Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Remote Data Possession Checking Scheme with Supporting Efficient Group User Authority Management for Shared Cloud Data

Yilin Yuan (yilinyuan922@126.com)
Beijing University of Technology School of Computer: Beijing University of Technology Faculty of Information Technology https://orcid.org/0000-0002-9021-5616

Jianbiao Zhang
Beijing University of Technology School of Computer: Beijing University of Technology Faculty of Information Technology

Wanshan Xu
Beijing University of Technology School of Computer: Beijing University of Technology Faculty of Information Technology

Xiao Wang
Tianjin University of Finance and Economics

Yanhui Liu
Beijing University of Technology School of Computer: Beijing University of Technology Faculty of Information Technology

Research Article

Keywords: Shared cloud data, data possession checking, identity-based encryption, authorization invisible authenticator

Posted Date: September 16th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-884701/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Remote Data Possession Checking Scheme with Supporting Efficient Group User Authority Management for Shared Cloud Data

Yilin Yuan[1,2], Jianbiao Zhang[1,2,*], Wanshan Xu [1,2] Xiao Wang[3], and Yanhui Liu[1,2]

Abstract

Under the shared big data environment, most of the existing data auditing schemes rarely consider the authorization management of group users. Meanwhile, how to deal with the shared data integrity is a problem that needs to be pondered. Thus, in this paper, we propose a novel remote data checking possession scheme which achieves group authority management while completing the public auditing. To perform authority management work, we introduce a trusted entity – group manager. We formalize a new algebraic structure operator named authorization invisible authenticator (AIA). Meanwhile, we provide two versions of AIA scheme: basic AIA scheme and standard AIA scheme. The standard AIA scheme is constructed based on the basic AIA scheme and user information table (UIT), with advanced security and wider applicable scenarios. By virtue of standard AIA scheme, the group manager can perfectly and easily carry out authority management, including enrolling, revoking, updating. On the basis of the above, we further design a public auditing scheme for non-revoked users' shared data. The scheme is based on identity-based encryption (IBE), which greatly reduce the necessary certificate management cost. Furthermore, the detailed security analysis and performance evaluation demonstrate that the scheme is safe and feasible.

Keywords: Shared cloud data, data possession checking, identity-based encryption, authorization invisible authenticator

Introduction

As cloud computing becomes more accepted by the public, cloud storage is also widely used. Cloud Server Provider (CSP) supply users with large-capacity and reasonably priced storage services. Users can choose different types of cloud storage services to store outsourced data according to their needs. Cloud storage has many advantages, such as pay on- demand, location independence, and off-site management, which makes it more popular. From an objective point of view, because users no longer have absolute control over outsourced data, the security and integrity of data on the cloud is facing huge risks. Once an error occurs, it will inevitably cause serious losses for users.

With the assistance of data sharing services, users can share data in the cloud with other users in the group. The data sharing service in the group not only saves the burden of local data storage for group users, but also reduces the overhead of multiple interactions with the CSP. Within a group, only users with legal permissions can access the shared data and can access freely. But in fact, the authorization management is a hard task for groups. There may exist some difficulties: (1) For a new user, how to apply to enroll the group to become a legal group user? (2) How to withdraw the shared data access rights of the users who have applied for leaving the group; if the user has shared the shared data in the group, how to revoke the ownership of this data? (3) How do group users view shared data? In addition, how to deal with the shared data integrity of group users in a shared big data environment is a problem that needs to be considered. To solve the above-mentioned challenge, this paper conducts a detailed discussion on the authority management of group users under the shared data integrity verification problem.

Contribution

In order to share cloud data, we construct a remote data possession checking scheme with supporting efficient group user authority management in this paper.

(1) In order to deal with the issue of opening and recovering the permissions of users within a group, in our proposed scheme, we introduce a trusted entity that is engaged in rights management - group manager.

(2) In order to assist the group manager in authority

management, we further propose a novel algebraic structure operator called authorization invisible authenticator (AIA). AIA is an operator that can only be generated by the group manager, and can be used by group users, but is not visible. Furthermore, we provide two versions of AIA scheme: the basic AIA scheme and the standard AIA scheme. The standard AIA scheme is constructed based on the basic AIA scheme and user information table (UIT), with advanced security. In addition, we prove the security of the standard AIA scheme in terms of correctness, invisibility, unforgeability and non-malleability. By manipulating AIA, group manager can carry out authority management work perfectly and easily. There are advantages of AIA: a) Since AIA can only be distributed by the group manager, group users are available but cannot infer its algebraic structure and cannot forge legal AIA. Therefore, the security of authority management can be guaranteed. b) Integrity verification schemes for shared data with supporting user revocation are currently a research hotspot. With the aid of AIA, this problem can be well resolved. In addition, the cost of user revocation in a group no longer depends on the number of data blocks, but only on the number of group users. And it makes our scheme more practical. c) AIA is updated uniformly by the group manager only when a user leaves the group, so it can greatly save the calculation and transmission costs.

(3) To complete remote data possession checking for non-revoked group users, we further construct a shared data public auditing scheme on the basis of the above work. The proposed scheme provides the mathematical design of the data block tag, and the concrete process of public auditing. It is worth mentioning that the tags set is based on the identity-based encryption (IBE), which greatly saves the time it takes for the system to process the public key certificate. The correctness of our scheme, including private key correctness, AIA correctness and audit correctness, is well given. Moreover, under the given random oracle model, the soundness based on the Computational Diffie- Hellman Assumption and the Discrete Log Assumption is strictly proved. Besides, the performance evaluation demonstrates that the scheme is effective and feasible.

Related Work

The traditional remote data possession checking method requires users to download all their data stored in the cloud locally, and then perform checking. For users with low-capacity equipment or low computing capacity, the cost is fatal. In 2007, with the successive schemes of Provable Data Possession (PDP) [1] and Proof of Retrievability (PoR) [2], the research on data integrity verification was pushed to an un-precedented climax. The PDP scheme is based on homomorphic authenticators and random sampling strategy and uses RSA to design the tags set. The PoR scheme is implemented in bilinear pairing, which not only provides integrity verification, but also enables data retrieval. In addition, based on PoR, Shacham et al. [3] proposed two schemes which can complete the public verifiability and private verifiability using BLS signature and pseudorandom function respectively.

However, neither the PDP scheme nor the PoR scheme can realize the dynamic operations of the da-ta block. The schemes that cannot realize dynamic operations, such as block modification, deletion and append, which will be restricted in some scenarios. Aimed at this problem, Ateniese et al. [4] proposed a scalable PDP scheme based on symmetric key cryptography. By manipulating rank authenticated information table, Erway et al. [5] also provided an im-proved dynamic PDP scheme. Based on Merkle Hash Tree, Wang et al. [6] raised a public data integrity auditing scheme which can also realize dynamics by constructing block tag authentication. For simultaneous update of multiple replicas of files, Barsoum et al. [7] given a specific proposal. When the client's secret key is leaked, Yu et al. [8] put forward a strategy that can update the keys while achieving public auditing. Sookhak et al. [9] investigated the realization of data dynamics with the aid of a novel structure -divide and conquer tables. With the help of the authenticated structure named blockless Merkle tree, He et al [10] proposed a dynamic group-oriented PDP scheme which can realize full dynamics. Considering that the traditional scheme is only applicable to audits where there is no dispute over data possession, combined with blockchain technology, Xu et al. [11] provided a fairly arbitrable data auditing protocol. Wang et al. [12] formalized an

online/offline PDP model, which divides data processing into two procedures to deal with different types of operations respectively.

But unfortunately, all the schemes mentioned above are implemented by Public Key Infrastructure (PKI). Public key cryptography based on PKI distribution keys has been widely used since its introduction. However, due to the existence of certificates, a series of costs such as generation, forwarding, and renewal are inevitably increased. Using the user's public identifier, such as the email address, as an encryption key can solve a series of problems caused by public key certificates [13-14]. Based on Identity- Based Encryption (IBE), the first public data integrity auditing scheme was proposed [15]. Wang [16] formalized an identity-based distributed provable data possession model which can realize private verification, delegated verification, and public verification. Wang et al. [17] discussed how to perform public integrity checking when users are restricted to access the CSP. Assisting by the entity of sanitizer, Shen et al. [18] given a data sharing scheme that supports the hiding of sensitive information. Using the user's biometric data as the encryption key and combining the attribute set encryption, Li et al. [19] studied a novel scheme based on fuzzy identity- based encryption [20]. To resist malicious auditors, Zhang et al. [21] proposed an identity- based public auditing scheme, which is implemented with the help of blockchain. Shen et al. [22] provided a remote data possession auditing scheme with supporting privacy of authenticators. To solve the problem that privacy may be leaked after the data is deleted in the cloud, Xue et al. [23] built a specific security model based on attribute set encryption and Merkle tree.

