Proof of Storage with Corruption Identification and Recovery for Dynamic Group Users

Tao Jiang[1,4], Hang Xu[1], Qiong Cheng[2], and Wenjuan Meng[3(✉)]

[1] School of Cyber Engineering, Xidian University, Xi'an, China
taojiang@xidian.edu.cn, hangxu@stu.xidian.edu.cn
[2] Guangzhou Institute of Technology, Xidian University, Guangzhou, China
qcheng2020@stu.xidian.edu.cn
[3] College of Information Engineering, Northwest A&F University, Yangling, China
wjmeng2000@gmail.com
[4] Guangxi Key Laboratory of Cryptography and Information Security, Guilin, China

Abstract. With the widespread deployment of cloud storage, protecting the security and privacy of outsourced data becomes increasingly important. In the last decade, provable data possession/retrievability schemes with different flavors, such as public variability, efficient update, user revocation, protecting privacy, have been designed to protect the rights and interests of cloud users. Although extensive research have been conducted in protecting the security and privacy of group cloud users, there is still a lack of a way to efficiently identify the destroyed data and actualize cloud disaster recovery. In response to these problems, a publicly verifiable proof of storage scheme is proposed for group cloud users in this paper. We provide the construction details, where invalid files can be efficiently identificated and even recovered after being detected by a trusted third-party auditor. Further analysis indicate that the designed scheme is secure for dynamic group user data sharing.

Keywords: Cloud storage · Data sharing · Corruption identification · Proof of retrievability · Dynamic group

1 Introduction

With the rapid development of the Internet of Things, cloud computing, big data and other digital technologies, cloud storage services are widely used from all walks of life around the world. In cloud storage, users can access their data whenever and wherever [17], and even further reduce the storage cost by sharing their data with a group of users. However, in the outsourcing storage mode, users lose control of their data, and the cloud service provider may not be fully trusted [20]. Thus users may lose their data for a variety of reasons. For instance, Amazon has admitted that a small fraction of their customers' data was permanently lost

Y. Chenggang et al. (Eds.): MobiMedia 2022, LNICST 451, pp. 126–141, 2022.
https://doi.org/10.1007/978-3-031-23902-1_10

due to several days' electricity outages caused by lightning strikes [3]. Also, cloud data may be destroyed due to hardware/software failure [18] or misoperations [10]. In fact, when data is outsourced, it is difficult for users to verify that their data is corrupted without inuring serious communication cost. This security problem makes people reluctant to storage their data to the cloud.

To address this concern, various interactive data integrity/recoverability auditing schemes, e.g., provable data possession (PDP) [1] and proofs of retrievability (PoR) [7,12], are designed. Later, PDP and PoR are extended to support efficient and privacy-preserving auditing for shared cloud data [4–6,8,13,21]. Although the above auditing schemes can attest the availability of outsourced data, they can not quickly identify the invalid files/blocks among the data and recover them after the failure. The reason lies in that, a single destroyed file/block can cause the audit to fail when an integrity audit is performed on large-scale data stored on the server. After the failure of data integrity auditing, searching for the location of invalid files/blocks brings a lot of computation and communication overhead, which seriously affects the efficiency of the scheme. To solve the above challenge, the invalid files/blocks identification schemes based on different signature/tag failure identification technologies have been published, such as pruned search tree [9] or binary tree with shared randomizers [2], Exponents Test Based Search (ETBS) [14]. Also, there are many schemes proposed for recovering damaged data, such as extended POR [11] which erasures coding technique [22]. However, the above schemes are mainly designed for single client scenario and have a single function. More precisely, the auditing tags from multiple users are computed with different private keys, which makes it difficult and inefficient in aggregating the authentication tags and identifying the files/blocks in one challenge. Therefore, it is still challenging to design an auditing scheme to efficiently support location of the failed files/blocks and recovery them for dynamic group users with user revocation.

In this paper, we address these problems and propose a new solution for fast locating and recovering data of invalid files for dynamic group users. It enables users to efficiently identify the availability of all outsourced data from valid group users without multiple attestations. Notably, in our scheme, users need to process/upload their original files and corresponding tags only once irrespective of the group users' revocation. Especially, our scheme leverages rapid location and recovery techniques for corrupted data to impose significantly. By doing so, our scheme could efficiently identify the corruption files/blocks from different cloud users at one attestation.

Contribution. In summary, the paper makes the following contributions:

- We figure out that designing storage proof with efficient identifications of corrupt files and recovery for dynamic group users is still an unsolved problem. And on this basis, a corruption identification and recovery scheme dubbed CIRG is proposed in this paper.
- We provide concrete construction for CIRG, which transforms and accumulates data integrity audit results through association calculation and accumulation calculation, reducing the number of calculations for locating damaged

data. Aiming at the problem of recovery of damaged data of group users, the RS erasure code technology and the idea of shared coding are combined to achieve reliable recovery of damaged data, so that damaged data belonging to existing users of the group or revoked users can be effectively recovered.
– We prove the security of CIRG under its threat model, which indicates that the designed scheme is secure in the auditing of data integrity and efficient in the recovery of the corrupt files.

