RID and UID to group users and sends the UID to the CSP for storage. Then, the group user sends the RID to the KGC for registration. The KGC generates the corresponding private key according to the RID and sends it to the corresponding group user and the GM through the secret passage. In addition, the KGC generates system public parameters $params$ and master private key msk and sends msk to the GM for saving. When group users want to upload data, they first send the RID and data file F_j to the GM. The GM first checks the validity of the RID in the IAT, and the GM generates partial tags based on data block m_{ji} and sends them to the group user. Next, the group user uses the private key to generate a complete block tag and then sends the anonymous user identifier, the data, and the block tags to the CSP for storage.

In the tag verification stage, the CSP first checks whether the Block-Id tree has the UID. If so, he further verifies the correctness of the data block tags. Otherwise, he refuses to receive the data and returns it to the user.

In the audit stage, when other users want to download outsourced data from the CSP, they first apply to the GM. After the application is approved, the GM entrusts the TPA to conduct an audit. The CSP generates proof information based on public information and the number of blocks sent by the GM and sends it to the TPA for verification. After verification, the TPA informs the GM of the audit results. If the data is complete, the user applies to the CSP to download the data. The CSP checks whether the Block-Id tree has the user's identity information, and if so, the CSP sends the data to the user.

During the group users' registration and revocation phase, new users apply to the GM to join the group. They obtain the real user identifier RID and the anonymous user identifier UID from the GM, and the GM sends the UID to the CSP for saving. Next, the new user registers in the KGC using RID, and the KGC generates the private key to send to the authorized user. Suppose a user wants to exit the group. Firstly, the GM deletes the user's identity information in the IAT and entrusts the group user who has yet to be revoked to generate a new tag for the data block and send the new tag and the UID to the CSP. After receiving the information, the CSP generates a tag aggregation

for data files based on block tags. Then, the CSP modifies the block tag aggregation and the user's anonymous identifier in the Block-Id tree. Therefore, revoked users cannot access shared data.
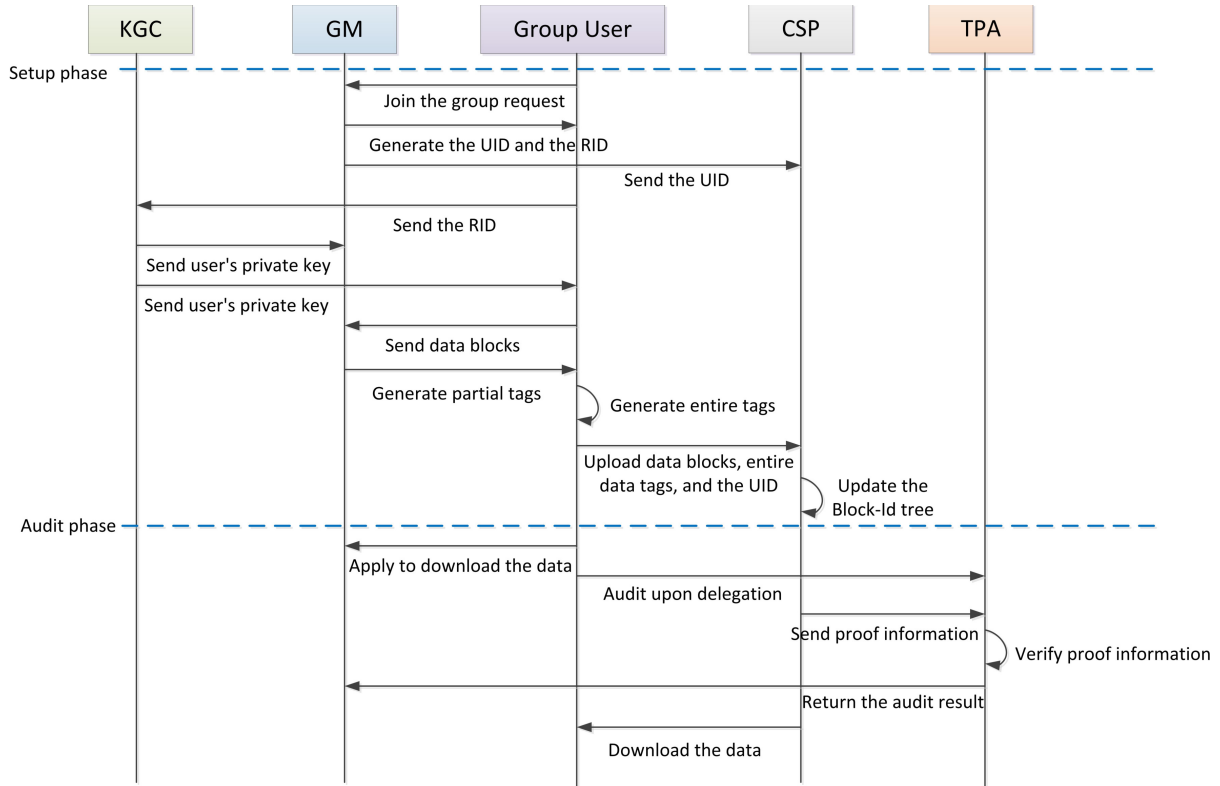