The group data sharing services under the integrity verification problem has been deeply studied in recent years. To address the issue of integrity verification of shared cloud data, Wang et al. proposed three different schemes [24-26]. Konx [24] focused on the privacy protection within the group and utilized group signatures to construct homomorphic authenticators to complete data checking. Different from Konx, Oruta [25] considered using ring signatures to achieve mathematical design. Panda [26] explored how to employ proxy re-signatures to realize group user revocation. Yuan et al. [27] discussed a public integrity auditing scheme that can support multiple users to modify shared data. Tian et al. [28] given a comprehensive group management integrity auditing scheme which considers privacy protection, dynamics, and traceability of shared data. In order to support user revocation effectively, Zhang et al. [29] explored a new strategy for key generation phase and a technique for private key update. By constructing a homomorphic verifiable group signature in the privacy-aware public auditing mechanism. Fu et al. [30] proposed a group shared data checking scheme which can prevent sensitive information from leaking to third-party auditors. Starting from the collusion-resistant attack, Luo et al. [31] provided the scheme to realize public auditing and user revocation.

In a group that shares cloud data, user revocation is a hard assignment. In the schemes mentioned above [26-31], the overhead of realizing user revocation in a group is directly associated with the number of users withdrawn, and even shows a linear relationship. It must inevitably lead to huge costs when the number of users is large. On the other hand, there are many problems to be solved in the group, such as: users' enrollment, data privacy.

Paper Organization

The rest of the paper is organized as follows. In section II, we present the preliminaries. And in section III, we present the system model and security model. A new algebraic structure operator named authorization invisible authenticator is explained in section IV. The detailed description of the proposed scheme is elaborated in section V. Section VI shows the security analysis. The performance evaluation is carried out in section VII. Section VIII is the conclusion of this paper.

Preliminaries

Bilinear Maps

Let G_1, G_2 are multiplicative cyclic group with the order p, g is a generator of G_1. A bilinear map $e: G_1 \times G_1 \to G_2$ satisfies the following properties:

a) Bilinearity: $\forall u, v \in G_1$ and $\forall a, b \in Z_p^*$, $e(u^a, v^b) = e(u, v)^{ab}$;

b) Non-degeneracy: $e(g_1,g_2)\neq 1$;

c) Computable: there is an efficient algorithm to calculate e.

Security Assumptions

Computational Diffie- Hellman Assumption. For unknown $\forall a,b\in Z_p^*$, given g, g^a and g^b as input, output $g^{ab}\in G_1$.

Definition 1 (CDH assumption). The advantage of a *PPT* (probabilistic polynomial time) algorithm $\mathcal{A}$ in solving the CDH problem in G_1 defined below is negligible:

$$AdvCDH_{\mathcal{A}}=\Pr[\mathcal{A}(g,g^a,g^b)=g^{ab}:a,b\overset{R}{\leftarrow}Z_p^*]$$

Discrete Logarithm Assumption. For unknown $x\in Z_p^*$, given g and g^x as input, output x.

Definition 2 (DL assumption). The advantage of a *PPT* (probabilistic polynomial time) algorithm $\mathcal{A}$ in solving the DL problem in G_1 defined below is negligible:

$$AdvDL_{\mathcal{A}}=\Pr[\mathcal{A}(g,g^x)=x:x\overset{R}{\leftarrow}Z_p^*]$$

System Model and Security Model

System Model

The system consists of five entities, and the model is shown in Fig. 1.

(1) Group Manager: the trusted entity who performs authority management work. The group manager opens the access authorizations for the users who apply to enroll the group, and at the same time revokes the access rights of shared data of users who leave the group. In addition, the data upload work of the group users is also executed by the group manager. Moreover, the public audit results of shared data within the group are recorded by the group manager, not the creator of the shared data.

(2) Group Users: When the user successfully enrolls the group, he becomes a legal group user; he can upload his data to the CSP and mark it as shared data; at the same time, he can access the shared data of other users in the group. That is, in our proposed scheme, all legal and unrevoked group users can upload data to the CSP and share it with other users. When a group user revokes and leaves the group, the group manager will deprive the user of the permission to view other shared files, but the shared files uploaded by this user still exist in the group. That is, once a group user upload files and share them with other group users, those files will be regarded as a shared data. The revoked user cannot access other users' shared files, but it does not affect other group users to view the shared files he has uploaded. In addition, the ownership of the shared file of the revoked group user will be transferred to the group manager; before the files are not modified, the revoked group user still knows the content of the shared file.

(3) Cloud Service Provider (CSP): an untrusted entity that provides public cloud storage service.

(4) Private Key Generator (PKG): the trusted entity who generates the private key for the group users.

(5) Third- Party Auditor (TPA): the trusted entity who is responsible for performing remote data checking for the shared data uploaded to the CSP after being authorized by the group users. When the TPA wishes to check the integrity of the shared data, it first sends an auditing challenge to the CSP. After receiving this challenge, the CSP responds to the TPA with an auditing proof of the possession of this shared data. Subsequently, the TPA will verify the correctness of this auditing proof, and then record it. The audit results will be feedback to the group manager on a regular basis.

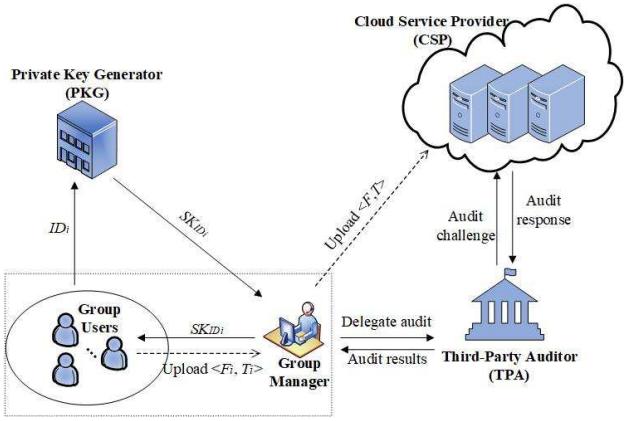

Fig. 1 The system model diagram

Remark 1 The private key generation process is briefly shown in Fig. 1, which will be described in detail in section V.

Remark 2 The responsibilities of the group manager include the four parts: a) Grant permissions to the users who apply to join the group. b) Deprive permissions to the users who leave the group. Note that even if the user opts out of the group, he still has the right to view the shared files he uploaded, but he does not have the right to modify it. c) Perform uploading of shared data on behalf of group users. d) Record the audit results of the integrity verification of the shared data. After the TPA checks the integrity of the shared files, it records the results of each audit and feeds it back to the group manager on a regular basis, rather than directly to the creator of the shared files.

System Components

A remote data possession checking scheme with supporting efficient group user authority management consists of eight algorithms: **Setup**, **AuthForUser**, **KeyGen**, **TagGen**, **UpdateForUsers**, **Challenge**, **ProofGen**, **ProofVerify**. Each algorithm is described as follows:

Setup is the "Setup" algorithm run by the PKG. It takes the security parameter k as input, and outputs the system public parameter pp, master public key mpk and master secret key msk.

AuthForUser is the "Authorized for Group User" algorithm which consists of a series of algorithms. It takes the pp as input, and outputs an AIA for the group user.

KeyGen is the "Private Key Generation" algorithm run by the PKG. It takes the pp, user's identity ID and mpk as input, and outputs user's private key SK_{ID}.

TagGen is the "Tags Set Generation" algorithm run by the group user. It takes SK_{ID}, the user's requirement message rm and shared file F as input, and outputs the tags set T for F. Then, the user uploads $<F, T>$ to the group manager. Subsequently, the group manager performs the data upload work.

UpdateForUsers is the "Update AIA for The Rest of Group Users" algorithm run by the group manager. When one user is revoked from the group, the group manager runs this algorithm.

Challenge algorithm run by the TPA that is to generate the auditing challenge $chal$.

ProofGen is the "Proof Generation" algorithm run by the CSP. It takes the $chal$, F and T as input, and outputs the integrity proof P.

ProofVerify is the "Proof Verification" algorithm run by the TPA. It takes pp, $chal$ and P as input, and outputs "0/1"; "1" indicates that the shared data stored in the CSP is intact; otherwise, it is not.

Design Goals

A remote data possession checking scheme with supporting efficient group user authority management should achieve following goals:

(1) **Correctness**: The correctness consists of three parts: private key correctness, AIA correctness and audit correctness.

a) **Private Key Correctness**: The private key generated by the PKG will only be accepted by the group user after passing the correctness verification of the group manager.

b) **AIA Correctness**: A legal AIA Φ can be passed the verification of the group manager.

c) **Audit Correctness**: If the CSP completely stores the user's cloud data, the proof generated by the CSP can be verified by the TPA.

(2) **Audit Soundness**: If the CSP does not possess user's intact data, it cannot pass the TPA's verification.

(3) **Secure authority management**: The group manager can safely and efficiently perform authority management work in the group. In addition, any user joining/leaving the group will not affect other users.

