A collusion-resistant certificateless provable data possession scheme for shared data with user revocation

Guang Yang
Hangzhou Normal University
Lidong Han (ldhan@hznu.edu.cn)
Hangzhou Normal University
Jingguo Bi
Beijing University of Posts and Telecommunications
Fuqun Wang
Hangzhou Normal University

Research Article

Keywords: Provable data possession, user revocation, cloud storage, shared data integrity, collusion resistance, certificateless

Posted Date: July 7th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1800460/v1

A collusion-resistant certificateless provable data possession scheme for shared data with user revocation

Guang Yang[1,2], Lidong Han[1,2*†], Jingguo Bi[3] and Fuqun Wang[1,2]

[1*]Key Laboratory of Cryptography of Zhejiang Province, Hangzhou Normal University, NO.2318, Yuhangtang Road, Yuhang District, Hangzhou, 311121, Zhejiang Province, P.R.China.

[2]School of Information Science and Technology, Hangzhou Normal University, NO.2318, Yuhangtang Road, Yuhang District, Hangzhou, 311121, Zhejiang Province, P.R.China.

[3]School of Cyberspace Security, Beijing University of Posts and Telecommunications, No.10 Xitucheng Road, Haidian District, BeiJing, 100876, Beijing, P.R.China.

*Corresponding author(s). E-mail(s): ldhan@hznu.edu.cn;
Contributing authors: yangguang@stu.hznu.edu.cn; jguobi@bupt.edu.cn; fqwang@hznu.edu.cn;
†These authors contributed equally to this work.

Abstract

Cloud storage service can provide a lot convenience for users to collect, store and share data within a group. However, there are some secure issues, one of which is that the cloud server may cheat users for a good reputation when the data is lost. A classical solution to deal with it is provable data possession (PDP) protocols. Most of PDP protocols are relied on either the public key infrastructure or identity-based cryptography. However, certificate management and key escrow issues place a significant burden on this. Furthermore, revoking the group's illegal users is a critical issue for PDP schemes. To address these problems, we put forward a certificateless PDP scheme for shared data with user revocation. Our proposed scheme not only achieves efficient user revocation but also withstands collusion attack. We also give a formal proof of security of our proposed scheme without random oracle model. Experimental results and analysis show that our scheme is quite effective in data auditing, verification, and user revocation.

Keywords: Provable data possession, user revocation, cloud storage, shared data integrity, collusion resistance, certificateless

1 Introduction

Cloud storage services have grown increasingly popular in recent years. Users can manage their data more efficiently with cloud storage, but it has raised some security issues. One of them is that the user data in the cloud is frequently out of their physical control. Due to technological failure or other human factors, the cloud server may lose the user data, so users are unsure whether their data is unchangeably stored in cloud. The cloud service provider (CSP) may even argue that the data is still preserved for goodwill and interests.

As a result, it is required to determine whether user data is completely saved on the cloud server.

To check the integrity of data, lots of methods such as digital signatures have been proposed. It requires users to download their data from the cloud and check it locally. But this consumes many resources and the verification of data integrity is very low. In 2007, provable data possession (PDP) scheme [2] was proposed for the first time. A third-party auditor (TPA) organization which communicates on behalf of users is introduced to conduct the verification in order to save users computing resources. In PDP, each data block has a unique authentication tag. The TPA determines if the data is entirely saved in the cloud by evaluating the accuracy of these authentication tags. On the other hand, TPA is not always an utterly trustworthy organization, and it may try to obtain sensitive information about users during the audit process. Because users may keep personal data such as business contracts and medical records on the cloud, data privacy is another security issue [23]. As a result, it is essential to ensure that data privacy is not compromised in data auditing [14].

Many efficient PDP schemes have been put forward since [2]. However, these schemes only pay attention to the integrity verification for personal data, without considering the shared data. Wang et al. [18] first proposed a scheme for data integrity in a group. In the group, each user can update the data shared within the group and is able to generate authenticators for modified data. Through these new authenticators, the verifier may ensure the integrity of the updated data. Their scheme adopts group signatures to conduct the data integrity verification in the group, but it ignores the issue of adding and removing users from the group. Wang et al. [19] utilizes proxy re-signatures to offer the function of user revocation and assigns the task of generating the authenticators to the server. This technique considerably reduces the computational costs for users. However, it does not resist collusion attack. The revoked user can get the private key of legal users with colluding the cloud server, which compromises the scheme.

Recently, some PDP schemes for group data integrity verification were proposed [7, 13, 16, 27]. However, these schemes rely on the mechanism of traditional public key infrastructure (PKI) or identity-based cryptography (IBC). Although PKI is extensively used, there are some problems, such as a large burden of certificate management. IBC eliminates these problems, but it requires a fully trusted key generation center (KGC), which generates all private keys of users. However, a malicious KGC is is an immediate threat to IBC based PDP scheme. When the security of KGC is compromised, the private keys of the users are disclosed. Recently, certificateless public key cryptography (CL-PKC) [1] is proposed to solve the above issues. Several certificateless-based PDP schemes with user revocation in group are proposed [5, 9, 24, 28]. However, their schemes are vulnerable to the collusion attack. Therefore, it is a challenging open problem to design a secure provable data possession scheme that provides group data integrity verification and user revocation under CL-PKC.

1.1 Contributions

To our knowledge, for the first time, we define and solve the problem of certificateless public integrity auditing that supports data privacy, group data sharing and group user revocation simultaneously without random oracle model. Our major contributions are as follows.

- For the first time, we explore the problem of constructing the PDP protocol for shared data based on certificateless public key cryptography. Our new scheme can not only avoid complex certificate management and key escrow issues, but also achieve public auditing.
- Furthermore, our scheme can provide efficient user revocation and collu-sion resistance. That is to say, the revoked user is not able to get the legal users' privacy and data information by colluding with the cloud server.
- We provide an analysis on security and efficiency of our scheme. Our new scheme is proven secure without random oracle model. From comparison with previous schemes, we show our scheme perform well in security and computation.

1.2 Related work

In 2007, Ateniese et al. [2] proposed the first PDP scheme, which employs a probabilistic algorithm to validate data integrity. Even if users do not download all the data and they may be confident that the data in cloud is not lost or modified

with great probability. Their scheme, however, does not allow users to alert their data dynamically. Ateniese et al. [3] presented an improved scheme to allow dynamic operation by employing symmetric encryption. Subsequently, Erway et al. [4] presented an expanded PDP model which employs the Rank-based Authenticated Skip List (RASL) structure to provide a complete dynamic data operation. Later, to facilitate dynamic data operations, Index Hash Table (IHT) [29], Merkle Hash Table (MHT) [23] and Dynamic Hash Table (DHT) [15] are introduced.

Zhu et al. proposed a type of attack by a malicious TPA, and the user data is disclosed during the auditing processes [30]. [22] protects data from being leaked by a random number, which is chosen by CSP and unknown to TPA. Therefore, it can achieve data privacy. In another way, Yu et al. [26] adopted the idea of zero-knowledge proof to achieve data privacy. [10] proposed an identity-based protocol to prevent TPA from obtaining data. Ji et al. introduced an improved version of [10] by increasing the flexibility and optimizing the efficiency [8]. Recently, many researchers proposed some PDP schemes to support data privacy [11, 14, 17, 25].

Most of the above schemes focused on data integrity detection only for individuals, not for group. Wang et al.[18] presented a scheme to check the data integrity in group by using group signatures. The broadcast encryption and the ring signatures are used to validate the data integrity within a group [12, 21]. None of these solutions, however, allows for user revocation. The authors in [19] utilized proxy re-encryption to realize user revocation, but it is vulnerable to collusion attack. The revoked user can recover the private keys of legal users by colluding with the cloud server. [16] introduced dynamic hash table and lazy revocation to provide user revocation and shared data dynamic operations. According to the lazy revocation model, the cryptographic information remains untouched when a user is revoked. If the related data is updated again by other legal users, the authenticators need to be re-computed, which is too slow in authenticators updating. Zhang et al. [27] introduced key updates to support user revocation, and they can resist collusion attack. Luo et al. [13] utilized threshold sharing to prevent revoked user from colluding with the cloud server. However, all previous schemes above are based on PKI or IBC, which require complicated certificate administration or inherent key escrow problem.