FIGURE 6. System process diagram

4.3. **The specific scheme. Setup phase:** The setup phase includes the key generation algorithm, the tag generation algorithm, and the tag verification algorithm. Suppose there are U users in the shared user group. Each group user is represented by u_j $(1 \le j \le U)$, the group user that is also the data uploader. The real identifier of the group user is RID_j, and the anonymous identifier is UID_j, where $UID_j = H(RID_j)$. At this stage, the KGC initializes the system parameters *params* and the master private key *msk*. Then the KGC generates their private keys sk_j based on the group users' real identifiers RID_j. The KGC transmits the group user's private key sk_j to the corresponding group user and the GM over a secure channel. Group users send data insert requests $\{RID_j, F_j\}$ to the GM. The GM splits the file F_j into data blocks m_{ji} and generates partial tags σ'_{ji} based on the data blocks. Group users use their private keys sk_j to generate entire data block tags σ_{ji}, then send $\{m_{ji}, \sigma_{ji}, UID_j\}$ to the CSP for storage. The CSP verifies the user's identifier UID_j and block tags σ_{ji}. After the verification is passed, the CSP calculates the tag aggregation θ_j for data files based on block tags. Then the CSP stores m_{ji} and σ_{ji}, and he stores $\{\theta_j, UID_j\}$ in the Block-Id tree. Otherwise, he returns the data to the user.

- $KeyGen(1^\lambda) \to (params, msk)$. The KGC runs the key generation algorithm. The KGC selects two multiplicative cyclic groups G_1 and G_2 with the same big prime order q, a bilinear map $e : G_1 \times G_1 \to G_2$, two safe hash functions $H : \{0,1\} \to G_1, h : \{0,1\} \to Z_q$, a pseudo-random function $\phi : \{0,1\}^* \to [1,n]$. Then the KGC randomly selects $x, x \in Z_q$, and sets the primary private key $msk = x$ and

the primary public key $mpk = g^x$. Finally, the KGC sets the system parameters $params = (q, g, G_1, G_2, e, H, h, \phi, mpk)$ to the public and sends the master key msk to the GM.

- $Extract(RID_j, msk) \rightarrow sk_j$. The KGC runs the user's private key generation algorithm. Upon receiving the real identifier RID_j from the group user $u_j(1 \le j \le U)$, the KGC calculates the private key $sk_j = H(RID_j)^x$ and sends it to the group user and the GM through a secure passage.

- $TagGen(F_j, sk_j) \rightarrow \Phi_j$. The group users run the algorithm. F_j refers to the data file composed of n data blocks m_{ji}, $F_j = \{m_{ji}\}_{1 \le i \le n}$, where $m_{ji} \in G_1$. If the group user u_j wants to upload a data file F_j, he needs to send $\{RID_j, F_j\}$ to the GM. The GM first looks for the corresponding private key in IAT and then calculates $sk'_j = x - sk_j$. Then the GM divides the data file F_j into n blocks m_{ji}, where $1 \le i \le n$, generates partial data tags $\sigma'_{ji} = (H(RID_j) \cdot m_{ji})^{sk'_j}$ and sends them to the group user. The group user first generates an entire data tag $\sigma_{ji} = \sigma'_{ji} \cdot (H(RID_j) \cdot m_{ji})^{sk_j} = (H(RID_j) \cdot m_{ji})^x$ based on partial tags generated by the GM, and the block tag set is $\Phi_j = \{\sigma_{ji}\}_{1 \le i \le n}$. The group user sends $\{F_j, \Phi_j, UID_j\}$ to the CSP for storage.

- $TagVerify(F_j, \Phi_j, params, UID_j) \rightarrow \{0, 1\}$. The CSP runs the tag verification algorithm. The CSP receives the data upload request from the user. He first looks for the validity of the UID_j in the Block-Id tree and then verifies the correctness of the data block tags according to equation (2).

$$e(\sigma_{ji}, g) = e(H(RID_j) \cdot m_{ji}, mpk) \tag{2}$$

If the equation does not hold, it is rejected. Otherwise, the CSP first stores m_{ji} and σ_{ji}. Then the CSP calculates the tag aggregation θ_j, where $\theta_j = \sigma_{j1} \times \sigma_{j2} \times \cdots \times \sigma_{ji}$ and stores θ_j in the Block-Id tree.

Audit phase: The audit stage includes the proof information generation algorithm and the proof information verification algorithm. The audit phase is performed in two cases: the GM commissions the TPA to conduct regular audits, and the GM commissions the TPA to audit before each data download. At this stage, the CSP periodically generates challenge information based on the public information τ in the blockchain and the number of challenge blocks c and then sends proof information to the TPA for integrity verification. The TPA verifies the correctness of the proof information and returns the verification result to the GM.

- $ProofGen(F_j, \Phi_j, params, \tau, c) \rightarrow P$. The CSP runs the ProofGen algorithm. τ refers to the public information in the blockchain, including the current block header information. τ cannot be known and modified by anyone in advance, which will change over time with each audit. The CSP gets the number of challenge blocks c from the GM, where $1 \le c \le n$. The CSP generates challenge information based on τ and c. For $i \in [1, c]$, the CSP calculates the challenge block index $I = \{s_{ji}\}$ and random value $\{v_{ji}\}_{1 \le i \le c}$, where $s_{ji} = \phi(\tau||ji)$, $v_{ji} = h(\tau||ji)$, the CSP calculates $\mu = \prod_{i \in I} m_{ji}^{v_{ji}}$ and $\sigma = \prod_{i \in I} \sigma_{ji}^{v_{ji}}$. Finally, the CSP sends the information $P = \{\mu, \sigma, \tau\}$ as proof to the TPA.