(4) **Efficient revocation**: When one user applies for revocation, only the group manager needs to perform the operation with lower computational overhead. In other words, the cost of user revocation is independent of the number of the revoked users.

(5) **Public auditing**: Under the authorization of the group users, the TPA performs remote data integrity verification for the shared data stored on the CSP; and periodically returns the audit results to the group manager.

Security Model

To formalize security model, we put forward a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is similar to reference [3]. The user is regarded as the challenger $\mathcal{C}$, and the untrusted CSP is regarded as the

adversary $\mathcal{A}$. This game shows how the adversary $\mathcal{A}$ is against the security of the proposed scheme, and specific phases are given as follows:

(1) **Setup phase**

The challenger $\mathcal{C}$ runs the Setup algorithm to obtain the system public parameter *pp* and master secret key (*msk*), then sends *pp* to the adversary $\mathcal{A}$.

(2) **Query phase**

In this phase, the adversary $\mathcal{A}$ makes two types of queries to the challenger $\mathcal{C}$.

a) **KeyGen** phase. The adversary $\mathcal{A}$ queries the user's private key. The challenger $\mathcal{C}$ runs the **KeyGen** algorithm to get the private key SK_{ID}, then sends it to the adversary $\mathcal{A}$.

b) **TagGen** phase. The adversary $\mathcal{A}$ queries the tags set of the encrypted file *F*. The challenger $\mathcal{C}$ runs the **TagGen** algorithm to obtain the tags set of the *F*, then sends it to the adversary $\mathcal{A}$.

(3) **Challenge phase**

In this phase, the adversary $\mathcal{A}$ acts as the prover and the challenger $\mathcal{C}$ as the verifier. The challenger $\mathcal{C}$ sends the integrity challenge to the adversary $\mathcal{A}$ and asks the adversary $\mathcal{A}$ to provide the integrity proof *P* against the challenge set *chal*.

(4) **Forgery phase**

After receiving the integrity challenge from the challenger $\mathcal{C}$, the adversary $\mathcal{A}$ outputs the integrity proof *P* and replies to the challenger $\mathcal{C}$. If the integrity proof *P* can pass the verification of the challenger $\mathcal{C}$ with non- negligible probability, we say the adversary $\mathcal{A}$ wins this game.

In the above security model, we need to prove that if an adversary $\mathcal{A}$ does not truly store all the blocks related to the challenge set, it cannot generate the valid integrity proof and can pass the verification of challenger $\mathcal{C}$.

Definition 3 If the integrity proof *P* forged by adversary $\mathcal{A}$ can pass the verification of challenger $\mathcal{C}$ with non- negligible probability, there exists a knowledge extractor that can extract the challenged data blocks except negligible probability.

Definition 4 If the CSP corrupts ρ fraction of the whole file, these corrupted data blocks can be detected with the probability of at least δ using the proposed scheme, where $(\rho,\delta)(0<\rho,\delta<1)$.

Authorization Invisible Authenticator

When a new user applies to enroll the group, the group manager performs authority management and authorized him to become a legal group user. Correspondingly, when a group user chooses to withdraw the group, the group manager should revoke all their permissions, including the right to modify the shared data they have uploaded, and the access right to view other shared data.

Thus, in this section, we construct a novel algebraic structure operator named authorization invisible authenticator to facilitate the group manager to deal with permission management issues. For simplify, we call it AIA. By manipulating AIA, the group manager can carry out work perfectly and easily. An AIA should meet the following requirements:

- **Invisibility**: AIA is distributed to group users by group administrators; group users possess legal AIAs are regarded as legal users in the group. AIA can only be used by the group users, but the algebraic structure is transparent to the group users; so, it cannot be inferred.
- **Unforgeability**: The group users cannot forge a legitimate AIA. The unforgeability of AIA determines that any unauthorized user cannot view the shared data in the group.
- **Non-malleability**: Given two AIA $\Phi_1,\Phi_2 \in Z_P^*$, and two values $a,b \in Z_P^*$. Any legal AIA cannot be computed by linearly aggregating $\Phi = a\Phi_1 + b\Phi_2 \in Z_P^*$, except with negligible probability. The non-malleability of AIA ensures that no two unauthorized users can collude to forge a legitimate AIA.

Overview of Basic AIA Scheme

Before giving the standard AIA scheme, we first introduce a basic AIA scheme. A basic AIA scheme consists of four algorithms:

(1) Init ($1^k \to pp$): This is "Init" algorithm to initialize some public parameters. It takes the security parameter *k* as input, and outputs the public parameter *pp* (public parameter *pp* mentioned here are same as those initialized in the "**Setup**" phase in Section V).

(2) ApplyForAuth ($pp \rightarrow rm$): This is "Apply for Authorization" algorithm run by the group user. When a new user applies to join the group, he needs to ask for the group manager to authorize him to become a legal group user. It takes the *pp* as input, and outputs a requirement message *rm*. In order to facilitate the work of group manager and consider the security of the basic AIA scheme, the basic AIA scheme requires users to send *rm* only once when applying to enroll the group.

(3) AuthGen ($rm, pp \rightarrow \Phi$): This is "AIA Generation" algorithm run by the group manager. It takes the *rm* and *pp* as input, and outputs an AIA Φ for the legal group user.

(4) AuthUpdate: This is "AIA Update" algorithm run by the group manager. When the user leaves the group, the group manager needs to redistribute the AIA for the remaining group users.

User Information Table (UIT)

The general AIA scheme is suitable for group user authorization management in most scenarios. However, under our proposed scheme, the basic AIA scheme may exist two security threats: 1) Despite in the **ApplyForAuth** algorithm, the basic AIA scheme requires that the user sends *rm* to the group manager only once; But there may be a malicious group user. After sending *rm* to the group manager and obtaining AIA, he continues to send *rm* to the group manager many times and expects to obtain more AIA. The ultimate purpose of the malicious group user is to obtain multiple AIAs, tries to derive the mathematical structure of the AIA, and then forge a reliable AIA. 2) There may be a legitimate but curious group user who tries to use another valid *rm* to apply for AIA from the group manager. Considering the above potential security issue, it is necessary for the group manager to preserve the UIT locally. Moreover, with the help of UIT, the security of the basic AIA scheme can be further improved.

The group manager maintains a local user-information-table (UIT) to record the information of every group user. The UIT contains five columns, and its structure is illustrated in Table 1. *UN* represents the number of current users in the group. *ID* is the identifier of the group user. *rm* is the requirement message sent to the group manager when the user joins the group. AIA is a legal access authenticator distributed to users by group manager. SK_{ID} is the private key of the group user obtained from the PKG (The usage of UIT will be shown in Section V).

Table 1 User Information Table (UIT)

UN	***ID***	***rm***	**AIA**	SK_{ID}
1	ID_1	$(z_1^{r_1}, ID_1)$	AIA_1	SK_{ID_1}
2	ID_2	$(z_2^{r_2}, ID_2)$	AIA_2	SK_{ID_2}
…	…	…	…	…

There are advantages of UIT: (1) UIT records the matching relationship between group user's *rm* and AIA. In the **ApplyForAuth** phase, the group manager will only respond to the group user's requirement message *rm* once. And when one user is withdrawn from the group, the group manager will immediately delete the information of this user from the UIT and update the AIA of the remaining group users. In other words, the cost of user revocation only needs to be borne by the group manager. (2) Only when a user's *ID*, *rm*, AIA and SK_{ID} form a one-to-one correspondence, that is, they are all in the same row of UIT, this user is regarded as a legal group user. Illegal group users will be removed from the group by the group manager.

Standard AIA Scheme

We demonstrate a standard AIA scheme which is composed of the basic AIA scheme and UIT, as shown in Fig. 2.

Remark 3 In order to save calculation and transmission costs, in our proposed standard AIA scheme, the group manager reassigns AIA for the rest of the group users only after one user leaves this group.

Remark 4 Note that in the standard AIA scheme we proposed, the cost of user revocation in a group is irrelevant to the shared data, but only with the number of group users.

Security of AIA Scheme

A standard AIA scheme should satisfy the above properties. We prove it to be secure in terms of

A standard AIA scheme is defined as:

(1) **Init**.

Let G_1, G_2 are two multiplicative groups with the same prime order p, and g is the generator of G_1. There exists a bilinear pairing $e: G_1 \times G_1 \to G_2$ and a cryptographic hash function $H: \{0,1\}^* \to Z_p^*$. Let the public parameter is $pp = (G_1, G_2, p, g, e, H)$.

(2) **ApplyForAuth**.

a) The new user selects two random elements $z, r \in Z_p^*$, and computes z^r.

b) The user sends $rm = (z^r, ID)$ to the group manager.

(3) **AuthGen**.

a) After receiving the requirement, the group manager chooses elements $\alpha, \varepsilon \in Z_p^*$, calculates $\Phi = z^r \cdot H(ID) + \alpha\varepsilon \bmod p$ and returns it to user.

b) The group manager sets $UN = UN + 1$.

c) The group manager updates the UIT.