To fill this security gap, Al-Riyami et al. [1] presented certificateless public key cryptography. For the first time, Literature [20] put forward a certificateless PDP scheme, however, he et al. [6] ponited out that the protocol in [20] could not withstand the attack from Type I adversary. The aforementioned schemes discussed the personal data auditing without considering shared data in a group. In 2012, Wang et al. [18] presented a privacy-preserving scheme to ensure the shared data integrity. After that they constructed a new protocol with proxy re-signatures to support user revocation [19]. Later, a series of schemes for group data based on CL-PKC are presented such as [5, 9, 24, 28]. However, these schemes can not solve the specific problem of collusion attack.

1.3 Organizations

The rest of paper is organized as follow. The preliminaries are introduced in Section 2. Section 3 describes the security model of our proposed scheme. The concrete scheme and the security proof are shown in Section 4 and 5. Section 6 describes the theoretical analysis and experiments. We give a conclusion in Section 7.

2 Preliminaries

2.1 Bilinear

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ are both cyclic groups of the same large prime order q, g be a generator of $\mathbb{G}_1$. Then a bilinear map can be represented by $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following conditions:

- **Computational**: for $\forall g_1, g_2 \in \mathbb{G}_1$, $e(g_1, g_2)$ can be computed efficiently.
- **Bilinear** : for $\forall g_1, g_2 \in \mathbb{G}_1, \forall a, b \in Z_q^*$, it has $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
- **Non-degenerate** : $\exists g_1, g_2 \in \mathbb{G}_1$, $e(g_1, g_2) \neq 1_{G_2}$.

2.2 Complexity Assumption

In this section we discuss the mathematical intractable problems. Assume $\mathbb{G}_1, \mathbb{G}_2$ are two multiplicative cyclic groups and g is a generator of $\mathbb{G}_1$.

Definition 1 (Computational Diffie-Hellman (CDH) problem). Given a tuple $T = (g, g^x, g^y)$ without the knowledge of $x, y \in_R Z_p^*$, where $g, g^x, g^y \in \mathbb{G}_1$. Then compute $g^{xy} \in \mathbb{G}_1$.

Definition 2 (Bilinear strong Diffie-Hellman (BSDH) problem). Given a tuple $T = (k, g, g^x, g^{x^2}, \cdots, g^{x^q})$ and a known k, $k, x \in_R Z_p^*$, where $g, g^x, g^{x^2}, \cdots, g^{x^q} \in \mathbb{G}_1$. Then compute $e(g,g)^{\frac{1}{k+x}} \in \mathbb{G}_2$.

Definition 3 (Extended bilinear strong Diffie-Hellman (EBSDH) problem). Given a tuple $T = (k, g, g^x, g^{x^2}, \cdots, g^{x^q})$ with and a known k, $k, x \in_R Z_p^*$, where $g, g^x, g^{x^2}, \cdots, g^{x^q} \in \mathbb{G}_1$. Then compute $e(g,g)^{\frac{k}{k+x}} \in \mathbb{G}_2$.

2.3 System model

Figure 1 shows the system model in our scheme, which includes four entities: user group, TPA , KGC and CSP. There are two kinds of users in a group: group members who can store and update the data within the group and the group manager (GM) who can allow group member to join the group and revoke a user when he leaves from group. In system setup, group manager initializes the original group. TPA with sufficient resources is in charge of auditing data integrity on behalf of group members. CSP has abundant storage and computational resources to provide group members with data storage service and generates the proof of data integrity as response to TPA. We assume that CSP and TPA are both semi-trusted. Specially, TPA and CSP can perform the algorithm honestly, but they want to obtain the information of data or forge the proof of inaccurate data.

2.4 Framework of our scheme

Here, our scheme contains the following algorithms:

- **Setup**: Given the security parameter λ, KGC generates the system public parameters *params*, the master private key *msk* and public key.

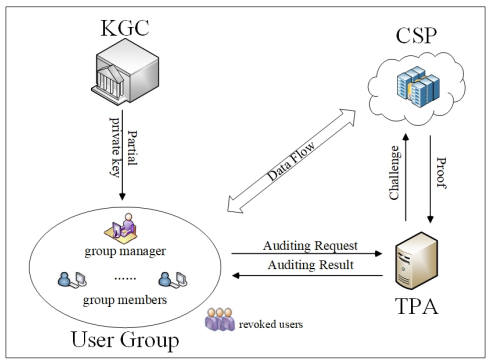

Fig. 1 System model

- **Group Manager KeyGen**: The group manager generates his private key sk_{gm} and public key pk_{gm}.
- **PartialKeyGen**: Given the identity ID_i of user, KGC generates the partial private key D_i and sends it to the user in a secure channel. The user can verifies the validity of D_i.
- **GroupKeyGen**: Inputting the number of revoked users Num and group manager private key, GM executes this algorithm to generate the user group private key gsk_i.
- **PrivateKeyGen**: User generates the secret value firstly. After receiving the group private key and partial private key, user executes the algorithm to compute the complete private key sk_i of a user by the secret value.
- **PublicKeyGen**: Given the secret value, the group private key and partial private key, the user generates his public key pk_i.
- **AuthenticatorGen**: Given the data block m_j and his complete private key sk_i, the user computes authenticator σ_j of the m_j.
- **Challenge**: TPA picks some data blocks as the challenged blocks with some random numbers, and then outputs the *chal* as the challenge requests.
- **ProofGen**: After receiving the challenge *chal*, CSP generates the proof P with the challenged blocks and corresponding authenticators.
- **ProofVerify**: Given the challenge *chal*, the proof P and the public key pk_i, TPA executes this algorithm to output the verifying results. The algorithm outputs 0 or 1.
- **Revoke**: Given the updated Num, GM outputs a new group key according to the new Num. GM sends the new group private key to legal users secretly.

3 Security Model

In our proposed scheme, there are two types of adversaries namely Type-I adversary $\mathcal{A}_1$ and Type-II adversary $\mathcal{A}_2$. $\mathcal{A}_1$ aims to forge the authenticators of data blocks with the ability to replace the public key of any user, but he can not get the master private key. $\mathcal{A}_2$ can access the master private key but is not able to replace the public key of user. Next, we introduce two games between a challenger $\mathcal{C}$ and the adversary $\mathcal{A}_1$ or $\mathcal{A}_2$. The details of two games are as follows.

Game 1. The game runs between $\mathcal{C}$ and $\mathcal{A}_1$.

Setup: The algorithm **Setup** is executed by $\mathcal{C}$ to get the system public parameters and the master private key. The master private key is held by $\mathcal{C}$ in secret.

Queries: Adversary $\mathcal{A}_1$ can make a series queries to $\mathcal{C}$ as follows.

- PartialKey-Query (ID_i): $\mathcal{A}_1$ picks the identity ID_i and queries $\mathcal{C}$ to get the partial private key of ID_i.
- GroupKey-Query (Num): $\mathcal{A}_1$ picks the number of revoked users Num and queries $\mathcal{C}$ to get the group private key of ID_i.
- PrivateKey-Query (ID_i): $\mathcal{A}_1$ picks the identity ID_i and queries $\mathcal{C}$ to get the private key of ID_i.
- PublicKey-Query (ID_i): $\mathcal{A}_1$ picks the identity ID_i and queries $\mathcal{C}$ to get the public key of ID_i.
- PublicKey-Replace (ID_i, PK_i): $\mathcal{A}_1$ is able to replace the public key of ID_i.
- Authenticator-Query (ID_i, m): $\mathcal{A}_1$ picks the tuple (ID_i, m) and gives it to $\mathcal{C}$ to get the authenticator σ of the data block m by ID_i.

Forge: At last, $\mathcal{A}_1$ produces a forged authenticator σ' for the data block m' with the public key $PK_{ID'}$ and the challenged identity ID'. If all conditions in follows are fulfilled, $\mathcal{A}_1$ wins the game.

1. $\mathcal{A}_1$ does not issue a private key query with ID'.
2. $\mathcal{A}_1$ does not issue a partial private key query with ID'. Replacing ID''s public key is prohibited.
3. $\mathcal{A}_1$ doesn't issue an authenticator query with (ID', m').
4. The forged authenticator σ' of m' is valid for ID' and $PK_{ID'}$.