1.1 Related Work

In recent years, various cloud storage data integrity audit verification schemes have been proposed. For example, Ateniese et al. proposed the concept of "provable data ownership" PDP for the first time in [1], and constructed homomorphic verification tags using RSA scheme and discrete logarithms, which saved the retrieval and download of files and protected data privacy. Shacham et al. [12], based on Ateniese et al.'s idea about homomorphic tags, used the BLS short message signature mechanism to construct the Homomorphic verification tag, which had reduced the communication overhead in the verification stage. Jules et al. [7] first considered how to recover the damaged data, and proposed a sensor-based POR scheme, which could not only conduct data integrity audit, but also recover the failure data to a certain extent. However, this scheme is only suitable for individual users. Matt et al. [9] proposed a pairing based signature scheme to identify multiple invalid files after batch audit failure, which substantially reduced the pairing computation cost by leveraging "divide-and-conquer" with pruning the search tree. Bernstein et al. [2] proposed a scheme for locating failure files after the batch audit of elliptic curve signatures, which could effectively reduce the cost of elliptic curve failure signature recognition based on binary tree with shared randomizer. Shin et al. [14] proposed the Exponents Test Based Search (ETBS), which identified the location of the failed files/blocks by comparing the validation values with and without exponents. ETBS, however, only applies to invalidation scenarios for individual files. This method has a large limitation, which is not suitable for the location of multiple failure files. Wang et al. [16] proposed a batch audit method to support fast query of invalid files. The characteristic of this method is that it can resist "invalid file" attack effectively and ensure the availability and efficiency of batch audit scheme.

Data sharing is one of the most widely used services provided by cloud storage. With data sharing services, users can share their data with a group of users on the cloud, reducing the burden of local data storage. When a group user acts improperly or leaves the group, the user shall be revoked from the group, and the data of the revoked user shall be managed by a legal user designated by the group administrator. With the rapid development of group information sharing, related technologies of group user cloud storage have been applied accordingly. Jiang et al. [6] proposed an audit scheme of user revocation of shared data integrity based on group signature, but the efficiency was slow. Recently, Zhang et al. [21] proposed an efficient user revocation data integrity audit scheme, which effectively improved the audit efficiency of user revocation. However, these schemes only

focus on the efficiency of data audit and do not consider the failure of data audit. In fact, data integrity audit fails means the data stored on the cloud server is damaged and no longer available, which is a direct cost to users. Therefore, it is very important to design a method that can efficiently locate and recover invalid files.

2 Models and Goals

This section firstly describes the system model and the threat model. Then, the design goals of the system is listed.

2.1 System Model

In our system, as shown in Fig. 1, there are five entities: multiple group users, a group manager, a Cloud Service Provider (CSP), a Private Key Generator ($\mathcal{PKG}$), and a trusted Third Party Auditor (TPA). Following, we introduce the functionalities of different entities in our solution.

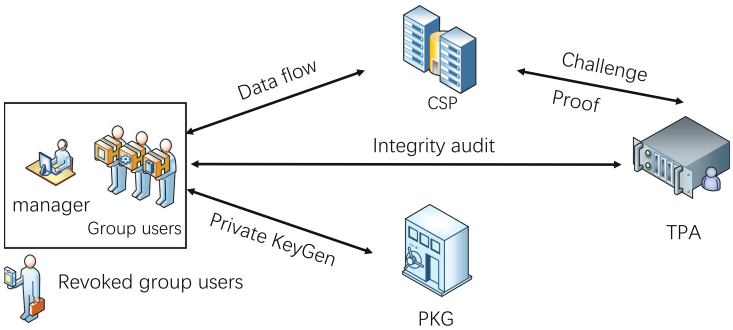

Fig. 1. The system model.

1. Multiple group users: Group users are the data owners, which share the data stored at the cloud server. They encode the files and tag each block of the files for integrity auditing and destroy recovery before outsourcing them to the cloud server. Users may join or leave a group because of personal or group management reasons. We assume that the group users are honest and will not share the private information with any illegal parity, including the revoked group users.
2. Group manager: The group manager is an administrator of a group, which manages the revocation or registration of users when group users leave or new users join the group. It also give assistance to private key generation.
3. CSP: The CSP stores the outsourced file from cloud users and allows group users to enjoy the data sharing service.

4. $\mathcal{PKG}$: The $\mathcal{PKG}$ is a trusted entity. It is responsible for generating the public parameters and keys through the group's identity (UI).
5. TPA: The TPA is responsible for auditing the integrity of cloud data on behalf of group users. Also, it assists the group users or CSP to efficiently locate the invalid file and recover the destroyed file(s) once the data integrity audit fails. The TPA cannot get the real data from the blinded ones of users in the phase of data uploading, nor can it derive the real data from the cloud's response in the phase of auditing.

2.2 Threat Model

This paper further considers the issue of remediation after a failed audit of data integrity. Specifically, once the data integrity audit fails, TPA is able to efficiently locate damaged files. It returns a list of damaged files to the user (or CSP), and then restores the corresponding invalid files based on those locations.

1. File corruption: The files outsourced by a group of cloud users may be corrupted or deleted due to server hardware failure, malicious server or human error. To avoid damage to the server and its reputation, CSP does not tell the user that the data has been corrupted.
2. File irretrievability: When the attestation from TPA reveals that the files are corrupted, the group users and the CSP may not be able to recover the file without a specific file recovery mechanism.

2.3 Design Goals

To ensure secure cloud storage auditing for shared data while supporting invalid files locating and recover efficiently, our scheme should meet the following goals:

Security Goals. For security, we consider the correctness and soundness of the designed scheme. *Correctness* requires that if the outsourced files are intact, the CSP can always pass the data integrity audit. When the destroyed file blocks are under a specific threshold, any group user or the CSP can always recover these blocks. *Soundness* means that if the files from the group users are not stored, less stored or wrongly stored, the CSP can pass the data integrity audit with a negligible probability. If there are invalid files in an attestation, the probability that the TPA cannot locate the destroyed locations is negligible.