(4) **AuthUpdate**.

a) When a user leaves the group, the group manager reselects random elements $\alpha', \varepsilon' \in Z_p^*$, recomputes AIA $\Phi' = z^r \cdot H(ID) + \alpha'\varepsilon' \bmod p$ and redistributes it for current user in group.

b) The group manager updates $UN = UN - 1$.

c) The group manager updates the *UIT*.

Fig. 2 The process of standard AIA scheme

correctness, invisibility, unforgeability and non-malleability.

Theorem 1 (Correctness) Given a legal AIA Φ, the group manager can verify its correctness.

Proof: The group manager can parse the AIA Φ through the following derivation and further to verify its correctness.

$$g^{\Phi} = g^{z^r \cdot H(ID) + \alpha\varepsilon} = g^{z^r \cdot H(ID)} \cdot g^{\alpha\varepsilon} = g^{z^r} \cdot g^{H(ID)} \cdot g^{\alpha\varepsilon}$$

Theorem 2 (Invisibility) The AIA is invisible for the group user.

Proof: Since the algebraic structure of AIA is confidential and transparent, AIA is not visible to any group users.

Theorem 3 (Unforgeability) For any adversary $\mathcal{A}$, the probability that he successfully forged AIA is computationally infeasible.

Proof: Assume Φ_1 is a legal AIA, and adversary $\mathcal{A}$ has successfully forged an AIA denoted as Φ_2. We give a brief proof as follows:

The valid AIA is $\Phi_1 = z^r \cdot H(ID) + \alpha\varepsilon \bmod p$, and we have:

$$g^{\Phi_1} = g^{z^r \cdot H(ID) + \alpha\varepsilon} \tag{1}$$

The forged AIA is $\Phi_2 = z^r \cdot H(ID) + \alpha'\varepsilon' \bmod p$, and we also have:

$$g^{\Phi_2} = g^{z^r \cdot H(ID) + \alpha'\varepsilon'} \tag{2}$$

we divide (1) by (2), and obtain:

$$g^{\Phi_1} / g^{\Phi_2} = g^{z^r \cdot H(ID) + \alpha\varepsilon} / g^{z^r \cdot H(ID) + \alpha'\varepsilon'} = g^{\alpha\varepsilon} / g^{\alpha'\varepsilon'}$$

According to the above assumptions, we can know $1 = g^{\alpha\varepsilon} / g^{\alpha'\varepsilon'} = g^{\alpha\varepsilon - \alpha'\varepsilon'}$, and further we get $\alpha\varepsilon = \alpha'\varepsilon' = 0 \bmod p$. Here $\alpha, \varepsilon \in Z_p^*$ are two large prime selected randomly by the group manager. Obviously, the probability of $\alpha\varepsilon = \alpha'\varepsilon' = 0 \bmod p$ is $p \times p = p^2$ which is negligible. Therefore, it can be inferred that the probability that the adversary $\mathcal{A}$ successfully forges a legal AIA is negligible.

Theorem 4 (Non-malleability) For any two adversaries $\mathcal{A}_1, \mathcal{A}_2$, the AIA cannot be forged by collusion.

Proof: Suppose that two adversaries $\mathcal{A}_1, \mathcal{A}_2$ possess rm_1, rm_2 certified by the group users, and haven got AIA Φ_1, Φ_2 respectively. $\Phi = z^r \cdot H(ID) + \alpha\varepsilon \bmod p$ is a legal AIA. The non-malleability of the AIA scheme can be proved by the following:

Two adversaries $\mathcal{A}_1, \mathcal{A}_2$ respectively choose random values $a, b \in Z_p^*$, and they try to linearly aggregate AIA $\Phi' = a\Phi_1 + b\Phi_2 \in Z_p^*$ to obtain Φ. But in fact, there is no way to infer a valid AIA Φ, the reason is as follows:

The algebraic structure of valid AIA is

$\Phi = z^r \cdot H(ID) + \alpha\varepsilon \mod p$. And the algebraic structure of the aggregated AIA is

$$\begin{aligned}\Phi' &= a\Phi_1 + b\Phi_2 \\ &= a(z_1^{r_1} \cdot H(ID_1) + \alpha\varepsilon \mod p) \\ &\quad + b(z_2^{r_2} \cdot H(ID_2) + \alpha\varepsilon \mod p) \\ &= (az_1^{r_1} \cdot H(ID_1) + bz_2^{r_2} \cdot H(ID_2)) \\ &\quad + (a+b) \cdot \alpha\varepsilon \mod p \\ &\neq z^r \cdot H(ID) + \alpha\varepsilon \mod p = \Phi\end{aligned}$$

Obviously, the above equation does not hold. So, the valid AIA Φ cannot be forged by two adversaries $\mathcal{A}_1$, $\mathcal{A}_2$ through linear aggregation. Therefore, it can be seen that for any two adversaries $\mathcal{A}_1$, $\mathcal{A}_2$, the AIA cannot be forged by collusion.

The Proposed Scheme

In our proposed scheme, all the users who have not been revoked can upload data and share them with other group users. The group user authorized the TPA to perform remote data integrity verification of shared data. Further, in order to facilitate the authority management and ensure the correct execution of the public auditing of the revoked user's shared data, the audit results of the shared data are no longer returned to the group user but are recorded by the TPA and fed back to the group manager periodically.

A remote data possession checking scheme with supporting efficient group user authority management consists of eight algorithms, which are introduced in detail in this section.

(1) **Setup**

a) The PKG chooses two multiplicative cyclic groups G_1 and G_2 of prime order p, and g is a generator of G_1. The PKG selects cryptographic hash function $H, H_1 : \{0,1\}^* \to Z_P^*$, $H_2 : \{0,1\}^* \to G_1$, the bilinear map $e : G_1 \times G_1 \to G_2$, pseudo-random function (PRF) $f : Z_P^* \times \{1,2,...,n\} \to Z_P^*$, pseudo-random permutation (PRP) $\pi : Z_P^* \times \{1,2,...,n\} \to \{1,2,...,n\}$.

b) The PKG selects elements $x_{ID} \in Z_P^*$, computes the master secret key $msk_{ID} = x_{ID}$ and master public key $mpk_{ID} = g^{x_{ID}}$.

c) The PKG randomly picks elements $u_{ID}, \mu_{ID} \in G_1$.

d) The PKG publishes the system public parameter $pp = (G_1, G_2, p, e, g, f, \pi, H, H_1, H_2, u_{ID}, \mu_{ID}, mpk_{ID})$ and holds the master secret key $msk_{ID} = x_{ID}$ private.

(2) **AuthForUser**

a) The new user enrolled into a group sends its requirement message $rm = (z^r, ID)$ to the group manager, and then obtains a valid AIA as a proof of access to the shared data. The AIA is $\Phi = z^r \cdot H(ID) + \alpha\varepsilon \mod p$ (the calculation steps of the standard AIA has been detailed discussed in Section IV).

b) The group manager adds the related information into UIT and updates the number of users in the group, namely, $UN = UN + 1$.

(3) **KeyGen**

a) The PKG picks $r_{ID} \in Z_P^*$ and computes $R_{ID} = g^{r_{ID}}$.

b) The PKG calculates $R_{SK} = r_{ID} + x_{ID} H_1(ID)$. And then the PKG returns $SK_{ID} = (R_{ID}, R_{SK})$ to the group manager.

c) After receiving the private key SK_{ID}, the group manager verifies its correctness according to (3):

$$g^{R_{SK}} \stackrel{?}{=} R_{ID}(mpk_{ID})^{H_1(ID)} \tag{3}$$

If (3) holds, the group manager accepts it and adds it into the UIT. Otherwise, reject it.

d) The group manager forwards the SK_{ID} to the group user. The specific process of the user applying for AIA and the private key is shown in Fig. 3.

(4) **TagGen**

For the group users, the calculation process of the shared data tags set is depicted as follows:

a) The user's shared file is F. Before creating the tags set, he divides F into n blocks $F = \{b_i\}_{1 \le i \le n}$, where $b_i \in Z_P^*$.

b) The user sets the $\varpi = H_2(name \| n)$, where $name \in Z_P^*$ is a random value selected as the file identifier.

c) For block b_i, the user computes $T = \{\sigma_i\}_{1 \le i \le n}$ where $\sigma_i = (\varpi \cdot \mu^{b_i})^{R_{SK}}$.

d) The user sends shared file and its corresponding tags set $\{F, T\}$ to the group manager, and deletes the local storage. Subsequently, the group manager will perform the data upload work.