Game 2. The game runs between $\mathcal{C}$ and $\mathcal{A}_2$.

Table 1 Notations

Notation	Description
p	A large prime
$\mathbb{G}_1$	Cyclic multiplicative group with order p
$\mathbb{G}_2$	Cyclic multiplicative group with order p
g_1, g_2	Two generators of $\mathbb{G}_1$
H_1, H_4	$\{0,1\}^* \to \mathbb{G}_1$
H_2	$\{0,1\}^* \times \mathbb{G}_1 \times \{0,1\}^* \to Z_p^*$
H_3	$\{0,1\}^* \to Z_p^*$
e	A bilinear pairing
ID_i	The identity of user u_i
Num	The number of revoked users
s	The master private key msk_i
x_{gm}	The group manager private key sk_{gm}
D_i	The partial private key for u_i
gsk_i	The group private key of u_i
sk_i	The private key of u_i
F	The shared file identifier
$chal$	The challenge information
m_j	The jth data block
T_j	The authenticator of m_j
P	The proof message

Setup: The algorithm **Setup** is executed by $\mathcal{C}$ to get the master key and the system public parameters. Then $\mathcal{C}$ sends both of them to $\mathcal{A}_2$.

Queries: An adversary $\mathcal{A}_2$ can make a series queries to $\mathcal{C}$ as follows.

- PrivateKey-Query (ID_i), GroupKey-Query (Num), PublicKey-Query (ID_i) and Authenticator-Query(ID_i, m): which are defined in Game 1.

Forge: Finally, $\mathcal{A}_2$ produces a forged authenticator σ' for the data block m' with the challenged identity ID'. If the three conditions in follows are satisfied, $\mathcal{A}_2$ wins the game.

1. $\mathcal{A}_2$ does not issue a private key query with ID'.
2. $\mathcal{A}_2$ doesn't issue an authenticator query with (ID', m').
3. The forged authenticator σ' of m' is valid for ID'.

Definition 4 If $\mathcal{A}_1$ and $\mathcal{A}_2$ can both win the Game 1 and Game 2 with a negligible probability in probabilistic polynomial time respectively, the authenticators of each block are unforgeable.

4 Our new scheme

In this section, we give a concrete description of our proposed scheme. Suppose that there are z

group members in a group. ID_i is the identity of one group member u_i for $1 \leq i \leq z$. We divide the file F into n blocks $\{m_j\}_{1 \leq j \leq n}$, where $m_j \in Z_p^*$. To ensure that the value of auxiliary data is not tampered with, we utilize a secure identity-based digital signature $SSig$ to check the integrity of auxiliary data, where ssk and spk are the private key and public key. The details of our scheme are demonstrated as follows:

Setup: Given a security parameter λ, KGC randomly picks a large prime p and two multiplicative cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with order p. $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear map and g_1, g_2 are both the generator of $\mathbb{G}_1$. KGC selects a random number $s \in Z_p^*$ as the master private key msk and $mpk = g_1^s$. After that, KGC computes $h = e(g_1, g_1)^s$. There are four secure hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \{0,1\}^* \times \mathbb{G}_1 \times \{0,1\}^* \to Z_p^*$, $H_3 : \{0,1\}^* \to Z_p^*$ and $H_4 : \{0,1\}^* \to G_1$. KGC keeps the master private key secretly and publishes public parameters as $params = \{p, g_1, g_2, h, \mathbb{G}_1, \mathbb{G}_2, e, H_1, H_2, H_3, H_4, mpk\}$.

Group Manger KeyGen: The group manager randomly chooses $x_{gm} \in Z_p^*$, and sets group private key sk_{gm} as x_{gm} and computes $pk_{gm} = g_1^{x_{gm}}$. Finally, the group manager initializes the number of the revoked users $Num = 0$.

PartialKeyGen: On receiving $params$ and the identity ID_i of u_i, KGC computes $h_i = H_1(ID_i)$ and

$$y_i = g_1^{\frac{s \cdot h_i}{s + h_i + r_i}}$$

where r_i is a random number from Z_p^*. KGC sets $D_i = (y_i, R_i = g^{r_i})$ as partial private key and delivers it to u_i by a secure channel. u_i can check D_i by the following equation.

$$e(g_1, mpk)^{h_i} = e(y_i, g_1^{h_i} \cdot R_i \cdot mpk)$$

If the equation holds, u_i accepts D_i as his partial private key; otherwise, he refuses it.

GroupKeyGen: On receiving the identity ID_i of u_i, the group manager randomly picks $r_{Num} \in Z_p^*$ and computes $R_{Num} = g_1^{r_{Num}}$ and $x_i = r_{Num} + x_{gm} H_2(ID_i, R_{Num}, Num) \bmod p$. The group manager sends $gsk_i = (R_{Num}, x_i)$ to u_i. After receiving gsk_i, u_i can verify it by checking whether

$$g_1^{x_i} = R_{Num} \cdot pk_{gm}^{H_2(ID_i, R_{Num}, Num)}.$$

If it holds, u_i accepts gsk_i as his group private key; otherwise, he refuses it.

PrivateKeyGen: The group member u_i picks a random number k_i from Z_p^* secretly. On receiving $D_i = (y_i, R_i)$ and $gsk_i = (R_{Num}, x_i)$, u_i generates $\alpha_i = k_i \cdot x_i$ and sets his private key sk_i as (R_i, α_i).

PublicKeyGen: After receiving D_i and gsk_i, u_i computes the public key

$$pk_i = (Y_1 = y_i^{x_i}, Y_2 = h^{x_i}, Y_3 = y_i^{\frac{1}{k_i}}).$$

AuthenticatorGen: Each group member generates the corresponding authenticators of data blocks with his private key. If u_i intends to generate the authenticator of data block m_j, he computes the authenticator

$$\sigma_j = [(g_1^{h_i} \cdot R_i \cdot mpk)^{m_j} \cdot g_2^{H_3(\omega_j)}]^{\alpha_i}$$

where $h_i = H_1(ID_i)$ and auxiliary information $\omega_j = FnamejNum$. $Fname$ is the identity of F, j is the block index and Num is the number of revoked users. After that, u_i generates the file tag $Tag = FnameNumpk_i \ Sig_{ssk}(FnameNumpk_i)$. At last, u_i sends $F = (m_1, \cdots, m_n)$, $T = (\sigma_1, \cdots, \sigma_n)$ and Tag to the cloud. Cloud server executes the following operations to verify the validity of Tag and T. The cloud server extracts Num in Tag and checks whether it is the newest. Then The cloud server proceeds to the next two steps if they hold. Otherwise, he rejects it and the user is deemed to be revoked.

- The cloud server verifies the integrity of Tag using the public key spk of $SSig$. If yes, it continues to next step; otherwise, rejects it.
- The cloud checks the correctness of T by the equality

  $$e(\sigma_j, Y_3) = e(g_2, Y_1)^{H_3(\omega_j)} \cdot Y_2^{H_1(ID_i) m_j}$$

  where $1 \leq j \leq n$. If the equation holds, the cloud accepts it; otherwise, he refuses it.

Challenge: TPA creates a random challange $chal = \{(i, v_i)\}, 1 \leq i \leq c$, where $c \in [1, n]$ is the number of challenged blocks, i is the index of each challenged block and $v_i \in Z_p^*$ is random. Afterwards TPA transmits $chal$ to the cloud.

ProofGen: In this algorithm, the cloud server first randomly selects $t \in Z_p^*$, and computes $\mu = h^t, \pi^* = \sum_{i=0}^{c} v_i m_i, \pi = \pi^* + t \cdot H_4(\mu)$ and $\Theta = \prod_i^c \sigma_i^{v_i}$. Finally, the cloud server sends the proof $P = \{\mu, \pi, \Theta\}$ and Tag to TPA.

ProofVerify: On receiving P and Tag, TPA first verifies the validity of Tag. If it is valid, TPA parses $Fname, Num$ and pk_i from Tag. Then TPA performs integrity verification by judging whether:

$$e(\Theta, Y_3) \cdot \mu^{H_4(\mu)} = e(g_2, Y_1)^{\sum v_i H_3(FnamejNum)} \cdot Y_2^{H_1(ID_i)\pi}$$

If yes, TPA accepts the proof; otherwise, he refuses it.