- $ProofVerify(P, params) \rightarrow \{0, 1\}$. The TPA performs the validation algorithm, which first verifies the validity of the public information τ, then generates the challenge information according to τ and c. Finally, verify the correctness of data tags according to equation (3).

$$e(\sigma, g) = e(\mu \cdot \prod_{i=s_{j1}}^{s_{jc}} (H(RID_j))^{v_{ji}}, mpk) \tag{3}$$

Group user dynamic phase: The group user dynamic phase includes group users' registration and revocation algorithms. At this stage, authorized group users perform joining or exiting operations. The following is an introduction to two operations.

- $EnrollMen(u_{j+1})$. The GM runs the algorithm. If the new group user u_{j+1} wants to join the group, apply to the GM and obtain the corresponding user identifier information. The GM stores the RID_j and UID_j in the IAT and sends UID_j to the CSP for later verification. The CSP updates the tree accordingly. For example, if a new user u_5 wants to join the group, he first applies to the GM. Then he obtains the real identifier RID_5 and the anonymous identifier UID_5, and the GM sends UID_5 to the CSP. After receiving the information, the CSP inserts UID_5 into the Block-Id tree, as shown in Figure 7. When the number of users joining the group reaches 2^k, where k represents the number of layers of the multi-tree, the dynamic multi-tree is upgraded.
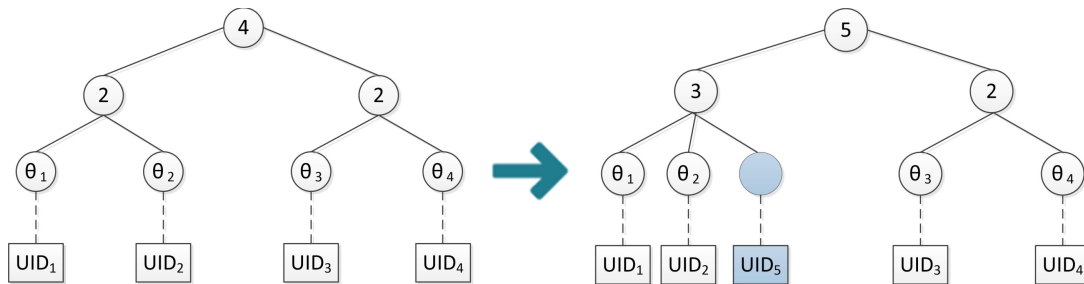

FIGURE 7. Register user structure diagram

- $RevokeMen(u_j)$. The GM runs the algorithm. If the group user u_j wants to be withdrawn or revoked, the GM first deletes the user's identifier in the IAT. Then GM selects an unrevoked group user to regenerate block tags σ_{ji}'' of the revoked user's data. Finally, the GM sends the revoked user identifier, the newly generated data block tags, and the anonymous identifier of the user who regenerated the tags to the CSP. The CSP updates the Block-Id tree based on the information initiated by the GM. For example, suppose we want to revoke a group user u_2. In that case, the GM downloads the revoked user's data from the CSP and delegates the non-revoked user u_1 to regenerate the tag $\sigma_{2i}'' = (H(RID_1) \cdot m_{2i})^x$ for the revoked user's data. The GM sends the data block m_{2i}, generated the data tag σ_{2i}'' and UID_1 to the CSP for storage. After receiving the information, the CSP replaces θ_2 with θ_2'' in the Block-Id tree, where $\theta_2'' = \sigma_{21}'' \times \sigma_{22}'' \times \cdots \times \sigma_{2i}''$, as shown in Figure 8. The dynamic multi-tree is demoted if the number of unrevoked group users is reduced to $2^{k-2} - 1$.

Dynamic data update phase: The dynamic data update phase includes data insertion, modification, and deletion. In our scheme, the CSP stores the data tags aggregation and the user's anonymous identifier in the Block-Id tree. Here, we use tag aggregation changes to represent updates to the data. Adding, deleting, and modifying data is essentially updating the tag aggregation. For example, authorized user u_2 wants to add a data
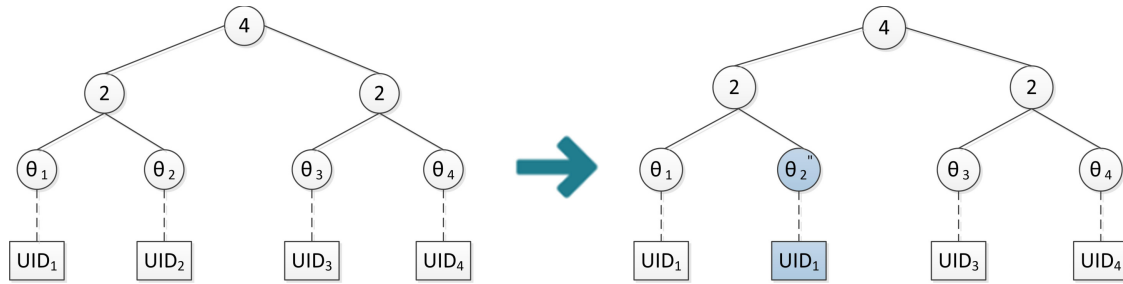
FIGURE 8. Revoke user structure diagram

block, delete a data block, and modify a data block are all updates the tag aggregation θ_2 to θ_2^*, as shown in Figure 9.