Efficiency Goals. Efficiency goals consist of the computing, communication and storage efficiency of our scheme. In our design, we consider efficient file recovery and public auditing. The former means that when the data stored in the cloud server is damaged or lost due to physical disk damage, failure and other reasons, the proposed scheme ensures that users can effectively recover the failed files by calculating valid blocks of the damaged data. The latter means that TPA, on behalf of a group of users, is able to verify the integrity of stored data without accessing shared data stored in the cloud server.

3 Preliminaries and Definitions

This section firstly describes the cryptographic tools as preliminaries. Then, the definition of data encoding and decoding are listed.

3.1 Preliminaries

Bilinear Map. A bilinear map is a map $e : G_1 \times G_1 \rightarrow G_2$, where G_1 is a Gap Diffie-Hellman (GDH) group and G_2 is another multiplicative cyclic group of prime order p with the following three properties: 1) *Bilinear:* for all $P, Q \in G_1$ and $a, b \in Z_p$, $e(P^a, Q^b) = e(P, Q)^{ab}$. 2) *Non-Degenerate:* $e(P, Q) \neq 1$, where P, Q are generators of G_1. 3) *Computable:* there exists an efficiently computable algorithm for computing e.

Hash Function of Cryptography. Set three different cryptographic hash functions. Let $H_1 : \{0,1\}^* \times G_1 \rightarrow Z_q^*$ be a hash function computing a number in Z_q^* with an input bit string of arbitrary length and Elements in G_1, $H_2 : \{0,1\}^* \times G_1 \times \{0,1\}^* \rightarrow Z_q^*$ be a hash function computing a number in Z_q^* with two input bit strings of arbitrary length and Elements in G_1, and $h : \{0,1\}^* \rightarrow G_1$ be a BLS hash. All of these hashes are modeled as random oracles and selected randomly in this paper by $\mathcal{PKG}$.

Recovery of Damaged Data of Group Users. This paper provides efficient recovery of damaged data through erasure coding techniques. RS erasure code firstly implements redundant coding on the original data F. Redundant coding is the expansion of F. The data obtained after the extended calculation is called encoded data E. Obviously, E has a larger size than F. We call this increase in E as redundancy *red*. E is finally outsourced to the cloud server. In the following, we define the encode and decode algorithms of RS erasure code:

Definition 1. *Encode*$(m_1, m_2, \cdots, m_k) \rightarrow (\omega_1, \omega_2, \cdots, \omega_n)$*: On input a message* $F = m_1, m_2, \cdots, m_k$*, return the encode data* $E = \omega_1, \omega_2, \cdots, \omega_n$ *where* $m_i, \omega_j \in Z_q^*, i \in [1, k], j \in [1, n], n = red + k$.

Definition 2. *Decode*$(\omega'_1, \omega'_2, \cdots, \omega'_k) \rightarrow (m_1, m_2, \cdots, m_k)$*: On input an arbitrary selection of* k *valid data blocks from the corrupted data* $E' = \omega'_1, \omega'_2, \cdots, \omega'_k$*, return the message* $F = m_1, m_2, \cdots, m_k$ *where* $m_i, \omega'_i \in Z_q^*, i \in [1, k]$.

3.2 Definitions

We target at designing a proof of storage scheme, called CIRG for short, which can efficiently identify the corruption files/blocks and support their recovery for dynamic group users. CIRG defines six algorithms: Setup, KeyGen, TagGen, Challenge, Prove and Recover, which behave as follows:

1. Setup *algorithm*: the setup algorithm is run by the $\mathcal{PKG}$. It takes as input a security parameter κ and outputs a secret key sk, a public key pk and the system public parameters *tuple*.

2. KeyGen *algorithm*: the private key generation algorithm is run by the group users, group manager and $\mathcal{PKG}$ to generate keys for data labels.
3. TagGen *algorithm*: the algorithm is run by the group users. Group users encode the data uploaded to the cloud server as shown in Fig. 2. It takes as inputs the encoded file E and σ, and generates a tag *tag* for each block. The encoded data and *tag* are then uploaded to the cloud server for storage.
4. Challenge *algorithm*: the challenge algorithm is run by the TPA. It takes as input the file information for the challenge and returns challenge chal to the CSP.
5. Prove *algorithm*: the proof generation algorithm is run by the CSP. It takes as inputs batch file $F_1, F_2, \cdots, F_N$, a set of the corresponding authenticators and auditing challenge chal, and generates a proof P which is used to prove the cloud accurately stores $F_1, F_2, \cdots, F_N$.
6. Recover *algorithm*: the proof verification algorithm is run by the TPA. It takes as inputs a proof P and system public parameters, and returns "success" if the proof is valid. Otherwise, returns "failure", which means that the stored file is damaged. At this point, RDI accumulation method is called to quickly locate the damaged file. Then, the user recovers invalid files according to the RS erasure code.

4 CIRG Scheme

In this section, we provide the construction details of CIRG scheme. Notably a high-level explanation of the proposed scheme is given before describing the details of each algorithm.

4.1 High-Level Explanation

The solution in [21] effectively solves the problem of data integrity audit when group users cancel. However, it fails for data integrity audit which means that there are invalid files in the audited data. In fact, querying and recovering invalid files require a lot of communication and computing resources. To address the challenge, we apply RS erasure codes to the data, as shown in Fig. 2. The user encodes the data before storing it on the cloud server. When a file auditing fails, the invalid data can be recovered effectively according to the RS erasure code. In the proposed CIRG structure, we start with a batch file audit. When it fails, the RDI is called to quickly locate the failed files. Then the RS erasure code is used to recover the damaged files. Specifically, after a failure of batch audit, the RDI algorithm is used to correlate the left and right sub-tree audit values of batch audit. This method can reduce the computation amount and effectively improve the efficiency of querying invalid files. After the location of the failed file is determined, the data of the failed file will be effectively recovered by RS erasure code. In this way, the user only needs to recover the invalid data according to RS erasure code. This solution effectively guarantees the security of the data stored by group users on the cloud and improves the solution for users to store and share data in the cloud.