Revoke: The algorithm will be executed by GM and non-revoked group members. When revoking a group member, GM updates the number of revoked users to $Num = Num + 1$ and broadcasts it to legal users and the cloud server. Later, he generates the new group private key based on the new Num. Specially, GM picks a random $r_{Num} \in Z_q^*$, and calculates $R_{Num} = g_1^{r_{Num}}$ and $x_i = r_{Num} + x_{gm} H_2(ID, R_{Num}, Num)$ mod q. GM transmits the new group private key $gsk_i = (R_{Num}, x_i)$ to u_i. The non-revoked group member u_i can check the correctness by judging whether

$$g_1^{x_i} = R_{Num} \cdot pk_{gm}^{H_2(ID_i, R_{Num}, Num)}$$

. If yes, u_i accepts gsk_i as his group private key; otherwise, he refuses it.

Note that the revoked group members are unable to calculate a valid private key without the new group private key. Therefore, the cloud server rejects any requests when a revoked user wants to store or update data in cloud. Furthermore, once a valid user receives the new group private key, he is required to compute his new public/private key pairs. The subsequent authenticators are generated by new public/private key pairs. So, it is efficient to verify the updated blocks

5 Security analysis

5.1 Correctness

Theorem 1 *TPA can audit data blocks correctly if the user, GM, CSP and TPA follow the proposed procedure.*

Proof: The equation in the $ProofVerify$ holds as follows.

$$\begin{aligned}
&e(\Theta, Y_3) \cdot \mu \\
&= e(\prod_i^c \sigma_i^{v_i}, y_i^{\frac{1}{c_i}}) \cdot e(g_1, g_1)^{st} \\
&= \prod_i^c e(g_1^{(h_i + r_i + s)m_i} \cdot g_2^{H_3(FnameiNum)}, y_i^{\frac{1}{c_i}})^{v_i \alpha} \cdot h^t \\
&= \prod_i^c e(g_1^{(h_i + r_i + s)m_i} \cdot g_2^{H_3(FnameiNum)}, y_i)^{x_i v_i} \cdot h^t \\
&= \prod_i^c e(g_1, g_1^{h_i})^{x_i y v_i m_i} \cdot e(g_2^{H_3(FnameiNum)}, y_i)^{x_i v_i} \cdot h^t \\
&= e(g_1, g_1)^{x_i h_i y \sum_i^c v_i m_i} \cdot e(g_2, y_i^{x_i})^{\sum_i^c v_i H_3(FnameiNum)} \cdot h^t \\
&= (Y_2)^{h_i \pi} \cdot e(g_2, Y_1)^{\sum_i^c v_i H_3(FnameiNum)} \cdot h^t
\end{aligned}$$

□

5.2 Detectability

Theorem 2 *If CSP stores n data blocks, deletes or modifies m bad blocks, and TPA picks c challenged blocks, the probability that at least one of m blocks is detected is larger than* $1 - (\frac{n-m}{n})^c$.

Proof: Denote P_X as the probability that at least one bad block is identified and X as the number of damaged blocks which are audited.

$$\begin{aligned}
P_X &= P\{X \geq 1\} = 1 - P\{X = 0\} \\
&= 1 - \frac{n-m}{n} \cdot \frac{n-1-m}{n-1} \cdots \cdot \frac{n-c+1-m}{n-c+1}.
\end{aligned}$$

So we get $P_X \geq 1 - (\frac{n-m}{n})^c$. □

5.3 User revocation

Theorem 3 *The revoked users can not comprise the legal users with colluding with the cloud server.*

Proof: A revoked user can comprise the legal users without the cloud server if and only if both of the following conditions hold.

- He has the newest Num.
- He can generate valid authenticators.

However, it is infeasible for a revoked user to meet the two requirements mentioned above. For the

first one, GM updates the Num and broadcasts it to the cloud server and legal users when a user is revoked. The revoked user can not know the newest Num. Secondly, GM generates the new group private key for each non-revoked user. The revoked user can not generate the valid authenticators with no knowledge of new group private key. Even if a revoked user utilizes the prior private key to generate authenticator and sends it to the cloud, the authenticator can not pass the verification executed by a cloud server using the equation

$$e(\sigma_j, Y_3) = e(g_2, Y_1)^{H_3(FnamejNum)} \cdot Y_2^{H_1(ID_i)m_j}.$$

On the other hand, if a revoked user colludes with the cloud server, he could know the newest Num from the cloud server. However, the cloud server does not know the new group private key and the revoked user can not generate the valid private key without new group private key. The user private key is not leaked. Therefore, our proposed scheme **collusion-resistant**. □

5.4 Data privacy

Theorem 4 *In the proposed scheme, TPA can not obtain the original user blocks.*

Proof: Firstly, we show that TPA can not obtain information from π. Note that $\pi = \pi^* + t \cdot H_4(\mu)$, and the data blocks of user are included by π^*. However, cloud server blinds π^* by $t \cdot H_4(\mu)$, where t is a random number from Z_p^* and $\mu = h^t$. Even if TPA knows μ, solving t is equivalent to solve CBDH problem. Hence, it is hard to obtain the privacy of block π^* from π for TPA.

Secondly, we show that π^* can not be obtained from Θ, where

$$\Theta = \prod_j^c \sigma_j^{v_j} = \prod_j^c [(g_1^{h_i} \cdot R_i \cdot mpk)^{m_j} \cdot g_2^{H_3(\omega_j)}]^{v_j \alpha_i}$$
$$= \prod_j^c [(g_1^{h_i} \cdot R_i \cdot mpk)^{m_j v_j \alpha_i}] \cdot g_2^{H_3(\omega_j) v_j \alpha_i}$$

We can analyze that $(g_1^{h_i} \cdot R_i \cdot mpk)^{m_j v_j \alpha_i}$ is blinded by $g_2^{H_3(\omega_j) v_j \alpha_i}$. However, TPA only knows $g_2^{H_3(\omega_j) v_j}$ without knowledge of private key α_i. Therefore, TPA can not get the value of $g_2^{H_3(\omega_j) v_j}$.

Finally, we illustrate that TPA can not obtain m_j from the proof $P = (\mu, \pi, \Theta)$. We regard π^* as a private key and $H_4(\mu)$ as a challenge value. We consider the data auditing between TPA and CSP as a provably secure honest zero knowledge. It implies that no information about π^* will be leaked. Thus, data privacy of user is guaranteed. This completes our proof. □

5.5 Unforgeability of authenticators

Theorem 5 *The proposed scheme is unforgeable if $\mathcal{A}_1$ and $\mathcal{A}_2$ both win the game with a negligible probability.*

The theorem is correct if the following two lemmas are proved.

Lemma 1 For any adversary $\mathcal{A}_1$, if he wins the Game 1 in polynominal time t with a nonnegligible probability ϵ after making n_{pri} PrivateKeyGen queries, n_{ag} AuthenticatorGen queries, n_p PublicKeyGen queries and n_c Create-User queries, then there is a challenger $\mathcal{C}_1$ who has the ability to solve the q-EBSDH problem with the probability ϵ' in polynomial time t' for

$$\epsilon' \geq (1 + \frac{n_c + n_{pri} + n_{ag}}{n})(\frac{1}{n_c})\epsilon$$
$$t' = t + O((n_c(q+3) + 3n_p + 4n_{ag})T_E)$$

where T_E is exponentiation cost.

Proof: Given $\phi = (\mathbb{G}, g, g^x, g^{x^2}, \cdots, g^{x^q})$, the adversary $\mathcal{A}_1$ wins the Game 1 means that the challenger $\mathcal{C}_1$ can compute $e(g,g)^{\frac{x}{x+\theta}}$ for known $\theta \in Z_p^*$ at non-negligible probability. The multiplicative group $\mathbb{G}$, its generator g, g_2, and the maximum number of queries q are included by the following proof. Assume $A_i = g^{x^i}, \forall i \in [1, q]$.