(5) **UpdateForUsers**

When one user is revoked from the group, the group

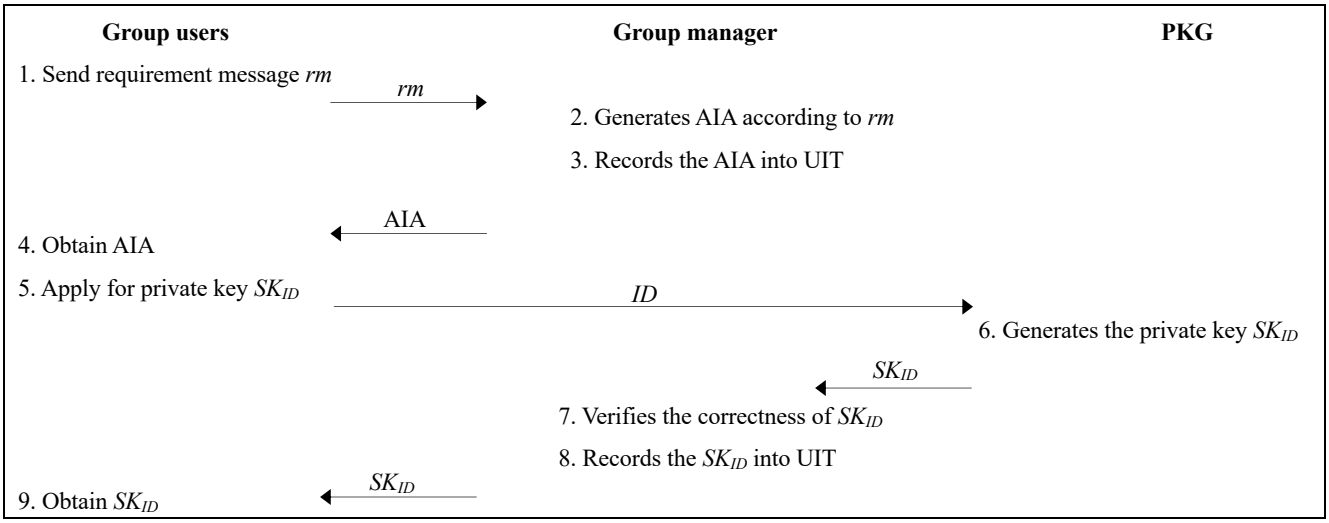

Fig. 3 The process of applying for AIA and private key

manager executes the **AuthUpdate** algorithm to recompute and redistribute AIA for the rest of group users, resets $UN = UN$ -1, and then updates the UIT (the detailed operations are mentioned in Section IV).

(6) **Challenge**

After the group users authorize the public auditing of their shared data to TPA, TPA periodically performs remote data checking, records audit results, and regularly feeds back it to the group manager. The specific process of single shared data integrity verification is as follows:

a) The TPA determines a c, where $1 \le c \le n$.

b) The TPA generates a PRP key k_1 and a PRF key k_2, where $k_1, k_2 \in Z_P^*$.

c) The TPA sends the $chal = \{c, k_1, k_2\}$ to the CSP.

(7) **ProofGen**

After receiving the challenge set *chal* sent by the TPA, the CSP responds it, generates the auditing proof, and returns to the TPA

a) The CSP computes $\{i\} = \pi_{k_1}(l)_{1 \le c \le n}, \{v_i\} = f_{k_2}(l)_{1 \le c \le n}$.

b) The CSP calculates $\lambda = \sum_{(i,v_i) \in Q} v_i b_i$, $\sigma = \prod_{(i,v_i) \in Q} \sigma_i^{v_i}$.

c) The CSP sends the auditing proof $P = \{\lambda, \sigma\}$ to the TPA.

(8) **ProofVerify**

After receiving the auditing proof $P = \{\lambda, \sigma\}$ returned by the CSP, the TPA checks its correctness through equation (4):

$$e(\sigma, g) \stackrel{?}{=} e(\prod_{(i,v_i) \in Q} \varpi^{v_i} \cdot \mu^{\lambda}, R_{ID}(mpk_{ID})^{H_1(ID)}) \quad (4)$$

If (4) holds, returns "1", which means that the shared data stored in the CSP is intact; Otherwise, returns "0". TPA carefully and accurately records the results of each audit result, and regularly feeds back the audit results to the group manager. The detailed public auditing process is given in Fig. 4.

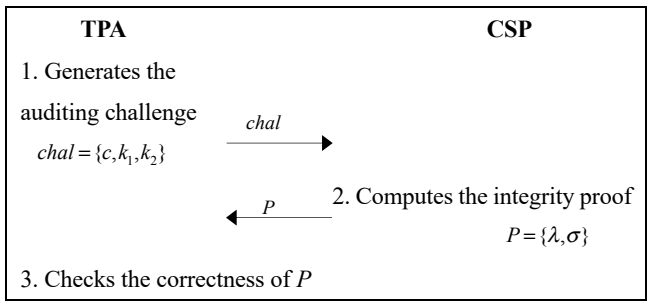

Fig. 4 The auditing process of the proposed scheme

Security Analysis

In this section, we prove that the proposed scheme is secure in term of the correctness and audit soundness.

Theorem 5 (**Correctness**) Our proposed scheme satisfies the following correctness:

a) **Private Key Correctness**: A valid private key can be passed the correctness verification of the group manager.

b) **AIA Correctness**: A valid AIA Φ can be passed the verification of the group manager.

c) **Audit Correctness**: Only when the CSP completely stores all the shared data of the user, during the audit phase, the auditing proof P he generates can pass the correctness verification of the TPA.

Proof:

a) Given a private key SK_{ID} generated by the PKG, the group manager can verify whether the left and right sides of (3) are equal to determine the availability of the SK_{ID}.

$$g^{R_{SK}} = g^{r_{ID}+x_{ID}H_1(ID)} = g^{r_{ID}} g^{x_{ID}H_1(ID)}$$
$$= g^{r_{ID}} g^{(x_{ID})H_1(ID)} = R_{ID}(mpk_{ID})^{H_1(ID)}$$

b) The AIA's correctness proof has been given in Section IV.

c) Given a proof P returned by the CSP, the TPA can verify whether the left and right sides of (4) are equal to determine the correctness of auditing proof P.

$$\begin{aligned} e(\sigma, g) &= e(\prod_{(i,v_i)\in Q} \sigma_i^{v_i}, g) \\ &= e(\prod_{(i,v_i)\in Q} ((\varpi \cdot \mu^{b_i})^{R_{SK}})^{v_i}, g) \\ &= e(\prod_{(i,v_i)\in Q} (\varpi \cdot \mu^{b_i})^{v_i}, g^{R_{SK}}) \\ &= e(\prod_{(i,v_i)\in Q} \varpi^{v_i} \cdot \prod_{(i,v_i)\in Q} \mu^{b_i v_i}, g^{r_{ID}+x_{ID}H_1(ID)}) \\ &= e(\prod_{(i,v_i)\in Q} \varpi^{v_i} \cdot \mu^{\sum_{(i,v_i)\in Q} b_i v_i}, R_{ID}(mpk_{ID})^{H_1(ID)}) \\ &= e(\prod_{(i,v_i)\in Q} \varpi^{v_i} \cdot \mu^{\lambda}, R_{ID}(mpk_{ID})^{H_1(ID)}) \end{aligned}$$

If (4) holds, it indicates that the data stored in the CSP is intact, returns "1"; Otherwise, returns "0".

Theorem 6 (**Audit Soundness**) If the CDH assumption and DL assumption hold in G_1, and the tags set is existentially unforgeable; then, the CSP cannot pass the TPA's verification with negligible probability, in the case that it does not fully possess user's intact file.

Proof:

We construct a knowledge extractor to extract the challenged block through multiple interactions with the proposed scheme. That is, if the adversary $\mathcal{A}$ can pass the TPA' verification but not possess the intact data, we can extract the challenged block by repeated interactions between the knowledge extractor and the proposed scheme. Here, we assume that the CSP acts as an adversary $\mathcal{A}$, and the TPA is regarded as the challenger $\mathcal{C}$. By executing the following series of games to implement repeated interactions between $\mathcal{A}$ and $\mathcal{C}$, the audit soundness of proposed scheme can be proved.

Game 0: **Game 0** is defined in Section III.

Game 1: **Game 1** is the same as **Game 0**, but there exists a minor difference. The challenger $\mathcal{C}$ keeps a list with recording all tags that the adversary has ever queried. Whenever the adversary $\mathcal{A}$ makes the **TagGen** query, the challenger $\mathcal{C}$ adds a record to this list.

Game 2: **Game 2** is the same as **Game 1**, but there exists a minor difference. The challenger $\mathcal{C}$ keeps a list with recording all responses that the adversary $\mathcal{A}$ has ever responded.

Consider the following situation: the adversary $\mathcal{A}$ has passed the challenger $\mathcal{C}$'s correctness verification, but he does not hold the complete challenged blocks. Once this occurs, we say that the adversary $\mathcal{A}$ has won, but the game will be aborted.