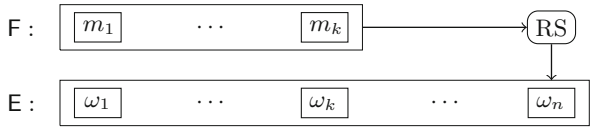

Fig. 2. Data encode.

4.2 Data Encoding of Group Users

As shown in Fig. 2, the file F is divided into $F = m_1, m_2, \cdots, m_k$, and then the data blocks $m_1, m_2, \cdots, m_k$ are coded according to RS erasure code. First of all, the group administrator selects a set of vectors $x_1, x_2, \cdots, x_k$ that are linearly independent and publishes them in the group. Then, F is encoded as $E = \omega_1, \omega_2, \cdots, \omega_n$, where $n = red + k$ and red is the redundancy of RS erasure code. Notably $E = x \times F^{\mathrm{T}}$, where x is a matrix of k linearly independent vectors of length n, x_i is in the finite domain 2^w and w must be large enough that $n \leq 2^w + 1$. The calculation is provided as follow:

$$E = x \times F^{\mathrm{T}} = \begin{bmatrix} vm_{1,1} & vm_{1,2} & \cdots & vm_{1,k} \\ & & \ddots & \\ vm_{k,1} & vm_{k,2} & \cdots & vm_{k,k} \\ vm_{k+1,1} & vm_{k+1,2} & \cdots & vm_{k+1,k} \\ & & \ddots & \\ vm_{n,1} & vm_{n,2} & \cdots & vm_{n,k} \end{bmatrix} \times \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_k \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_k \\ \vdots \\ \omega_n \end{bmatrix}$$

4.3 Description of CIRD Scheme

In previous cloud storage audit schemes [8,15,19,21], the efficiency of user data integrity audit is mainly addressed. However, this article mainly deals with the situation when the data stored by group users on the cloud server is corrupted. That is, the damaged files can be quickly located and effectively recovered. In addition, for the overall perfection, efficiency and security of the scheme, Schnorr signature [21] is selected in this paper to generate the label key of user data, so as to have an efficient audit for dynamic group users. Our scheme continues to use UI and ssk as the group identity corresponding to $SSig$ and secret key [21]. Further more, we continue to adopt the feature of ID-based digital signature $SSig$ and use it to ensure the integrity of file identifier names, user retract numbers and verification values. As applied in the scenario, RN is the key to update the data label key. Finally, the dedicated description of our scheme is described as follows.

1. Setup *algorithm*: In this algorithm, the $\mathcal{PKG}$ generates its own public keys, secret keys and some system parameters.
   - The $\mathcal{PKG}$ chooses the master secret key $\gamma \in Z_q^*$ and master partial key $\beta \in Z_q^*$. Then, the $\mathcal{PKG}$ computes two public values $Y_1 = g^{\gamma}$ and $Y_2 = g^{\beta}$.

The $\mathcal{PKG}$ holds the master secret key itself for generating the group user's identity key and sends the master partial key to group manager for generating the partial key.
- The $\mathcal{PKG}$ arbitrarily selects and publishes *tuple* as the system public parameters, where g and u are arbitrarily selected from G_1 by $\mathcal{PKG}$, e is a bilinear mapping $e : G_1 \times G_1 \rightarrow G_2$, q is the prime order of a group and *tuple*$= \{G_1, G_2, e, q, g, u, Y_1, Y_2, H_1, H_2, h\}$.

2. KeyGen *algorithm*: The well-known Schnorr signature [21] is utilized to compute the identity key and the partial key. This signature enables dynamic group users to audit efficiently. In this algorithm the $\mathcal{PKG}$ generates the identity key, the group manager generates the partial key and group users generate their private keys using the identity key and the partial key. The calculation of this key are described as follows:
   - The $\mathcal{PKG}$ runs $\{UI, r_{UI}\} \rightarrow \{UIK_{UI}\}$. When $\mathcal{PKG}$ receives the group's identity UI, it chooses a random number $r_{UI} \in Z_q^*$, and computes $R_{UI} = g^{r_{UI}}$ and $\sigma_{UI} = r_{UI} + \gamma H_1(UI, R_{UI}) \bmod q$. Then, the $\mathcal{PKG}$ sends the identity key $UIK_{UI} = (R_{UI}, \sigma_{UI})$ to group users.
   - The group users run $\{UIK_{UI}\} \rightarrow \{0, 1\}$. After receiving the identity key, the group users verify the correctness of the UIK_{UI}. If $g^{\sigma_{UI}} = R_{UI} \cdot Y_1^{H_1(UI, R_{UI})}$, the group users accept the identity key UIK_{UI}; otherwise they refuse it.
   - The group manager runs $\{RN, r_{RN}, UI\} \rightarrow \{TK_{UI,RN}\}$. First, we default the group to start with no one leaving. The group manager sets the number of user revocations $RN = 0$, then sends it to group users and cloud. Later, the group manager chooses a random number $r_{RN} \in Z_q^*$, and computes $R_{RN} = g^{r_{RN}}$ and $\sigma_{RN} = r_{RN} + \beta H_2(UI, R_{RN}, RN) \bmod q$. The group manager sends the partial key $TK_{UI,RN} = (R_{RN}, \sigma_{RN})$ to the group users, where RN is initially set to 0, and when one user leaves the group, $RN = RN + 1$. As long as any user leaves the group, the partial key $TK_{UI,RN}$ is updated as RN changes.
   - The group users run $\{TK_{UI,RN}\} \rightarrow \{0, 1\}$. After receiving the partial key, the group users verify the correctness of the $TK_{UI,RN}$. If $g^{\sigma_{RN}} = R_{RN} \cdot Y_2^{H_2(UI, R_{RN}, RN)}$, the group users accept the partial key $TK_{UI,RN}$; otherwise, refuse it.
   - The group users run $\{UIK_{UI}, TK_{UI,RN}\} \rightarrow \{SK_{UI,RN}\}$. Then, they calculate the private keys based on the identity key $UIK_{UI} = (R_{UI}, \sigma_{UI})$ and the partial key $TK_{UI,RN} = (R_{RN}, \sigma_{RN})$(Here $RN = 0$). Finally, $\sigma = (\sigma_{UI} + \sigma_{RN}) \bmod q$ is computed and the private keys $SK_{UI,RN}$ turns to be (R_{UI}, R_{RN}, σ).