1. Setup: $\mathcal{A}_1$ sets two lists $\mathcal{L}_1 = (ID, Y_1, Y_2, Y_3, x, k, \alpha, h, y, R)$ and $\mathcal{L}_2 = (ID, x, k, \alpha, Y_1, Y_2, Y_3)$ firstly after receiving ϕ. $\mathcal{L}_1$ and $\mathcal{L}_2$ are empty initially. $\mathcal{A}_1$ selects three polynomials $P(x), \Phi(x)$ and $Q(x)$ of degree $q-1$ as

$$P(x) = \sum_{i=0}^{q-1} a_i x^i, \Phi(x) = \sum_{i=0}^{q-1} b_i x^i, Q(x) = \sum_{i=0}^{q-1} c_i x^i$$

where $(a_i, b_i, c_i) \in_R (Z_p^*)^3$. It computes $g_1 = \prod_{i=0}^{q-1}(A_i)^{a_i} = g^{P(x)}$ and $mpk = \prod_{i=0}^{q-1}(A_{i+1})^{a_i} = g_1^x$. Finally, it computes $h =$

$e(g_1, mpk) = e(g_1, g_1)^x$ and sends system public parameter $params = (\mathbb{G}_1, \mathbb{G}_2, q, e, g_1, g_2, h, mpk, H_1, H_2, H_3, H_4)$ to $\mathcal{A}_1$, where $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \{0,1\}^* \times \mathbb{G}_1 \times \{0,1\}^* \to Z_q^*$, $H_3 : \{0,1\}^* \to Z_q^*, H_4 : \{0,1\}^* \to G_1$. To solve q-EBSDH, $\mathcal{A}_1$ executes an algorithm $\mathcal{C}_1$ as follows.

2. Create-User-Query: $\mathcal{A}_1$ chooses $r_i = \Phi(ID_i), r_i' = Q(ID_i)$ and computes $R_i = g^{r_i}, R_i' = g^{r_i'}$. Since the identity of user is public, we denote $P(x)$ as

$$P(x) = \prod_{j=0}^{q-1}(x + r_j + h_j).$$

where $h_i = H_1(ID_i)$ and $(\mu_1, \cdots, \mu_q) \in_R (Z_p^*)^q$. Denote $P_i(x)$ as the polynomial for ID_i.

$$\begin{aligned} P_i(x) &= \frac{x \cdot P(x)}{x + r_i + h_i} + \mu_0 \\ &= \frac{x \cdot \prod_{j=0}^{q-1}(x + r_j + h_j)}{x + r_i + h_i} + \mu_0 \\ &= x \cdot \prod_{j=0, j\neq i}^{q-1}(x + r_j + h_j) + \mu_0 \\ &= \sum_{j=0}^{q-1} \mu_j x^j \end{aligned}$$

$\mathcal{C}_1$ sets $y_i = \{\frac{\prod_{i=0}^{q-1}(A_i)^{\mu_i}}{g^{\mu_0}}\}^{h_i} = g^{\frac{x \cdot P(x) \cdot h_i}{x + r_i + h_i}} = g_1^{\frac{x \cdot h_i}{x + r_i + h_i}}$, $\forall ID_i$. Then, it randomly chooses $x_1 \in Z_p^*$ and computes $x_i = r_i' + x_1 H_2(ID_i, Num), pk_{gm} = g^{x_1}$. Finally, it stores $(ID_i, \perp, \perp, \perp, \perp, r_i, h_i, y_i, R_i, x_i, Num)$ in $\mathcal{L}_1$. It is noted that partial private key and group private key can checked by $e(g_1, mpk)^{h_i} = e(y_i, g_1^{h_i \cdot r_i} \cdot mpk)$ and $g^{x_i} = R_i' \cdot pk_{gm}^{H_2(ID_i, Num)}$ respectively.

3. Group-Key-Query: $\mathcal{A}_1$ picks ID_i and performs the query about it. If ID_i is included by $\mathcal{L}_1$, $\mathcal{C}_1$ returns x_i and Num. Otherwise, $\mathcal{C}_1$ makes Create-User query for $ID_i \neq ID^*$ and responds x_i and Num.
4. Partial-Key-Query: $\mathcal{A}_1$ picks ID_i and performs the query about it. If ID_i is included by $\mathcal{L}_1$, $\mathcal{C}_1$ returns $D_i = (y_i, R_i)$. Otherwise, $\mathcal{C}_1$ makes Create-User query for $ID_i \neq ID^*$ and responds $D_i = (y_i, R_i)$.
5. Private-Key-Query: For $ID_i = ID^*$, $\mathcal{C}_1$ aborts. If not, it checks whether $ID_i (\neq ID^*)$ is included by $\mathcal{L}_1$. If $\mathcal{L}_1$ contains ID_i (other than $\perp$), then $\mathcal{C}_1$ returns (α_i, R_i); otherwise, it chooses $k_i \in_R Z_p^*$ and computes $\alpha_i' = k_i x_i$. Now, if $\alpha_i = \perp$, then $\mathcal{C}_1$ updates only α_i as $\alpha_i = \alpha_i'$; otherwise, he makes Create-User query and sets $\alpha_i = \alpha_i'$ in $\mathcal{L}_1$.
6. Public-Key-Query: $\mathcal{A}_1$ picks ID_i and performs the query about it. If ID_i is included by $\mathcal{L}_1$, then $\mathcal{C}_1$ returns $Y_i = (Y_1, Y_2, Y_3)$. Otherwise, $\mathcal{C}_1$ makes Create-User query for $ID_i \neq ID^*$ and the details are illustrated as follows.
   - Check whether ID_i is included by $\mathcal{L}_1$ where $k_i = \perp$. If k_i does not exist, $\mathcal{C}_1$ picks $k_i \in_R Z_p^*$. Then, it calculates $Y_i = (Y_{i1}, Y_{i2}, Y_{i3})$, where $Y_{i1} = y_i^{x_i}$, $Y_{i2} = h^{x_i}$ and $Y_{i3} = y_i^{k_i}$.
   - It replaces the tuple $(\perp, \perp, \perp, \perp, \perp)$ by $(Y_{i1}, Y_{i2}, Y_{i_3}, \alpha_i, k_i)$ to $\mathcal{L}_1$. At last, $\mathcal{C}_1$ transmits public key $Y_i = (Y_{i1}, Y_{i2}, Y_{i3})$ to $\mathcal{A}_1$.
7. Replace-Public-Key: Now, for invoked query $(ID_i, Y_i' = (Y_{i1}', Y_{i2}', Y_{i3}'))$, $\mathcal{C}_1$ sets $Y_{i1} = Y_{i1}', Y_{i2} = Y_{i2}', Y_{i3} = Y_{i3}'$ and $k_i = k_i', \alpha_i = \alpha_i'$. If the corresponding tuple is included by $\mathcal{L}_1$, $\mathcal{C}_1$ updates these values. Finally, $\mathcal{C}_1$ inserts $(ID_i, k_i, \alpha_i, Y_{i1}, Y_{i2})$ to $\mathcal{L}_1$.
8. Authenticator-Query: When the query $q = (ID_i, m_j, \omega_j)$ is received, $\mathcal{C}_1$ first checks whether q is included by $\mathcal{L}_1$. If it does not exist, $\mathcal{C}_1$ generates an authenticator according to the original scheme. Otherwise, $\mathcal{C}_1$ takes list $\mathcal{L}_2$ into account and simulates as follows.
   - Retrieve the private key α_i from list $\mathcal{L}_1$.
   - Compute $\sigma_j = [(g_1^{h_i} \cdot R_i \cdot mpk)^{m_j} \cdot g_2^{H_3(\omega_j)}]^{\alpha_i}$ for $h_i = H_1(ID_i)$ and responds σ_j to $\mathcal{A}_1$.
9. Forge: $\mathcal{A}_1$ stops making queries and forges a σ' for data block m' with $Y_{ID'} = (Y_{ID'1}, Y_{ID'2}, Y_{ID'3})$, where $Y_{ID'}$ is the public key for ID'. If $ID' \neq ID^*$, $\mathcal{C}_1$ aborts. Otherwise, it calculates $\psi(x) = \sum_{i=0}^{q-2} \tau_i x^{i+1}$ for some coefficients $(\tau_1, \tau_2, \cdots, \tau_{q-1}) \in (Z_p^*)^{q-1}$ and reforms $P(x)$ as

$$P(x) = x^{-1} \cdot \psi(x) \cdot [(r_i + h_i) + s] + d$$

where $d \in Z_p^*$. Then, it finds $Y_{ID'3}$ from $\mathcal{L}_1$ where $Y_{ID'3} = g_1^{\frac{x h_i}{[r_i + h_i + x] k_i}}$. After that, $\mathcal{C}_1$

computes $\mathcal{T}$ as

$$\begin{aligned}\mathcal{T} &= [(Y_{ID'3})^{\frac{k_i}{h_i}} \cdot \prod_{i=1}^{q-1}(A_{i+1})^{-\tau_i}]^{\frac{1}{d}} \\ &= [g^{\frac{xP(x)}{r_i+h_i+x}} \cdot g^{-\psi(x)}]^{\frac{1}{d}} \\ &= [g^{\psi(x)+\frac{dx}{r_i+h_i+x}} \cdot g^{-\psi(x)}]^{\frac{1}{d}} \\ &= [g^{\frac{dx}{r_i+h_i+x}}]^{\frac{1}{d}} = g^{\frac{x}{r_i+h_i+x}}\end{aligned}$$