- Data tag insertion. Given $F_j = \{m_{ji}\}_{1 \leq i \leq n}$, let's assume that the group user u_j needs to insert a new block $m_{j(n+1)}$, the GM generates the partial tag $\sigma'_{j(n+1)} = (H(RID_j) \cdot m_{j(n+1)})^{sk'_j}$, and the group user generates an entire tag based $\sigma_{j(n+1)}$ on it. Then the group user sends the data insertion request $\{RID_j, jn, I, m_{j(n+1)}, \sigma_{j(n+1)}\}$ to the GM, I represent the operation type as data tag insertion. The GM first verifies the RID_j of validity and sends $\{i, I, m_{j(n+1)}, \sigma_{j(n+1)}\}$ to the CSP. After the CSP receives the information, it proves the correctness of the generated data tags, regenerates aggregation tags θ_j and then updates the Block-Id tree, where $\theta_j^* = \sigma_{j1} \times \sigma_{j2} \times \cdots \times \sigma_{jn} \times \sigma_{j(n+1)}$. For example, for the authorized user u_2, u_2 wants to insert the new block m_{25}, the CSP first stores the data m_{25} and the data tag σ_{25}, then the CSP updates the Block-Id tree based on the information, updates the tag aggregation θ_2^* in the Block-Id tree, where $\theta_2^* = \sigma_{21} \times \sigma_{22} \times \cdots \times \sigma_{24} \times \sigma_{25}$.

- Data tag modification. Given $F_j = \{m_{ji}\}_{1 \leq i \leq n}$, let's assume that the group user u_j wants m_{jn}^* to replace the block m_{jn}, the group user first generates the block tag $\sigma_{jn}^* = (H(RID_j) \cdot m_{jn}^*)^x$, then the group user u_j sends $\{RID_j, jn, M, m_{jn}^*, \sigma_{jn}^*\}$ to the GM, where M represents the operation type as data tag modification. The GM verifies the validity before sending the modification request $\{ji, M, m_{jn}^*, \sigma_{jn}^*\}$ to the CSP. After receiving the modification request, the CSP replaces the stored block m_{jn} with m_{jn}^* and the stored tag σ_{jn} with σ_{jn}^*, then updates the Block-Id tree to replace θ_j with θ_j^*, where $\theta_j^* = \sigma_{j1} \times \sigma_{j2} \times \cdots \times \sigma_{jn}^*$. For example, if the authorized group user u_2 wants to modify the data m_{22}, the CSP changes m_{22} to m_{22}^* and σ_{22} to σ_{22}^*, the CSP replaces θ_2 with θ_2^* in the Block-Id tree, where $\theta_2^* = \sigma_{21} \times \sigma_{22}^* \times \cdots \times \sigma_{2n}$.

- Data tag deletion. Given $F_j = \{m_{ji}\}_{1 \leq i \leq n}$, let's assume that the group user u_j wants to delete the data m_{jn}, u_j first sends the request $\{RID_j, jn, D\}$ to the GM, where D indicates that the operation type is data tag deletion. Then, the GM queries whether the block belongs to the user. If not, reject the delete request. Otherwise, the GM sends requests $\{jn, D\}$ to the CSP. After the CSP receives the message, he deletes the block m_{jn}, σ_{jn} and regenerates tags aggregation θ_j^* in the Block-Id tree, where $\theta_j^* = \sigma_{j1} \times \sigma_{j2} \times \cdots \times \sigma_{j(n-1)}$. For example, if the authorized group user u_2 wants to delete data m_{22}, the CSP deletes m_{22}, the data tag σ_{22} and regenerate tags aggregation θ_2^* in the Block-Id tree, where $\theta_2^* = \sigma_{21} \times \sigma_{23} \times \cdots \times \sigma_{2i}$.
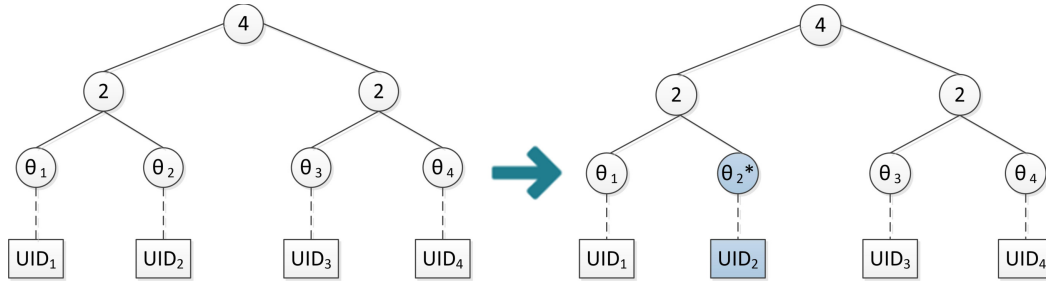
FIGURE 9. Data tags aggregation update structure diagram

5. **Security analysis.** We analyze the scheme's security from six aspects: audit correctness, audit reliability, identity privacy protection, data privacy protection, non-interactivity, and dynamic shared data.