3. TagGen *algorithm*: This algorithm is run by the group user. As shown in Fig. 2, the user first encodes the file F as E. Then, the user computes $t_{i_j} = (h(FID \| i_j \| RN) \cdot u^{\omega_{i_j}})^{\sigma}$, $i \in [1, N], j \in [1, n]$, where FID is the file identifier, i is the ith user, j is the data block index, u is the random value and $u \in G_1$, RN is the number of user revocations. The group user computes the file tag $Ftag = FID \| RN \| R_{UI} \| R_{RN} \| \tau$, where $\tau = SSig_{ssk}(FID \| RN \| R_{UI} \| R_{RN})$,

ssk is a file tag private key. Notably, calculation of the key is referenced from [15].

4. Challenge *algorithm*: In this algorithm, the TPA generates an auditing challenge for the cloud. The calculation of this chal are described as follows:
   - Randomly pick a set I with $N \cdot s$ elements, where $s \in [1, n]$, N represents the number of the audit users and $I = \{1_1, \cdots, 1_s, \cdots, N_1, \cdots, N_s\}$.
   - Generate a random value $v_{i_s} \in Z_q^*$ for each $i_s \in I, i \in [1, N]$.
   - Send the auditing challenge $\text{chal} = \{C_1, C_2, \cdots, C_N\}$ to the CSP, where $C_i = \{i_j, v_{i_j}\}, i \in [1, N], j \in [1, s]$.
5. Prove *algorithm*: In this algorithm, after receiving the chal of TPA, the CSP generates a corresponding proof to demonstrate its ownership of the intact cloud data. The calculation of this proof are described as follows:
   - Compute $T = \{T_1, \cdots, T_N\}$, where $T_i = \prod_{j=1}^{s} t_{i_j}^{v_{i_j}}$. $\mu = \{\mu_1, \cdots, \mu_N\}, i \in [1, N]$ and $\mu_i = \sum_{j=1}^{s} v_{i_j} \cdot \omega_{i_j}$. Then computes P where $P = (T, \mu)$.
   - Send P along with the file tag to the TPA as the proof.

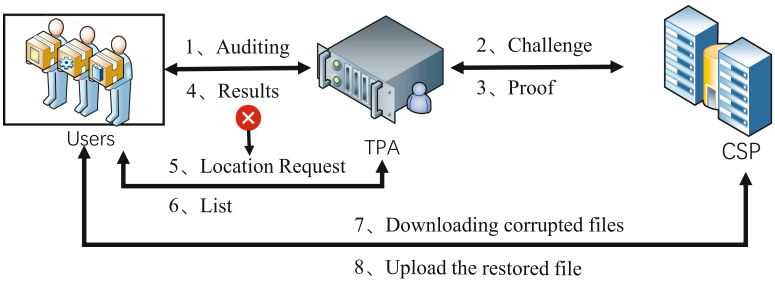

Fig. 3. Procedure of failed file identification and recovery scheme

6. Recover *algorithm*:
   As shown in Fig. 3, there may be two situations when the TPA verifies the integrity of the data: data integrity verification passes/fails. As mentioned earlier, the failure of the integrity verification means that the files stored by the user in the CSP are damaged and no longer available. In view of this situation, the CIRG scheme designed in this paper can efficiently locate the damaged file and restore it.
   The TPA first retrieves the file tag $Ftag$, and verifies the validity of the file tag by checking whether τ is a valid signature via UI. If it is, the TPA receives the data possession evidence set P returned by the server and verifies whether the server has correctly stored the client's files in the form of batch processing. The TPA runs $\{P, V\} \rightarrow \{0, 1\}$. As shown in Fig. 4, the TPA verifies the correctness of the proof P by checking whether the following equation holds:

$$\prod_{i=1}^{i=N} e(T_i, g) \stackrel{?}{=} \prod_{i=1}^{i=N} e(\prod_{j=1}^{s} h(FID \| i_j \| RN)^{v_{i_j}} \cdot u^{\mu_i}, \mathsf{v}), \tag{1}$$

where $\mathsf{v} = R_{UI} \cdot R_{RN} \cdot Y_1^{H_1(UI, R_{UI})} \cdot Y_2^{H_2(UI, R_{RN}, RN)}$.