$\mathcal{C}_1$ computes $Z = e(g,\mathcal{T}) = e(g,g)^{\frac{x}{(r_i+h_i)+x}}$. If $a = r_i+h_i$, the value of Z is equal to $e(g,g)^{\frac{x}{a+x}}$. So, $\mathcal{C}_1$ can solve the q-EBSDH problem.

Probability analysis: $\mathcal{A}_1$ successfully forges a valid signature if and only if three events occur concurrently as follows.

Γ_1: $\mathcal{C}_1$ does not abort during the above simulation.

Γ_2: σ' is a valid forged authenticator on m' for ID^*.

Γ_3: The forged authenticator σ' allows $ID = ID^*$.

Hence, we can write

$$Pr[\Gamma_1 \wedge \Gamma_2 \wedge \Gamma_3] = Pr[\Gamma_1] \cdot Pr[\Gamma_2|\Gamma_1] \cdot Pr[\Gamma_3|\Gamma_1 \wedge \Gamma_2].$$

Now, we describe a probabilistic analysis of each event.

- The probability of Γ_1 is equivalent to the probability that the simulations of Partial-Key-Query, Private-Key-Query, Authenticator-Query succeed simultaneously. Let us denote the number of iterations as n and the challenged identity ID^* is one of n identities.
  - The probability of failure in Create-User-Query is $\frac{1}{n}$. Therefore, the probability that the simulation of Partial-Key-Query succeeds after n_c queries is $(1-\frac{1}{n})^{n_c}$.
  - Similarly, the probability that simulation of Private-Key-Query succeeds is $(1-\frac{1}{n})^{n_{pri}}$ since there are n_{pri} Private-Key queries.
  - When $ID \neq ID^*$, the probability that Authenticator-Query successfully produces an authenticator is $(1-\frac{1}{n})$. Hence, after n_{ag} queries the probability for succeessful simulation of Authenticators-Query is $(1-\frac{1}{n})^{n_{ag}}$.
- The probability of $\mathcal{A}_1$ forges a valid authenticator is ϵ.
- $Pr[\Gamma_3|\Gamma_1 \wedge \Gamma_2]$ means that the probability $\mathcal{A}_1$ generates a valid forged authenticator for $ID = ID^*$ by conditioning on the events Γ_1 and Γ_2, which is $\frac{1}{n_c}$ since there are n_c queries for Creat-User-Query.

Therefore

$$\begin{aligned}Pr[\Gamma_1] &= (1-\frac{1}{n})^{n_c+n_{pri}+n_{ag}} \\ &\geq (1-\frac{n_c+n_{pri}+n_{ag}}{n}) \\ Pr[\Gamma_2|\Gamma_1] &\geq \epsilon \\ Pr[\Gamma_3|\Gamma_1 \wedge \Gamma_2] &\geq \frac{1}{n_c}.\end{aligned}$$

Therefore, $\mathcal{A}_1$ can break q-EBSDH problem with the probability

$$\begin{aligned}\epsilon' &= Pr[\Gamma_1 \wedge \Gamma_2 \wedge \Gamma_3] \\ &= Pr[\Gamma_1] \cdot Pr[\Gamma_2|\Gamma_1] \cdot Pr[\Gamma_3|\Gamma_1 \wedge \Gamma_2] \\ &\geq (1-\frac{n_c+n_{pri}+n_{ag}}{n})\frac{\epsilon}{n_c}\end{aligned}$$

As ϵ is a negligible value, $\mathcal{C}_1$ can not break the q-EBSDH problem. Hence, our scheme is resistant to Type-I attack.

Estimated time: We focus on exponentiation cost T_E because it has significantly higher cost than other operations in our simulation than the others. $\mathcal{C}_1$ requires $n_c(q+3)T_E, 3n_pT_E$, and $4n_sT_E$ during *partial-private-key*, *public-key* and *Authenticator* queries, respectively. So, the time cost is $t^* = (n_c(q+3)+3n_p+4n_{ag})T_E$. Therefore, $t' = t + O((n_c(q+3)+3n_p+4n_{ag})T_E)$ is the total time to break q-EBSDH problem by $\mathcal{C}_1$. □

Lemma 2 For any adversary $\mathcal{A}_2$, if he wins the Game 2 in polynominal time t with a nonnegligible probability ϵ after making n_{pri} PrivateKeyGen queries, n_{ag} AuthenticatorGen queries, n_p PublicKeyGen queries and n_c Create-User queries, then there is a challenger $\mathcal{C}_2$ who has ability to solve the q-BSDH problem with the probability ϵ' in polynomial time t' for

$$\epsilon' \geq (1+\frac{n_{pri}+n_{ag}}{n})(\frac{1}{n_c})\epsilon$$

$$t' = t + O((n_c(q+3)+3n_p+4n_{ag})T_E)$$

where T_E is exponentiation cost.

Proof: Given $\phi = (\mathbb{G}, g, g^x, g^{x^2}, \cdots, g^{x^q})$, the adversary $\mathcal{A}_2$ wins the Game 2 means that the challenger $\mathcal{C}_2$ can calculate $e(g,g)^{\frac{1}{x+\theta}}$ for known $\theta \in Z_p^*$ at non-negligible probability. The multiplicative group $\mathbb{G}$, its generator g, g_2, and the maximum number of queries q are the same as in Lemma 1. Let $A_i = g^{x^i}, \forall i \in [1,q]$.

1. Setup: $\mathcal{A}_2$ sets two lists $\mathcal{L}_1 = (ID, Y_1, Y_2, Y_3, x, k, \alpha, h, y, R)$ and $\mathcal{L}_2 = (ID, x, k, \alpha, Y_1, Y_2, Y_3)$ firstly after receiving ϕ. $\mathcal{L}_1$ and $\mathcal{L}_2$ are empty initially. $\mathcal{A}_2$ selects three polynomials $P(x)$, $\Phi(x)$ and $Q(x)$ of degree $q-1$ as

$$P(x) = \sum_{i=0}^{q-1} a_i x^i, \Phi(x) = \sum_{i=0}^{q-1} b_i x^i, Q(x) = \sum_{i=0}^{q-1} c_i x^i$$

where $\forall i \in [0, q-1]$ and $(a_i, b_i, c_i) \in_R (Z_p^*)^3$. It calculates $g_1 = \prod_{i=0}^{q-1}(A_i)^{a_i} = g^{P(x)}$ and $mpk = g_1^s$. Finally, it computes $h = e(g_1, g_1)^s$ and sends system parameter $params = (\mathbb{G}_1, \mathbb{G}_2, q, e, g_1, g_2, h, mpk, H_1, H_2, H_3, H_4)$ to $\mathcal{A}_2$, where $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \{0,1\}^* \times \mathbb{G}_1 \times \{0,1\}^* \to Z_q^*$, $H_3 : \{0,1\}^* \to Z_q^*$, $H_4 : \{0,1\}^* \to G_1$. Now, $\mathcal{A}_2$ executes an algorithm $\mathcal{C}_2$ to break q-BSDH problem.
2. Create-User-Query(ID_i): Firstly $\mathcal{A}_2$ picks $r_i' = Q(ID_i)$ and computes $R_i' = g^{r_i'}$. If $ID_i \neq ID^*$, $\mathcal{A}_2$ selects $r_i = \Phi(ID_i)$ and calculates $R_i = g^{r_i}$ and $y_i = (g_1)^{\frac{sh_i}{h_i + r_i + s}}$. Otherwise, because the identity of user is public, $P(x)$ can be extended as

$$P(x) = \prod_{j=0}^{q-1}(x + s + h_j).$$

where $h_i = H_1(ID_i)$ and $(\mu_1, \mu_2, \cdots, \mu_{q-2}) \in_R (Z_p^*)^{q-2}$. Denote $P_i(x)$ as the polynomial for ID_i.