According to the above description, we know that if the response $P = \{\lambda, \sigma\}$ has passed the challenger $\mathcal{C}$'s verification, the P must be correct and can pass the verification of (5).

$$e(\sigma, g) = e(\prod_{(i,v_i)\in Q} \varpi^{v_i} \cdot \mu^{\lambda}, R_{ID}(mpk_{ID})^{H_1(ID)}) \quad (5)$$

Suppose the forged response is $P' = \{\lambda', \sigma'\}$. Note that since the adversary $\mathcal{A}$ has won, then, we have $P = P'$; but the game will be aborted. Thus, there are two situations: $\sigma \neq \sigma', \lambda = \lambda'$ or $\sigma = \sigma', \lambda \neq \lambda'$. Let 's gives the detailed analysis.

a) If the aggregated tags $\sigma \neq \sigma'$, that is, $\sigma' = \prod_{(i,v_i)\in Q} \sigma_i'^{v_i}$ is not equal σ; but the returned response $P' = \{\lambda', \sigma'\}$ has successfully passed the correctness verification, the game will be aborted.

Analysis: The forged proof is $P' = \{\lambda', \sigma'\}$. Because the forgery is successful, we have the following formula holds.

$$e(\sigma', g) = e(\prod_{(i,v_i)\in Q} \varpi^{v_i} \cdot \mu^{\lambda'}, R_{ID}(mpk_{ID})^{H_1(ID)}) \quad (6)$$

Now we construct a simulator which can break CDH assumption with the non- negligible probability, in case that the adversary $\mathcal{A}$ makes the game aborted.

The goal of the simulator is to output h^{α} when g, g^{α}, h as input. The simulator acts like the challenger $\mathcal{C}$ in **Game** 1 but has some difference.

The simulator chooses two random values $a, b \in Z_P^*$ and set $u = g^a h^b$.

Then, the simulator interacts with the adversary $\mathcal{A}$. When received the forged proof $P' = \{\lambda', \sigma'\}$, the simulator checks whether (6) holds.

Obviously, $\lambda' \neq \lambda$; otherwise, $\sigma' = \sigma$. We define $\Delta\lambda = \lambda' - \lambda$. By dividing (6) by (5), setting the public value $mpk_{ID} = g^{\alpha}$, we get:

$$\begin{aligned}e(\sigma'/\sigma,g) &= e(u^{\Delta\lambda}, R_{ID}(mpk_{ID})^{H_1(ID)}) \\ &= e((g^a h^b)^{\Delta\lambda}, g^{R_{ID}}(mpk_{ID})^{H_1(ID)}) \\ &= e(g^{a\Delta\lambda}, g^{R_{ID}}(mpk_{ID})^{H_1(ID)}) \\ &\cdot e(h^{b\Delta\lambda}, g^{R_{ID}}(mpk_{ID})^{H_1(ID)}) \\ &= e(g^{a\Delta\lambda R_{ID}} \cdot (mpk_{ID})^{a\Delta\lambda H_1(ID)}, g) \\ &\cdot e(h^{b\Delta\lambda R_{ID}}, g) \cdot e(h^{b\Delta\lambda}, (mpk_{ID})^{H_1(ID)}) \\ &= e(g^{a\Delta\lambda R_{ID}} \cdot (mpk_{ID})^{a\Delta\lambda H_1(ID)} \cdot h^{b\Delta\lambda R_{ID}}, g) \\ &\cdot e(h^{b\Delta\lambda}, (mpk_{ID})^{H_1(ID)})\end{aligned}$$

Next, we have

$$\begin{aligned}&e(\sigma' \cdot \sigma^{-1} \cdot g^{-a\Delta\lambda R_{ID}} \cdot (mpk_{ID})^{-a\Delta\lambda H_1(ID)} \cdot h^{-b\Delta\lambda R_{ID}}, g) \\ &= e(h^{b\Delta\lambda}, (mpk_{ID})^{H_1(ID)}) \\ &= e(h, mpk_{ID})^{b\Delta\lambda H_1(ID)} \\ &= e(h, g^{\alpha})^{b\Delta\lambda H_1(ID)} \\ &= e(h^{\alpha}, g)^{b\Delta\lambda H_1(ID)}\end{aligned}$$

Further, we can obtain $h^{\alpha} = (\sigma' \cdot \sigma^{-1} \cdot g^{-a\Delta\lambda R_{ID}} \cdot (mpk_{ID})^{-a\Delta\lambda H_1(ID)} \cdot h^{-b\Delta\lambda R_{ID}}, g)^{1/b\Delta\lambda H_1(ID)}$. If the CDH problem can be solved in G_1, then h^{α} must be calculable. So, it's clearly that the probability of solving the CDH problem in G_1 is equivalent to compute the probability of $b\Delta\lambda H_1(ID) = 0$. The probability of $b\Delta\lambda H_1(ID) = 0$ is $1/p$ which is negligible since p is a large prime. So, the probability of solving the CDH problem in G_1 is 1-1/p. Further, the probability of the game abort is 1-1/p.

Hence, if there is a non- negligible difference between the adversary $\mathcal{A}$'s probabilities of success in **Game** 1 and Game 2, then the constructed simulator can solve the CDH problem.

b) If the aggregated message $\lambda \neq \lambda'$; but the returned response $P' = \{\lambda', \sigma'\}$ has successfully passed the correctness verification, the game will be aborted.

Analysis: We also construct a simulator can break DL assumption with the non- negligible probability, in case that the adversary $\mathcal{A}$ makes the game aborted.

The goal of the simulator is to output α when $g, h = g^{\alpha}$ as input. The simulator acts like the challenger $\mathcal{C}$ in **Game** 1 but has some difference.

The simulator also chooses two random values $a, b \in Z_P^*$ and set $u = g^a h^b$.

Here, we have $\sigma' = \sigma$ but $\lambda' \neq \lambda$. Then we define $\Delta\lambda = \lambda' - \lambda$. By dividing (6) by (5), we can obtain:

$$1 = u^{\Delta\lambda} = (g^a h^b)^{\Delta\lambda} = g^{a\Delta\lambda} \cdot h^{b\Delta\lambda}$$

Obviously, that $\Delta\lambda \neq 0 \bmod p$; otherwise, $\lambda' = \lambda \bmod p$ which contradicts after mentioned assumption.

Further we have $h = g^{-a\Delta\lambda/b\Delta\lambda} = g^{a/b}$. So, the probability of solving the DL problem in G_1 is equivalent to compute the probability of b=0. The probability of b=0 is 1/p which is negligible since p is a large prime. So, the probability of solving the DL problem in G_1 is 1-1/p. Further, the probability of the game abort is 1-1/p.

Hence, if there is a non- negligible difference between the adversary $\mathcal{A}$'s probabilities of success in **Game** 1 and **Game** 2, then the constructed simulator can solve the DL problem.

Based on the above proof, we can know that the differences between these games can be ignored. Further, it can be inferred that if the CSP does not store intact data of user, it must not pass the TPA's verification.

Performance Evaluation

In this section, we first give the functionality comparison between our scheme and other related schemes and show the computation and communication overheads of a group user at the different phases. Then we evaluate the performance of the proposed scheme through several experiments.

Functionality Comparison

We discuss the functionality comparison between our scheme and scheme [3], [24], [29], [30] in terms of public verification, group authority management and user revocation. The results are shown as Table 2. Note that our scheme is the only one that satisfies those four properties. Moreover, none of the schemes [3], [24], [29], [30] can support group authority management.

Performance Analysis and Comparison

For simplicity, we give the meaning of the following notations firstly. Here $Hash_{G_1}, Hash_{Z_P^*}$ denote the hash operation in G_1, Z_P^*. $Exp_{G_1}, Exp_{G_T}, Exp_{Z_P^*}$ express the

Table 2 Functionality comparison with other related schemes

Schemes	Public verification	Group authority management	User revocation
Shacham et al. [3]	Yes	No	No
Wang et al. [24]	No	No	No
Zhang et al. [29]	Yes	No	Yes
Fu et al. [30]	Yes	No	Yes
Our scheme	Yes	Yes	Yes

Table 3 The computation overhead of a group user at different phases

Phase	Apply for Authorization	AIA Generation	AIA Update	Proof Generation	Proof Verification
Computation overhead	$Exp_{Z_P^*}$	$Hash_{Z_P^*} + Mul_{Z_P^*} + Exp_{Z_P^*} + Add_{Z_P^*}$	$Hash_{Z_P^*} + Mul_{Z_P^*} + Exp_{Z_P^*} + Add_{Z_P^*}$	$cExp_{G_1} + (c-1)Mul_{G_1} + cMul_{Z_P^*} + (c-1)Add_{Z_P^*}$	$2Pair + (c+2)Exp_{G_1} + 2Mul_{G_1} + (c-1)Add_{G_1} + cHash_{G_1} + Hash_{Z_P^*}$

Table 4 The communication overhead of a group user at different phases

Phase	Apply for Authorization	AIA Generation	AIA Update	Challenge	Proof Generation
Communication overhead	$2\|p\|$	$\|p\|$	$\|p\|$	$\|p\|+\log_2 c$	$\|p\|+\|q\|$

exponentiation operation in G_1, G_T, Z_P^*. $Mul_{G_1}, Mul_{G_T}, Mul_{Z_P^*}$ mean the multiplication operation in G_1, G_T, Z_P^*. $Pair$ represents the pairing operation. $Add_{G_1}, Add_{Z_P^*}$ indicate the addition operation in G_1, Z_P^*. c indicates the number of the challenged blocks and n is the total number of data blocks. $|n|$ is the size of an element of set [1, n]. $|p|$ is the size of an element in Z_P^*, and $|q|$ is the size of an element in G_1.