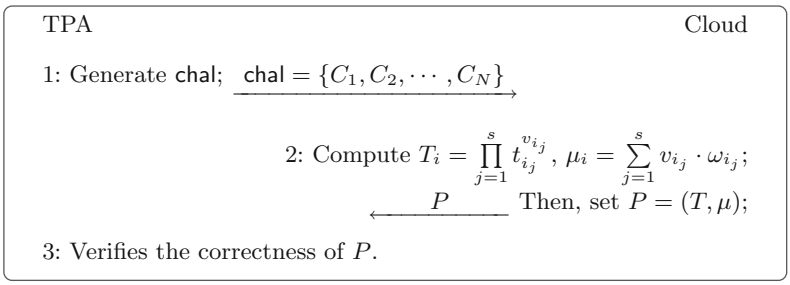

Fig. 4. The protocol of data integrity verification.

If this equation holds, it indicates that all files are intact, and returns 1; otherwise, it indicates that at least one invalid file exists, and returns 0. Next, we elaborate on how to locate corrupted data.

After inputing the challenged file information, the TPA uses the query algorithm to identify all the invalid files and outputs the corresponding list. First, the calculation of RDI is given. We denoted the audit result of Eq. (1) as "Le = Re". Then, set RDI$(\cdot)$ and RDI$(\cdot) \rightarrow (\mathsf{L}e - \mathsf{R}e \stackrel{?}{=} 0)$. Thus this RDI converts audit formula (1) to a value 0 judgment: $IFN_{\mathsf{RDI}} = \mathsf{L}e - \mathsf{R}e$. The detail of this calculation is shown as follows:

$$IFN_{\mathsf{RDI}} \stackrel{?}{=} e(T_i, g) - e(\theta_i, \mathsf{v}), \tag{2}$$

where $\theta_i = \prod_{j=1}^{s} h(FID \| i_j \| RN)^{v_{i_j}} \cdot u^{\mu_i}$, v is the same as Eq. (1).

We associate Le with Re in terms of moving positions. Then, the audit results of multiple user files are associated with the added method. And the intermediate value of the audit results is used to reduce the calculation and improve the efficiency of the query damage file. For the files of N users, the RDI batch audit formula is:

$$IFN_{\mathsf{RDI}}^{[1,N]} \stackrel{?}{=} \sum_{i=1}^{N} e(T_i, g) - \sum_{i=1}^{N} e(\theta_i, \mathsf{v}), \tag{3}$$

where θ_i and v are the same as above.

If $IFN_{\mathsf{RDI}}^{[1,N]} = 0$, the batch audit is passed, indicating that the files are complete; If $IFN_{\mathsf{RDI}}^{[1,N]} \neq 0$, there is at least one invalid file. According to the above formula, we set $X_{[1,N]}$ to represent the file audit of N users. As shown

in Fig. 5, the validation value of the parent node in the binary search tree is equal to the sum of the validation values of the left and right child nodes. In the following formula, by comparing the verification values of the parent node and the left child node, the validity of the right child node can be directly determined without performing batch auditing.

$$IFN_{\mathsf{RDI}}^{[N/2+1,N]} = IFN_{\mathsf{RDI}}^{[1,N]} - IFN_{\mathsf{RDI}}^{[1,N/2]} \tag{4}$$

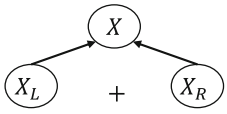

Fig. 5. Schematic diagram of the relational dichotomy

Next, we present a general algorithm for querying multiple corrupted files. The algorithm provides an N-bit string $(b_1, b_2, \ldots, b_N)$ as output. With overwhelming probability $b_i = 1$ if and only if file i is valid. The details of this algorithm are shown as follows:

(a) First, compute $IFN_{\mathsf{RDI}}^{[1,N]}$, if $IFN_{\mathsf{RDI}}^{[1,N]} = 0$, output N bits $(1, 1, \ldots, 1)$ and stop; otherwise, the TPA does the following steps.
(b) Assuming $X_{[1,N]}$ is the root node, calculate the left child node $X_{L=N/2}$, set $IFN_{\mathsf{RDI}}^{[1,N/2]}$.
(c) Calculate the verification value for the right child. If $IFN_{\mathsf{RDI}}^{[1,N/2]} = 0$, set $IFN_{\mathsf{RDI}}^{[N/2+1,N]} = IFN_{\mathsf{RDI}}^{[1,N]}$; If $IFN_{\mathsf{RDI}}^{[1,N/2]} = IFN_{\mathsf{RDI}}^{[1,N]}$, set $IFN_{\mathsf{RDI}}^{[N/2+1,N]} = 0$; Otherwise compute $IFN_{\mathsf{RDI}}^{[N/2+1,N]} = IFN_{\mathsf{RDI}}^{[1,N]} - IFN_{\mathsf{RDI}}^{[1,N/2]}$.
(d) Apply the same algorithm recursively to left sub-tree and right sub-tree. Finally find all the damage files and output $(b_1, b_2, \ldots, b_N)$.

According to the above analysis, the RDI method based on value 0 eliminates the batch audit process of the right child node, thus reduces the number of batch audits. With this method, the location of all the invalid files can be found. Similarly, we can then apply RDI again to locate the specific failed data block of the failed file.