5.1. **Audit correctness.** If the cloud service provider is honest and trusted, storing all the correct data truthfully, the proof information generated by the CSP can pass the verification of the TPA. According to the basic properties of bilinear mapping, the TPA can derive the right-hand equation from the left-hand equation to verify data integrity.

$$
\begin{aligned}
e(\sigma, g) &= e(\prod_{i=s_{j1}}^{s_{jc}} (H(RID_j) \cdot m_{ji})^{x \cdot v_{ji}}, g) \\
&= e(\prod_{i=s_{j1}}^{s_{jc}} (H(RID_j)^{v_{ji}} \cdot m_{ji}{}^{v_{ji}}), g^x) \\
&= e(\prod_{i=s_{j1}}^{s_{jc}} (H(RID_j)^{v_{ji}}) \cdot m_{ji}^{v_{ji}}, mpk) \\
&= e(\mu \cdot \prod_{i=s_{j1}}^{s_{jc}} (H(RID_j))^{v_{ji}}, mpk)
\end{aligned}
$$

5.2. **Audit reliability.** Since the DL problem in the group G_1 is relatively difficult to solve, the malicious CSP can not pass the audit with a negligible probability, except that the CSP has correct data blocks and generates effective proof information. We assume that the data block m_{ji} is to be verified, and the CSP generates proof information $P = \{\mu, \sigma\}$ to the TPA for verification. If P is valid information, it can be verified by the TPA, which means that equation (3) holds. If the CSP uses tampered data blocks m'_{ji} to generate data tags as $\sigma'_{ji} = (H(RID'_j) \cdot m'_{ji})^x$. The proof information $P = \{\mu', \sigma'\}$, where $\mu' = \prod_{i \in I} m'_{ji}{}^{v_{ji}}$, $\sigma' = \prod_{i \in I} \sigma'_{ji}{}^{v_{ji}}$, assuming the CSP sets $\sigma_{ji} = (\sigma'_{ji})^\beta$, where $\beta \in Z_q$. We can calculate $(H(RID_j) \cdot m_{ji})^x = (H(RID'_j) \cdot m'_{ji})^{x \cdot \beta}$, $H(RID_j) = (H(RID'_j))^\beta$ and $m_{ji} = (m'_{ji})^\beta$. Based on the DL problem, it is difficult to calculate the random number β when the CSP knows the $H(RID_j)$ and $H(RID'_j)$. It means that it is difficult for the CSP to find a β satisfy $m_{ji} = (m'_{ji})^\beta$. Therefore, $P = \{\mu', \sigma'\}$ cannot be verified through the TPA, and the CSP cannot perform substitution attacks. Therefore, this theorem holds.

5.3. **Identity privacy protection.** Identity privacy protection means that the CSP and TPA cannot obtain the user's identity information by any means. In the previous scheme, which did not consider protecting user identity information, the CSP and TPA can obtain

user identity information from the audit process. So, in our scheme, we designed an anonymous identity table to make the user's identity anonymous. The GM maintains the IAT to protect the user's real identity. When generating tags, the GM takes advantage of the underivable nature of one-way functions to hide the user's valid identifier, thus better protecting the user's real identity privacy, where $UID = H(RID)$. Specifically, on the one hand, in the storage phase, the CSP can only know the anonymous user identifier UID. Since the hash function H is a one-way safe function, the RID cannot be recovered from $H(RID)$. Also, the TPA can only obtain the UID when verifying the proof information in the audit process, and the TPA cannot infer the real value RID from the proof information generated by the CSP. Therefore, our scheme realizes the privacy protection of identity.

5.4. **Data privacy protection.** In our scheme, data privacy refers to the inability of the TPA to obtain data information. When the CSP generates data proof in the audit process, he re-aggregates and generates the μ based on the data, where $\mu = \prod_{i \in I} m_{ji}{}^{v_{ji}}$, μ rather than linear combination, thus ensuring the privacy of the data. The proof information generated by the CSP cannot disclose the data information of the DO. In our scheme, the CSP sends $P = \{\mu, \sigma, \tau\}$ to the TPA, which μ includes data blocks and σ includes data tags. The attacker is unable to obtain the value of the block tag σ_{ji} from σ, where $\sigma = \prod_{i \in I} \sigma_{ji}{}^{v_{ji}}$. Even if σ_{ji} is obtained, it cannot be recovered m_{ji} from $\sigma_{ji} = (H(RID_j) \cdot m_{ji})^x$, because it does not know the value of x, which is equivalent to solving the DL problem. Therefore, the attacker cannot obtain data information from the proof information. In summary, this scheme meets the nature of privacy protection.

5.5. **Non-interactivity.** In our scheme, the group users do not need to send audit challenge information to the CSP. The group administrator sends the number of challenge data blocks to the CSP, and the CSP can independently obtain public information from the blockchain and generate proof information. There is no data interaction between CSP and group users, which can reduce the communication overhead.

5.6. **Dynamic shared data.** In our scheme, on the one hand, we design a dynamic multi-tree to realize the dynamic nature of shared data, including the insertion, modification, and deletion of data tags. On the other hand, we also propose an upgrade and downgrade strategy to solve the problem of low search efficiency caused by multi-tree compared with the binary tree. In addition, we also cite the corresponding examples in the specific program to describe.