$$\begin{aligned} P_i(x) &= \frac{P(x)}{x + s + h_i} \\ &= \frac{\prod_{j=0}^{q-1}(x + s + h_j)}{x + s + h_i} \\ &= \prod_{j=0, j\neq i}^{q-1}(x + s + h_j) \\ &= \sum_{j=0}^{q-2} \mu_j x^j \end{aligned}$$

It sets $y_i = \{\prod_{i=0}^{q-2}(A_i)^{\mu_i}\}^{s \cdot h_i} = \{g_1^{\frac{1}{x+s+h_i}}\}^{s \cdot h_i}$, $\forall ID_i$. Then, $\mathcal{C}_2$ randomly chooses $x_1 \in Z_p^*$ and computes $x_i = r_i' + x_1 H_2(ID_i, Num), pk_{gm} = g^{x_1}$. Finally, it stores $(ID_i, \perp, \perp, \perp, \perp, r_i, h_i, y_i, R_i, x_i, Num)$ in $\mathcal{L}_1$. It is noted that partial private key and group private key can checked by $e(g_1, mpk)^{h_i} = e(y_i, g_1^{h_i \cdot r_i} \cdot mpk)$ and $g^{x_i} = R_i' \cdot pk_{gm}^{H_2(ID_i, Num)}$ respectively.
3. Group-key-Query: $\mathcal{A}_2$ performs the query about ID_i and if it is included by $\mathcal{L}_1$, $\mathcal{C}_2$ responds x_i and Num. Otherwise, $\mathcal{C}_2$ makes Create-User query for $ID_i \neq ID^*$ and outputs x_i and Num.
4. Partial-Key-Query: $\mathcal{A}_2$ performs the query about the selected ID_i. If it is included by $\mathcal{L}_1$, $\mathcal{C}_2$ responds $D_i = (y_i, R_i)$. Otherwise, $\mathcal{C}_2$ makes Create-User query for $ID_i \neq ID^*$ and outputs $D_i = (y_i, R_i)$.
5. Private-Key-Query: For $ID_i = ID^*$, $\mathcal{C}_2$ aborts. Otherwise, it checks whether $ID_i (\neq ID^*)$ is included by $\mathcal{L}_1$. If it exists, $\mathcal{C}_2$ responds (α_i, R_i); otherwise, it chooses $k_i \in_R Z_p^*$ and computes $\alpha_i' = k_i x_i$. If $\alpha_i = \perp$, $\mathcal{C}_2$ updates $\alpha_i = \alpha_i'$; otherwise, makes Create-User query and sets $\alpha_i = \alpha_i'$ in $\mathcal{L}_1$.
6. Public-Key-Query: $\mathcal{A}_2$ performs the query about its selected ID_i. If ID_i is included by $\mathcal{L}_1$, $\mathcal{C}_2$ returns $Y_i = (Y_1, Y_2, Y_3)$. Otherwise, $\mathcal{C}_2$ makes Create-User query for $ID_i \neq ID^*$ and the details are illustrated as follows.
   - Check that whether ID_i is included by $\mathcal{L}_1$. If k_i does not exist, $\mathcal{C}_2$ picks $k_i \in_R Z_p^*$. Then, it calculates $Y_i = (Y_{i1}, Y_{i2}, Y_{i3})$, where $Y_{i1} = y_i^{x_i}$, $Y_{i2} = h^{x_i}$ and $Y_{i3} = y_y^{k_i}$.
   - Then, it replaces $(\perp, \perp, \perp, \perp, \perp)$ by $(Y_{i1}, Y_{i2}, Y_{i_3}, \alpha_i, k_i)$ to $\mathcal{L}_1$. At last, $\mathcal{C}_2$ transmits public key $Y_i = (Y_{i1}, Y_{i2}, Y_{i3})$ to $\mathcal{A}_2$.
7. Authenticator-Query: When the query $q = (ID_i, m_j, \omega_j)$ is received, $\mathcal{C}_2$ first checks whether q is included by $\mathcal{L}_1$. If it does not exist, $\mathcal{C}_2$ generates an authenticator according to the original scheme. Otherwise, $\mathcal{C}_2$ takes list $\mathcal{L}_2$ into account and simulates as follows.
   - Retrieve the private key α_i from list $\mathcal{L}_1$.
   - Compute $\sigma_j = [(g_1^{h_i} \cdot R_i \cdot mpk)^{m_j} \cdot g_2^{H_3(\omega_j)}]^{\alpha_i}$ for $h_i = H_1(ID_i)$ and respond σ_j to $\mathcal{A}_2$.

8. Forge: $\mathcal{A}_2$ stops making queries and forges a σ' for data block m' with $Y_{ID'} = (Y_{ID'1}, Y_{ID'2}, Y_{ID'3})$, where $Y_{ID'}$ is the public key for ID'. If $ID' \neq ID^*$, $\mathcal{C}_1$ aborts simulation. Otherwise, $\mathcal{C}_2$ calculates $\psi(y) = \sum_{i=0}^{q-2} \tau_i y^{i+1}$ for some $(\tau_1, \tau_2, \cdots, \tau_{q-1}) \in (Z_p^*)^{q-1}$ and expands $P(y)$ as

$$P(y) = \psi(y) \cdot [(s + h_i) + y] + d$$

where $d \in Z_p^*$. From $\mathcal{L}_1$, it finds $Y_{ID'3} = g_1^{\frac{sh_{ID'}}{[s+h_i+x]} \cdot \frac{1}{x_{ID'}}}$. After that, $\mathcal{C}_2$ computes $\mathcal{T}$ as

$$\begin{aligned}\mathcal{T} &= [(Y_{ID'3})^{\frac{x_{ID'}}{s \cdot h_{ID'}}} \cdot \prod_{i=1}^{q-2} (A_{i+1})^{-\tau_i}]^{\frac{1}{d}} \\ &= [g^{\frac{P(x)}{s+h_i+x}} \cdot g^{-\psi(x)}]^{\frac{1}{d}} \\ &= [g^{\psi(x) + \frac{d}{r_i+h_i+x}} \cdot g^{-\psi(x)}]^{\frac{1}{d}} \\ &= [g^{\frac{d}{r_i+h_i+x}}]^{\frac{1}{d}} = g^{\frac{x}{r_i+h_i+x}}\end{aligned}$$

$\mathcal{C}_2$ computes $Z = e(g, \mathcal{T}) = e(g, g)^{\frac{x}{(r_i+h_i)+x}}$. If $a = r_i + h_i$, the value of Z is equal to $e(g, g)^{\frac{x}{a+x}}$. As a result, the q-EBSDH problem is solved.

Probability analysis: The advantage of solving the q-BSDH problem is computed similarly to Lemma 1 as follows. Now, we describe a detailed analysis of each event. Let us denote the number of iterations as n and the challenged identity ID^* is one of n identities.