Next, the corrupted data recovery method is instantiated. Notably that for multiple corrupted data recovery cases only the instantiation method needs to be extended. For the sake of description, damaged file E_1 is taken as an example to explain how to restore it. We get k valid data blocks $E_1' = \omega_{1_1}', \omega_{1_2}', \cdots, \omega_{1_k}'$ from E_1 and the sub-matrix B from x (Details in Sect. 4.2) in terms of E_1'. This B and E_1' correspond one to one. The data recovery is carried out according to the formula: $B^{-1} * E_1' = F_1^{\mathrm{T}}$, where B^{-1} is the inverse of the k by k matrix consisting of the B. Thus, the original data F_1 can be restored. The details of this calculation are shown as follows:

$$B = \begin{bmatrix} b_{1,1}\ b_{1,2} \cdots b_{1,k} \\ b_{2,1}\ b_{2,2} \cdots b_{2,k} \\ \qquad \ddots \\ b_{k,1}\ b_{k,2} \cdots b_{k,k} \end{bmatrix} \quad B^{-1} = \begin{bmatrix} b'_{1,1}\ b'_{1,2} \cdots b'_{1,k} \\ b'_{2,1}\ b'_{2,2} \cdots b'_{2,k} \\ \qquad \ddots \\ b'_{k,1}\ b'_{k,2} \cdots b'_{k,k} \end{bmatrix} \quad F_1^{\mathrm{T}} = B^{-1} \times \begin{bmatrix} \omega'_{1_1} \\ \omega'_{1_2} \\ \vdots \\ \omega'_{1_k} \end{bmatrix}$$

4.4 Dynamic Group Users and Data Recovery

The CIRG scheme in this paper supports dynamic group users environment, that is, the case of user cancellation. When a user leaves the group, the label key in the group needs to be updated, thus the value of RN is increased by 1. Meanwhile the partial key $TK_{UI,RN}$ of the user is updated [21], so that the label key in the group is updated. Once a user leaves the group, his data will be managed by legitimate members of the group. CIRG supports dynamic group users environments, which adopts a unified algorithm to encode data. That is, the data uploaded to the cloud server by a group of users adopts this data encoding algorithm. Once the user has not uploaded all the data before leaving the group, the data he shares in the group will be corrupted. At this point, the legitimate user being charged of managing his data can still recover the failed shared data through the encoding algorithm exposed by the group administrator.

5 Security Analysis

Theorem 1 (The Correctness and Security of Batch Audit). *If the group users, group manager, CSP and TPA can perform calculations correctly, the proposed CIRG scheme is both correct and secure in the batch audit of shared files among multiple users.*

Proof. Since this article adopts the label key feature, all users in the group use this key. Therefore, using batch auditing can simultaneously audit different files of multiple users. In order to prove the security of batch audit, consider the following batch auditing. According to the formula of integrity audit, we set the number of stored files of the user as 1 for the convenience of description. The following uses the data integrity audit of users A, B, and C as an example. First, if RN is equal, these three user label keys are the same, so they can be audited simultaneously. The details of this formula are shown as follows:

$$\begin{aligned} \text{left} = \prod_{i=1}^{i=3} e(T_i, g) &= \prod_{i=1}^{i=3} e(\prod_{j=1}^{j=s} (h(FID \| i_j \| RN) \cdot u^{\omega_{i_j}})^{\sigma \cdot v_{i_j}}, g) \\ &= \prod_{i=1}^{i=3} e(\prod_{j=1}^{j=s} (h(FID \| i_j \| RN)^{v_{i_j}} \cdot u^{\mu_i}, g^{\sigma_{UI} + \sigma_{RN}}) \\ &= \prod_{i=1}^{i=3} e(\prod_{j=1}^{j=s} (h(FID \| i_j \| RN)^{v_{i_j}} \cdot u^{\mu_i}, \mathrm{v}) \end{aligned} \tag{5}$$

$$\text{right} = \prod_{i=1}^{i=3} e(\prod_{j=1}^{j=s} h(FID\|i_j\|RN)^{v_{i_j}} \cdot u^{\mu_i}, \mathsf{v}) \tag{6}$$

where $\mathsf{v} = R_{UI} \cdot R_{RN} \cdot Y_1^{H_1(UI,R_{UI})} \cdot Y_2^{H_2(UI,R_{RN},RN)}$.

If a malicious CSP stored incorrect or invalid user data, it must have a private key (σ in equation (5)) to ensure that the left is equal to the right in order to pass the audit. Since a malicious CSP cannot obtain the private key, it cannot generate the correct storage proof, let alone forge the correct data signature. In other words, if the above equation left is equal to right, if and only if the files of A, B and C respectively pass the audit, so only if the CSP has to store the correct data to pass the audit. Obviously, if CIRG is executed correctly, the left is equal to the right,which ensures that batch auditing is correct and secure. On the contrary, the label key will be different while the three RN users are not equal. Therefore, A, B and C perform data integrity audit, respectively. The same as the situation when RN is equal.

Theorem 2 (The Security and efficiency of Data Recovery). *In our scheme, CIRG can effectively recover original data when the group user's data stored on the cloud server is corrupted.*

Proof. The possibility of data recovery in this paper depends on the relationship between the number of damaged data blocks dd and the redundancy red. When $dd \leq red$, damaged data can be recovered. First, the sub-matrix $B_{k\times k}$ of the x-matrix is correctly calculated, according to k uncorrupted data $E_1' = \omega_{1_1}', \omega_{1_2}', \cdots, \omega_{1_k}'$, where E_1' has k valid blocks of data for this file and B is the linearly independent vector corresponding to E_1'. And then, since x is invertible, the inverse of the B-matrix can be correctly figured out. Therefore, the data recovery formula is: $B^{-1} * E_1' = F_1^{\mathrm{T}}$. Finally, the original data F_1 can be restored. When $dd > red$, we can obtain less than k valid data E_1' and calculate the corresponding sub-matrix B. Obviously, the number of columns of B is not equal to the number of rows of E_1', which does not satisfy the mathematical calculation of the matrix. Therefore, the data cannot be recovered.