(1) **Performance Analysis**

In Table 3 and Table 4, we show the computation and communication overheads of a group user at different phases.

a) Computation overhead. In Table 3, we show the computation overheads of a group user at different phases of the proposed scheme. In the apply for authorization phase, users who apply to join the group need to generate a requirement message *rm*, and the computation overhead is $Exp_{Z_P^*}$. In AIA generation phase, the group manager generates an AIA for users according to its *rm*, and the computation overhead is $Hash_{Z_P^*} + Mul_{Z_P^*} + Exp_{Z_P^*} + Add_{Z_P^*}$. In AIA update phase, the group manager needs to reselect random elements to get the new z^r, but spending is the same as the AIA generation phase. The computation overhead of proof generation phase is $cExp_{G_1} + (c-1)Mul_{G_1} + cMul_{Z_P^*} + (c-1)Add_{Z_P^*}$. In order to conduct the integrity verification, the TPA needs to check the correctness of (4). For computing $\prod_{(i,v_i)\in Q} \varpi^{v_i} \cdot \mu^{\lambda}$, our scheme requires to perform these operations in G_1, including (c+1) exponentiation operations, 2 multiplicative operations, (c-1) addition operation and c hash operations. In addition, the computation cost of calculating $R_{ID}(mpk_{ID})^{H_1(ID)}$ is $Exp_{G_1} + Mul_{G_1} + Hash_{Z_P^*}$.

b) Communication overhead. In Table 4, we also discuss the communication overheads of a group user at different phases of the proposed scheme. In the apply for

Table 5 Comparison in different schemes

Scheme	Proof generation	Proof verification
Wang et al. [24]	$cExp_{G_1}+(c-1)Mul_{G_1}$ $+cMul_{Z_P^*}+(c-1)Add_{Z_P^*}$	$4Pair+17cExp_{G_1}+11cMul_{G_1}+2cExp_{Z_P^*}$ $+(4c+k)Mul_{Z_P^*}+cExp_{G_T}+cMul_{G_T}$
Zhang et al. [29]	$cExp_{G_1}+(c-1)Mul_{G_1}$ $+cMul_{Z_P^*}+(c-1)Add_{Z_P^*}$	$2Pair+(2c+3)Exp_{G_1}+(2c+2)Mul_{G_1}$ $+(c-1)Add_{Z_P^*}+cMul_{Z_P^*}+cHash_{G_1}+2Hash_{Z_P^*}$
Fu et al. [30]	$cExp_{G_1}+(c-1)Mul_{G_1}$ $+cMul_{Z_P^*}+(c-1)Add_{Z_P^*}$	$2Pair+2cExp_{G_1}+2cMul_{G_1}+22cExp_{Z_P^*}+14cMul_{Z_P^*}$
Our scheme	$cExp_{G_1}+(c-1)Mul_{G_1}$ $+cMul_{Z_P^*}+(c-1)Add_{Z_P^*}$	$2Pair+(c+2)Exp_{G_1}+2Mul_{G_1}$ $+(c-1)Add_{G_1}+cHash_{G_1}+Hash_{Z_P^*}$

authorization" phase, the communication overhead comes from transmitting *rm*. In order to distribute the AIA for every group user, the communication cost is the same and all requires $|p|$ in the AIA generation and AIA update phases. In our proposed scheme, the communication overhead is mainly from the integrity auditing process. In the challenge phase, the TPA sends $chal=\{c,k_1,k_2\}$ to the CSP, and the size of auditing challenge is $|p|+\log_2 c$ bits. To generate the auditing proof, the CSP compute $P=\{\lambda,\sigma\}$ and reply the TPA. And the size of an auditing proof is $|p|+|q|$ bits.

(2) **Performance Comparison**

In Table 5, we compare the computation overhead in terms of proof generation and proof verification algorithms with Wang et al.'s scheme [24], Zhang et al.'s scheme [29] and Fu et al.'s scheme [30]. From the Table 5, we know that all schemes have the same computation overhead at the phase of proof generation. It should be noted that both schemes [24] and [30] are implemented by means of asymmetric bilinear pairing $G_1\times G_2\to G_T$. However, scheme [24] involves the multiplicative and exponentiation operations on G_T, while scheme [30] does not. We also know that the scheme [29] uses the most types of calculations, and scheme [30] the least. To verify the correctness of auditing proof *P*, the computation cost of our scheme is $2Pair+(c+2)Exp_{G_1}+2Mul_{G_1}+(c-1)Add_{G_1}+cHash_{G_1}+Hash_{Z_P^*}$.

Experimental Results

In this subsection, we evaluate the performance of the proposed scheme by several experiments. We run a series of experiments on a 1.8 GHZ Intel Core i5 processor and 8GB RAM. All the experiments using the Type A with the free Pairing- Based Cryptography (PBC) Library. In the implementation, we choose the based filed size to be 512 bits, and the size of Z_P^* to be 160 bits, this is, |*p*| = 160 bits. We set the file size to 20MB which divided into 1,000,000 data blocks.

(1) **Standard AIA generation phase**

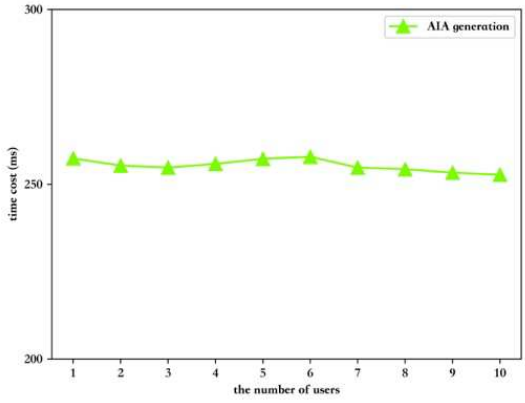

Fig. 5 The time cost of AIA generation.

In the first part, we evaluate the computation cost of AIA generation phase. It can be seen from Fig. 5 that as the number of users increases, the time to calculate AIA remains almost unchanged, and it is parallel to the x-axis,

which takes about 260ms. The reason is that the AIA generated by the group manager for the user is calculated separately, and the overhead of calculation required is small.

(2) **Auditing phase**

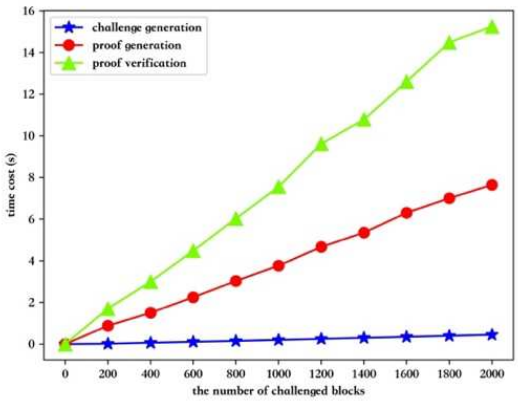

Fig. 6 Computation cost in the auditing process.

The integrity auditing process divide into challenge generation, proof generation and proof verification algorithms. In order to evaluate the computation cost during the auditing process of our scheme, we set the number of challenged blocks from 0 to 2000, with 200 as the interval. As shown in Fig. 6, we can see that the time cost in the three phases is directly proportional to the number of data blocks to be challenged. That is, as the number of challenged data blocks increases, the calculation overhead also increases. The challenge phase takes the least time, and the time cost varies from 0.015840s to 0.450362s. When the number of challenged data blocks equals to 200, the time of proof generation is 0.882022s; while it needs nearly 7.645638s when the challenged data blocks is 2000. Because different kinds of operations are involved, the proof verification phase is most time-consuming. And the time ranged from 1.698569s to 15.251921s.

(3) **Proof Verification phase**

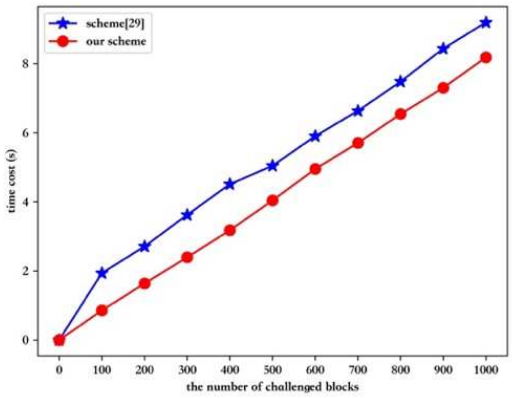

Fig. 7 Comparison of the time cost in the proof verification phase.