6. **Efficiency evaluation.**

6.1. **Theoretical analysis.** We analyze the main computational overhead generated in the scheme and compare it with the scheme proposed by Yan et al. [19]. The computational cost is mainly from bilinear pairing, exponentiation, and multiplication operations, while addition, subtraction, and hash operations can be ignored. We mainly calculate the computational cost of the following three stages: the TagGen stage mainly includes the operation of generating tags; The TagVerify stage mainly involves verifying the operation of equation (2); The main overhead of the ProofGen phase comes from computation μ and σ; The ProofVerify stage mainly involves verifying the operation of equation (3). T_{mul} represents a multiplication operation; T_{exp} denotes a power operation; T_p denotes a bilinear map operation; n denotes the number of data blocks; c denotes the number of challenge blocks. The main computational expenses for each specific stage are shown in Table 3.

In addition, to prove the feasibility of our scheme, we compared it with existing shared data audit schemes from different perspectives, and the results are shown in Table 4. Where "✓" means the scheme achieves the attribute, and "×" does not. Our scheme meets all the above attributes and has better security and feasibility.

TABLE 3. Computational cost in each phase

	Yan et al.'s scheme [19]	Our Scheme
TagGen Phase	$nT_{mul} + 2nT_{exp}$	$nT_{mul} + nT_{exp}$
ProofGen Phase	$cT_{mul} + cT_{exp}$	$2cT_{exp}$
ProofVerify Phase	$2T_p + T_{mul} + (c+1)T_{exp}$	$2T_p + T_{mul} + cT_{exp}$

TABLE 4. Comparison of scheme properties

	Yang et al.'s scheme [18]	Yan et al.'s scheme [19]	Yan and Gui's scheme [20]	Rao et al.'s scheme [21]	Our scheme
Public auditing	×	×	✓	✓	✓
Identity privacy protection	✓	✓	✓	×	✓
Data privacy protection	✓	✓	✓	✓	✓
Data dynamics	×	×	×	✓	✓
Group user dynamics	✓	✓	×	✓	✓

6.2. **Experimental analysis.** Due to our scheme's support for dynamic data manipulation, we compared the memory space of the newly proposed multi-tree model with the previous complete binary tree storage model under different data numbers. As shown in Figure 10, the horizontal axis denotes different numbers of data tag aggregation, and the vertical axis denotes the corresponding memory space for different numbers of data tag aggregation. In a 32-bit processor, the pointers of memory space occupy 4 bytes. Non-leaf nodes store the denominator of leaf nodes under that node, which are 4 bytes. Leaf nodes store data tag aggregation, which is 16 bytes.
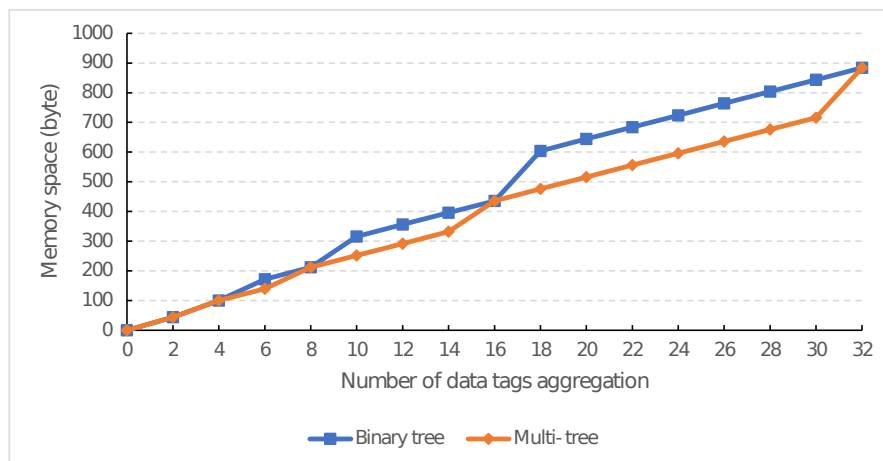

FIGURE 10. Comparison of memory space between binary trees and multi-trees with different numbers of data tags aggregation

To calculate the cost of each stage time in the scheme. Our experiment is running under the Windows 11 operating system. We implemented our experiment using Java

language, placing the experimental code on the 16GB RAM, AMD Ryzen 7 6800H with Radeon Graphics experimental platform, and running it on IntelliJ IDEA 2022.1.3 using jdk-8u221-windows-64bit.

Firstly, we designed an experiment to evaluate the performance of generating tags in our scheme. In this experiment, we generate tags for 0 to 10000 data blocks. The computational overhead of generating tags is proportional to the number of data blocks. In addition to calculating the computational overhead generated tags in our scheme, we also designed a comparative experiment of the tag algorithm, and the experimental results are shown in Figure 11. The experimental results indicate that our scheme has less computation overhead in tag generation than Yan et al. [19].

FIGURE 11. Tag generation time cost

The second experiment evaluates the efficiency of generating and verifying proof information. The experiment compared the time cost of generating proof information between Yan et al.'s scheme and our proposed scheme for different challenge block numbers. As shown in Figure 12, the horizontal axis represents the number of challenge blocks, ranging from 0 to 1000, and the vertical axis represents the time cost of generating proof information. Although our scheme has a larger computational overhead in generating proof information than the scheme proposed by Yan et al., our scheme is more secure. In addition, we also designed an experiment to compare the computational cost of verifying proof information between our scheme and the scheme proposed by Yan et al., as shown in Figure 13. The computational overhead of these two algorithms is proportional to the number of challenge blocks. Our time cost during the validation phase is the same as Yan et al..