Next, the efficiency of corrupt data recovery is briefly discussed. This article focuses on the implementation of functions and does not consider the optimal data recovery technique. Therefore, RS erasure code technology, which is easily described in [22], is selected as the coding tool in our scheme. Variety of techniques to improve the computing efficiency of erasure codes were mentioned in [22], such as optimized bit matrix design and optimized computing scheduling, reducing common XOR operations, caching management techniques and vectorization techniques. These techniques can also be similarly applied to our scheme to improve coding efficiency.

6 Conclusion

In this paper, we propose an audit scheme that supports fast location and recovery of corrupted data in the environment of dynamic group users. With our

solution, group users can audit the integrity of data stored on the cloud server. Once the data uploading is corrupted, users can efficiently identify the corrupted files and recover them. Finally, the designed scheme is theoretically proved to be secure under the defined threat model.

Acknowledgment. This work was supported by National Natural Science Foundation of China (No. 61702402), Doctoral Research Funds for Northwest A&F University (No. Z1090121092), Fundamental Research Funds for the Central Universities (No. XJS211502), and Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS201716).

References

1. Ateniese, G., et al.: Provable data possession at untrusted stores. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, 28–31 October 2007, pp. 598–609. ACM (2007)
2. Bernstein, D.J., Doumen, J., Lange, T., Oosterwijk, J.-J.: Faster batch forgery identification. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 454–473. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34931-7_26
3. Gumaste, P.: Amazon AWS outage (2019). https://www.whizlabs.com/blog/amazon-aws-outage/
4. Guo, H., Zhang, Z., Xu, J., An, N., Lan, X.: Accountable proxy re-encryption for secure data sharing. IEEE Trans. Dependable Secur. Comput. **18**(1), 145–159 (2021)
5. Huo, H., Jiang, T., Tan, S., Tao, X.: Efficient public integrity auditing with secure deduplication in cloud computing. Int. J. Embed. Syst. **11**(6), 764–777 (2019). https://doi.org/10.1504/IJES.2019.103995
6. Jiang, T., Chen, X., Ma, J.: Public integrity auditing for shared dynamic cloud data with group user revocation. IEEE Trans. Computers **65**(8), 2363–2373 (2016)
7. Juels, A., Jr., B.S.K.: PORs: proofs of retrievability for large files. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, 28–31 October 2007, pp. 584–597. ACM (2007)
8. Li, Y., Yu, Y., Min, G., Susilo, W., Ni, J., Choo, K.R.: Fuzzy identity-based data integrity auditing for reliable cloud storage systems. IEEE Trans. Dependable Secur. Comput. **16**(1), 72–83 (2019)
9. Matt, B.J.: Identification of multiple invalid signatures in pairing-based batched signatures. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 337–356. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00468-1_19
10. Pan, Y., Hu, N.: Research on dependability of cloud computing systems. In: 2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS), pp. 435–439 (2014)
11. Ren, Z., Wang, L., Wang, Q., Xu, M.: Dynamic proofs of retrievability for coded cloud storage systems. IEEE Trans. Serv. Comput. **11**(4), 685–698 (2018)
12. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_7

13. Shen, J., Zhou, T., He, D., Zhang, Y., Sun, X., Xiang, Y.: Block design-based key agreement for group data sharing in cloud computing. IEEE Trans. Dependable Secur. Comput. **16**(6), 996–1010 (2019)
14. Shin, S., Kim, S., Kwon, T.: Identification of corrupted cloud storage in batch auditing. In: Khalil, I., Neuhold, E., Tjoa, A.M., Da Xu, L., You, I. (eds.) ICT-EurAsia 2015. LNCS, vol. 9357, pp. 221–225. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24315-3_22
15. Wang, C., Chow, S.S.M., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for secure cloud storage. IEEE Trans. Comput. **62**(2), 362–375 (2013)
16. Wang, H.F., Li, Z.H., Zhang, X., Sun, J., Zhao, X.N.: Batch auditing supporting fast searching invalid files in cloud storage. Chin. J. Comput. **40**(418), 144–157 (2017)
17. Weikai, W., Lihong, R., Lei, C., Yongsheng, D.: Intrusion detection and security calculation in industrial cloud storage based on an improved dynamic immune algorithm. Inf. Sci. **501**, 543–557 (2019)
18. Wu, D., Xia, Y., Sun, X.S., Huang, X.S., Dzinamarira, S., Ng, T.S.E.: Masking failures from application performance in data center networks with shareable backup. In: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, 20–25 August 2018, pp. 176–190. ACM (2018)
19. Yu, J., Ren, K., Wang, C., Varadharajan, V.: Enabling cloud storage auditing with key-exposure resistance. IEEE Trans. Inf. Forensics Secur. **10**(6), 1167–1179 (2015)
20. Zafar, F., et al.: A survey of cloud computing data integrity schemes: design challenges, taxonomy and future trends. Comput. Secur. **65**, 29–49 (2017)
21. Zhang, Y., Yu, J., Hao, R., Wang, C., Ren, K.: Enabling efficient user revocation in identity-based cloud storage auditing for shared big data. IEEE Trans. Dependable Secur. Comput. **17**(3), 608–619 (2020)
22. Zhou, T., Tian, C.: Fast erasure coding for data storage: a comprehensive study of the acceleration techniques. In: Merchant, A., Weatherspoon, H. (eds.) 17th USENIX Conference on File and Storage Technologies, FAST 2019, Boston, MA, 25–28 February 2019, pp. 317–329. USENIX Association (2019)