In the third part, we evaluate the performance of proof verification phase through comparing our scheme with Zhang et al.'s scheme [29]. We set the number of the challenged blocks from 0 to 1000, increased by an interval of 100. Fig. 7 gives more details. It is clear that our scheme spends less than Zhang et al.'s scheme [29], which is consistent with the results obtained by the previous computational overhead analysis. As the number of challenged data blocks increases, the time cost in the proof verification phase of two schemes increases linearly. And in this phase, our scheme time ranged from 0.857812s to 8.184904s, while the Zhang et al.'s scheme [29] varied from 1.939095s to 9.195359s.

Conclusion

In this paper, we propose a remote data possession checking scheme for shared data, which can support group user authority management within the group. In order to realize group user authority management, we introduce a new entity – group manager, and further propose a standard AIA scheme. In addition, we propose a public auditing scheme for non-revoked users' shared data; and it's worth mentioning that in our scheme, the cost of user revocation is no longer depended on the number of data blocks, but only related to the number of users. Our proposed scheme is designed based on IBE, which reduce the necessary certificate management cost caused by PKI distribution of private keys. Security analysis and experimental results demonstrate that the

scheme is safe and feasible.

Abbreviation

PKI: Public Key Infrastructure; CSP: Cloud Service Provider; PKG: Private Key Generator; TPA: Third- Party Auditor; CDH: Computational Diffie- Hellman; DL: Discrete Logarithm; AIA: Authorization Invisible Authenticator; UIT: User Information Table.

Acknowledgements

The authors would like to thank the editor and the anonymous reviewers who have helped to improve the paper.

Authors' contributions

Yilin Yuan conceived ideas and wrote the manuscript. Jianbiao Zhang gave the constructive suggestions and checked the manuscript. Wanshan Xu, Xiao Wang and Yanhui Liu reviewed and checked the manuscript.

Funding

This work was supported in part by the Natural Science Foundation of Beijing Municipality under Grant M21039.

Availability of data and materials

Data sharing not applicable to this paper as no datasets were generated or analyzed during the current study.

Competing interests

The authors declare that they have no competing interests.

Author details

[1] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China. [2] Beijing Key Laboratory of Trusted Computing, Beijing 100124, China. [3] Department of Information Science and Technology, Tianjin University of Finance and Economics, Tianjin 300222, China

References

1. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," in Proc. 14th ACM Conference on Computer and Communications Security. pp. 598-609, 2007.
2. A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," in Proc. 14th ACM Conference on Computer and Communications Security, pp. 584-597, 2007.
3. H. Shacham and B. Waters, "Compact Proofs of Retrievability," in Journal of Cryptology, vol. 26, no. 3, pp. 442-483, Jul, 2013.
4. G. Ateniese, R. Dipietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," in Proc. 4th International conference on Security and Privacy in Communication Netowrks, pp. 1-10, 2008
5. C. C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," in Proc. 16th ACM Conference on Computer and Communications Security, pp. 213-222, 2009.
6. Q. Wang, C. Wang, K. Ren, W. J. Lou and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," in IEEE Transactions on Parallel and Distributed Systems, vol 22, no 5, pp. 847 – 859, May 2011.
7. A. F. Barsoum and M. A. Hasan, "Provable Multicopy Dynamic Data Possession in Cloud Computing Systems," in IEEE Transactions on Information Forensics and Security, vol. 10, no. 3, pp. 485 - 497, Mar 2015.
8. J. Yu, K. Ren, C. Wang and V. Varadharajan, "Enabling Cloud Storage Auditing with Key-Exposure Resistance," in IEEE Transactions on Information Forensics and Security, vol 10, no. 6, pp. 1167 - 1179, Jun 2016.
9. M. Sookhak, F. R. Yu and A. Y. Zomaya, "Auditing Big Data Storage in Cloud Computing Using Divide and Conquer Tables," in IEEE Trans. on Parallel and Distributed Systems, vol 29, no 5, pp. 999-1012, Dec 2017.
10. K. He, J. Chen, Q. Yuan, S. L. Ji, D. B. He and R. Y. Du, "Dynamic Group-Oriented Provable Data Possession in the Cloud," in IEEE Trans. on Dependable and Secure Computing, 2019.
11. Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen and Y. X. Zhang, "Blockchain Empowered Arbitrable Data Auditing Scheme for Network Storage as a Service," in IEEE Trans. on Services Computing, Vol 13, no 2, pp. 289-300, 2019.
12. Y. J. Wang, Q. H. Wu, Q. B, S. H. Tang and S. Willy. "Online-offline Provable Data Possession," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 5, pp. 1182 – 1194, May 2017.
13. A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," in Cryptology, vol 196, pp.47-53, 1984.
14. D. Boneh and M. Franklin, "Identity-based encryption from the weil-pairing," in Cryptology, vol. 2139, pp. 213-229, 2001.
15. H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based remote data possession checking in public clouds," IET Inf. Secur., vol. 8, no. 2, pp. 114–121, 2014.
16. H. Q. Wang, "Identity-Based Distributed Provable Data Possession in Multicloud Storage," in IEEE Trans. on Services Computing, Vol 8, no 2, pp. 328-340, Mar 2014.
17. H. Q. Wang, D. B. He and S. H. Tang, "Identity-Based Proxy-Oriented Data Uploading and Remote Data Integrity Checking in Public Cloud," in IEEE Trans. on Information Forensics and Security, vol 11, no 6, pp. 1165 – 1176, Jun 2016.
18. W. T. Shen, J. Qin, J. Yu, R. Hao and J. K. Hu, "Enabling Identity-Based Integrity Auditing and Data Sharing With Sensitive Information Hiding for Secure Cloud Storage," in IEEE Trans. on Information Forensics and Security, vol 14, no 2, pp. 331 – 346, Feb 2019.
19. Y. N. Li, Y. Yu, G. Y. Min, W. Susilo, J. B. Ni and K. R. Choo, "Fuzzy Identity-Based Data Integrity Auditing for Reliable Cloud Storage Systems," in IEEE Trans. Dependable Secure Comput., vol. 16, no. 1, pp. 72- 83, 2019.
20. A. Sahai and B. Waters, "Fuzzy Identity- Based Encryption," in Proc. Annu. Int. Comf. Theory Appl. Cryptographic Tech., May 2005: 457-473.

21. Y. Zhang, C. X. Xu, X. D. Lin and X. S. Shen, "Blockchain-Based Public Integrity Verification for Cloud Storage against Procrastinating Auditors," in IEEE Trans. on Cloud Computing, Mar 2019.
22. W. T. Shen, G. Y. Yang, J. Yu, H. L. Zhang, F. Y. Kong and R. Hao, "Remote data possession checking with privacy-preserving authenticators for cloud storage," Future Generation Comput. Syst., vol. 76, pp. 136–145, 2017.
23. L. Xue, Y. Yu, Y. N. Li, M. H. Au, X. J. Du and B. yang, "Efficient attribute-based encryption with attribute revocation for assured data deletion," in Information Sciences, vol. 479, pp. 640-650, Apr 2019.
24. B. Y. Wang, B. C. Li and H. Li, "Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud," in Proc International Conference on Applied Cryptography and Network Security, pp. 507-525, Jun 2012.
25. B. Y. Wang, B. C. Li and H. Li, "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," in IEEE Trans. on Services Computing, vol. 2, no. 1, pp. 43-56, 2014.
26. B. Y. Wang, B. C. Li and H. Li, "Panda: Public Auditing for Shared Data with Efficient User Revocation in the Cloud," in IEEE Trans. on Services Computing, vol. 8, no. 1, pp. 92-106, 2015.
27. J. W. Yuan and S. C. Yu, "Public Integrity Auditing for Dynamic Data Sharing With Multiuser Modification," in Trans. on Information Forensics and Security, vol 10, no 8, pp. 1717 – 1726, Aug 2015.
28. H. Tian, F. L. Nan, H. Jiang, C. C. Chang, J. T. Ning and Y. F. Huang, "Public auditing for shared cloud data with efficient and secure group management," in Information Sciences, vol. 472, pp. 107-125, Jan 2019.
29. 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," in Trans. on Dependable Secure Comput., vol. 17, no. 3, pp. 608-619, 2020.
30. A. M. Fu, S. Yu, Y. Q. Zhang, H. Q. Wang and C. Y. Huang, "NPP: A New Privacy-Aware Public Auditing Scheme for Cloud Data Sharing with Group Users," in Trans. on Big Data, May 2017.
31. Y. Luo, M. Xu, S. Fu, D. Wang, and J. Deng, "Efficient integrity auditing for shared data in the cloud with secure user revocation," in Proc. IEEE Trustcom/BigDataSE/ISPA, pp. 434–442, 2015.