Finally, on the whole, the computational overhead of our scheme is lower than Yan et al.'s. Specifically, we have reduced group users' workload generating tags and shifted more work from the DO to the CSP to improve computational efficiency. The CSP has enormous computing power, so our scheme is feasible.

7. **Conclusion.** This paper proposes a shared data integrity audit scheme supporting group user and data dynamics. We construct a dynamic multi-tree storage mode better to achieve the group users and data dynamics. The identity anonymous table is designed to protect the user's identity information. We encrypt the certification information during the audit process to prevent auditors from obtaining user information. Our scheme can meet the security requirements for good cloud-shared data. The experimental results indicate that this scheme has high security and feasibility. Although our scheme solves
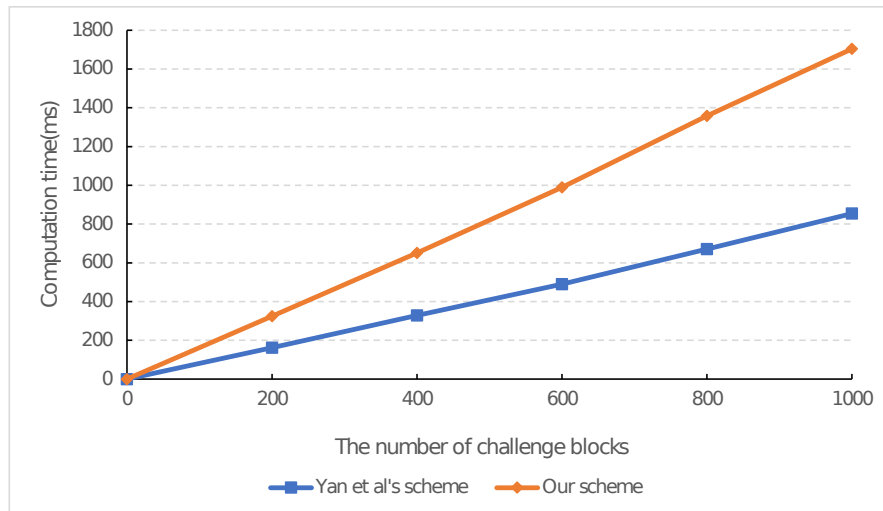
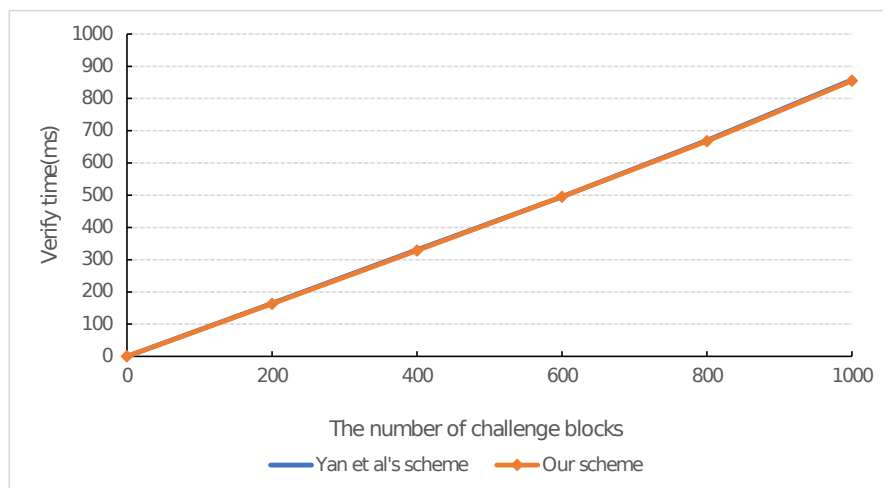FIGURE 12. Proof information generation time cost

FIGURE 13. Verify time cost

the privacy security of data and the privacy protection of identity, we need to consider that there may be some malicious behaviors of group users and administrators. In the future, we will focus on how to ensure the credibility of group users.

REFERENCES

[1] M. Ali, R. Dhamotharan, E. Khan, S. U. Khan, A. V. Vasilakos, K. Li, and A. Y. Zomaya, "Sedasc: secure data sharing in clouds," *IEEE Systems Journal*, vol. 11, no. 2, pp. 395–404, 2015.

[2] P. Yang, N. Xiong, and J. Ren, "Data security and privacy protection for cloud storage: A survey," *IEEE Access*, vol. 8, pp. 131 723–131 740, 2020.

[3] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo, "Blockchain-based fair payment smart contract for public cloud storage auditing," *Information Sciences*, vol. 519, pp. 348–362, 2020.

[4] P. Sharma, R. Jindal, and M. D. Borah, "Blockchain technology for cloud storage: A systematic literature review," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–32, 2020.

[5] D. Thilakanathan, S. Chen, S. Nepal, and R. A. Calvo, "Secure data sharing in the cloud," in *Security, Privacy and Trust in Cloud Systems.*   Springer, 2013, pp. 45–72.

[6] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *2010 International Conference on Intelligent Computing and Cognitive Informatics.*   IEEE, 2010, pp. 380–383.

[7] P. M. Reddy, S. Manjula, and K. Venugopal, "Secure data sharing in cloud computing: a comprehensive review," *International Journal of Computer (IJC)*, vol. 25, no. 1, pp. 80–115, 2017.

[8] C. Wang, S. S. 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, 2011.

[9] Y. Ji, B. Shao, J. Chang, and G. Bian, "Flexible identity-based remote data integrity checking for cloud storage with privacy preserving property," *Cluster Computing*, pp. 1–13, 2022.

[10] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 331–346, 2018.

[11] Y. Xu, L. Ding, J. Cui, H. Zhong, and J. Yu, "Pp-csa: A privacy-preserving cloud storage auditing scheme for data sharing," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3730–3739, 2020.

[12] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 43–56, 2014.

[13] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1432–1437, 2011.

[14] J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong, "Privacy-preserving public auditing protocol for low-performance end devices in cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2572–2583, 2016.

[15] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "Npp: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Transactions on Big Data*, vol. 8, no. 1, pp. 14–24, 2017.

[16] J. Xue, C. Xu, J. Zhao, and J. Ma, "Identity-based public auditing for cloud storage systems against malicious auditors via blockchain," *Science China Information Sciences*, vol. 62, pp. 1–16, 2019.

[17] J. R. Gudeme, S. Pasupuleti, and R. Kandukuri, "Certificateless privacy preserving public auditing for dynamic shared data with group user revocation in cloud storage," *Journal of Parallel and Distributed Computing*, vol. 156, pp. 163–175, 2021.

[18] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *Journal of Systems and Software*, vol. 113, pp. 130–139, 2016.

[19] Y. X. Yan, L. Wu, W. Y. Xu, H. Wang, Z. M. Liu *et al.*, "Integrity audit of shared cloud data with identity tracking," *Security and Communication Networks*, vol. 2019, 2019.

[20] H. Yan and W. Gui, "Efficient identity-based public integrity auditing of shared data in cloud storage with user privacy preserving," *IEEE Access*, vol. 9, pp. 45 822–45 831, 2021.

[21] L. Rao, H. Zhang, and T. Tu, "Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated merkle hash tree," *IEEE Transactions on Services Computing*, vol. 13, no. 3, pp. 451–463, 2017.

[22] Y. Yu, Y. Mu, J. Ni, J. Deng, and K. Huang, "Identity privacy-preserving public auditing with dynamic group for secure mobile cloud storage," in *Network and System Security: 8th International Conference, NSS 2014, Xi'an, China, October 15-17, 2014, Proceedings 8.*   Springer, 2014, pp. 28–40.

[23] S. Jones, A. Ball *et al.*, "The data audit framework: A first step in the data management challenge," *International Journal of Digital Curation*, vol. 3, no. 2, pp. 112–120, 2008.

[24] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 598–609.

[25] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1165–1176, 2016.

[26] J. Li, H. Yan, and Y. Zhang, "Identity-based privacy preserving remote data integrity checking for cloud storage," *IEEE Systems Journal*, vol. 15, no. 1, pp. 577–585, 2020.

[27] G. Bian, R. Zhang, and B. Shao, "Identity-based privacy preserving remote data integrity checking with a designated verifier," *IEEE Access*, vol. 10, pp. 40 556–40 570, 2022.

[28] A. Barsoum and A. Hasan, "Enabling dynamic data and indirect mutual trust for cloud computing storage systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 12, pp. 2375–2385, 2012.

[29] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Transactions on Information and System Security (TISSEC)*, vol. 17, no. 4, pp. 1–29, 2015.

[30] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C.-J. Hu, "Dynamic audit services for outsourced storages in clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 227–238, 2011.

[31] H. Xiong, Q. Mei, Y. Zhao, L. Peng, and H. Zhang, "Scalable and forward secure network attestation with privacy-preserving in cloud-assisted internet of things," *IEEE Sensors Journal*, vol. 19, no. 18, pp. 8317–8331, 2019.

[32] T.-Y. Wu, F. Kong, Q. Meng, S. Kumari, and C.-M. Chen, "Rotating behind security: an enhanced authentication protocol for iot-enabled devices in distributed cloud computing architecture," *EURASIP Journal on Wireless Communications and Networking*, vol. 2023, no. 1, p. 36, 2023.

[33] G. Thakur, P. Kumar, C.-M. Chen, A. V. Vasilakos, S. Prajapat *et al.*, "A robust privacy-preserving ecc-based three-factor authentication scheme for metaverse environment," *Computer Communications*, vol. 211, pp. 271–285, 2023.

[34] B. Wang, B. Li, and H. Li, "Knox: privacy-preserving auditing for shared data with large groups in the cloud," in *Applied Cryptography and Network Security: 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings 10.* Springer, 2012, pp. 507–525.

[35] Y. Li, Y. Li, K. Zhang, and Y. Ding, "Public integrity auditing for dynamic group cooperation files with efficient user revocation," *Computer Standards & Interfaces*, vol. 83, p. 103641, 2023.

[36] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 767–778, 2016.

[37] Y. Zhang, J. Yu, R. Hao, 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, 2018.

[38] X. Huang, H. Xiong, J. Chen, and M. Yang, "Efficient revocable storage attribute-based encryption with arithmetic span programs in cloud-assisted internet of things," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1273–1285, 2023.

[39] M. Szydlo, "Merkle tree traversal in log space and time," in *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23.* Springer, 2004, pp. 541–554.