- The probability of Γ_1 is equality to the probability that the simulation of Private-Key-Query and Authenticator-Query succeed simultaneously.
  - During the simulation of Private-Key-Query, $\mathcal{C}_2$ does not abort with success probability $(1 - \frac{1}{n})^{n_{pri}}$.
  - Authenticator-Query successfully produces an authenticator to train $\mathcal{A}_2$ for $ID \neq ID^*$ with probability $(1 - \frac{1}{n})$. Hence, the probability that the simulation does not terminate after n_{ag} times is at least $(1 - \frac{1}{n})^{n_{ag}}$.
- The probability of $\mathcal{A}_2$ forges a valid authenticator is ϵ.
- $Pr[\Gamma_3 | \Gamma_1 \wedge \Gamma_2]$ means that the probability $\mathcal{A}_2$ generates a valid forged authenticator for $ID = ID^*$ by conditioning on the events Γ_1 and Γ_2. As the advantage of generating a forged authenticator successfully is $\frac{1}{n_c}$

Therefore, there are

$$\begin{aligned}Pr[\Gamma_1] &= (1 - \frac{1}{n})^{n_{pri}+n_{ag}} \\ &\geq (1 - \frac{n_{pri} + n_{ag}}{n}) \\ Pr[\Gamma_2 | \Gamma_1] &\geq \epsilon \\ Pr[\Gamma_3 | \Gamma_1 \wedge \Gamma_2] &\geq \frac{1}{n_c}.\end{aligned}$$

Therefore, $\mathcal{A}_2$ can break q-BSDH problem

$$\epsilon' = Pr[\Gamma_1 \wedge \Gamma_2 \wedge \Gamma_3] \geq (1 - \frac{n_{pri} + n_{ag}}{n}) \frac{\epsilon}{n_c}$$

Since ϵ is negligible, $\mathcal{C}_2$ can not break q-BSDH problem. Therefore, our scheme can resist Type-II attack.

Estimated time: As in Lemma 1, the time cost to solve q-BSDH is $t' = t + O((n_c(q+3) + 3n_p + 4n_{ag})T_E)$.

□

Table 2 Security comparisons

	[9]	[5]	[24]	Ours
Revocation	✓	✓	✓	✓
Data Privacy	✗	✗	✓	✓
Withour ROM	✗	✗	✗	✓
collusion resistance	✗	✗	✗	✓

5.6 Security comparisons

As shown in Table 2, we compare our proposed scheme with Li et al.'s scheme [9], Gudeme et al.'s scheme [5] and Xu et al.'s scheme [24]. In Table 2, "✓" indicates that the scheme is secure, while "✗" indicates that the scheme is security weakness. We find that Gudeme et al.'s scheme [5] and Li et al.'s scheme [9] can not support data privacy. Although all schemes can provide user revocation, only our scheme can withstand collusion attack. Except for our proposed scheme, all of the above schemes have been proven secure using the random oracle model. Compared with other schemes, our proposed scheme can meet all secure requirements listed in Table 2.

Table 3 Comparison of computational cost

	Authenticator generation	Authenticator verification	user revocation
[9]	$2T_{exp} + 2T_{\mathbb{G}_1} + T_H$	$3T_P + 2T_{exp} + T_{\mathbb{G}_2}$	$2T_{exp} + T_{\mathbb{G}_1}$
[5]	$2T_{exp} + 2T_{\mathbb{G}_1} + T_H$	$3T_P + 2T_{exp} + 2T_{\mathbb{G}_1} + T_{\mathbb{G}_2}$	$2T_{exp} + T_{\mathbb{G}_1}$
[24]	$2T_{exp} + T_{\mathbb{G}_1} + T_H$	$3T_P + 2T_{exp} + T_{\mathbb{G}_1} + T_{\mathbb{G}_2}$	$2T_{exp} + T_{\mathbb{G}_1}$
ours	$4T_{exp} + 3T_{\mathbb{G}_1} + T_h$	$2T_P + 3T_{exp} + 2T_{\mathbb{G}_2}$	T_{exp}

6 Performance Analysis

In this section, we give the performance of our scheme in terms of the communication and computational cost. Assume n data blocks are stored in the cloud and c challenged blocks are picked. Denote id as the size of a block index. The performance evaluation and experimental results are illustrated below.

6.1 Performance Evaluation

Computational Cost: For convenience, we set T_P as the cost of one pairing operation, T_{exp} as the cost of one exponentiation operation on $\mathbb{G}_1$, $T_{\mathbb{G}_1}$ as the cost of one multiplication operation on $\mathbb{G}_1$ and $T_{\mathbb{G}_2}$ as the cost of one multiplication operation on $\mathbb{G}_2$. T_H represents the cost of a Hash-to-point operation and T_h represents the cost of a general hash function. In Table 3, the cost of *AuthenticatorGen* in our scheme is about $4T_{exp}+3T_{\mathbb{G}_1}$, which is lightly higher than previous schemes [5, 9, 24]. The verifier executes the algorithm *ProofVerify* and checks the correctness with the computational cost $2T_P+3T_{exp}+2T_{\mathbb{G}_2}$. In *Revoke*, GM generates the new group private key and its computational overhead is T_{exp}.

Communication Cost: In *Challenge* phase, TPA transmits the challenge sequences *chal* to CSP, whose size is $c \cdot (|id| + |q|)$ bits. The communication overhead of proof between CSP and TPA is $|\mathbb{G}_1| + |\mathbb{G}_2| + |q|$. The communication cost for user revocation is about $|q| + |\mathbb{G}_1|$.

6.2 Experimental Results

In recent years, several implementations of operations related to bilinear pairing have been reported. MIRACL, which is a well-known C/C++ library for multiprecision integer and rational arithmetic, was used to accomplish all bilinear pairing operations on Pentium IV with 3 GHz.

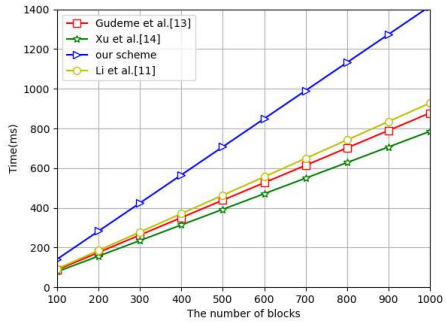

Fig. 2 Time of authenticator generation

Figure 2 evaluates the authenticator generation overhead of our scheme and [5, 9, 24]. The number of data blocks increases from 100 to 1000 in this experiment. The cost of authentication generation grows linearly in the number of data blocks. However, *AuthenticatorGen* executes once, which impacts lightly on the performance of our proposed scheme.

Figure 3 depicts the efficiency of integrity verification. In this experiment, the challenged block ranges from 100 to 1000. The cost of verification in all schemes is linearly proportional to the number of challenged blocks. However, the proposed scheme has less overhead costs than other schemes since they have one more bilinear operation. So we conclude that our scheme can save a lot of computational cost.

During the revocation, the cloud produces the proxy key between legal users and revoked users in [5, 9, 24], which is used to update the authenticators of revoked users. This is an additional computational overhead, however, our scheme has no the issue. Figure 4 gives the comparison in user revocation cost between our scheme and other schemes. The cost in previous schemes becomes larger as the number of blocks increases, while our scheme is unrelated to the block number. Thus, our scheme achieves high efficiency.

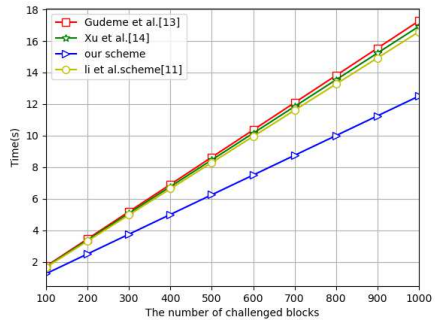

Fig. 3 Time of verifying proof

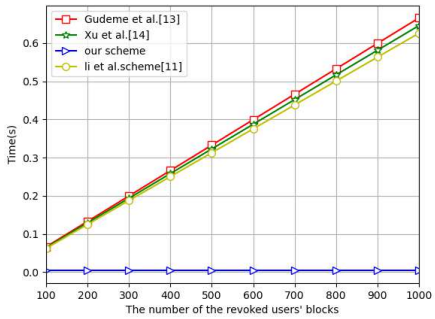

Fig. 4 Time of user revocation

7 Conclusion

In this paper, we present a novel provable data possession scheme based on certificateless public key cryptography for group shared data. The new scheme supports data privacy, user revocation and collusion resistance. Without considering random oracle model, our scheme is proved to be secure. The performance evaluation and experiment results demonstrate that our protocol has good efficiency.

Declarations

Author contributions GY contributed to the concrete scheme, performed the experiment and wrote the manuscript; LH helped design the scheme and analyzed the security of the scheme; JB and FW analyzed the security of our scheme.

Funding This research is sponsored by the National Natural Science Foundation of China under Grant U21A20466, Grant 61702153 and Grant 61972124.

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript

Data availability None.

Ethical approval We confirm that this work is original and has not been published elsewhere, nor it is currently under consideration for publication elsewhere. This article does not contain any studies with animals performed by any of the authors. This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] Al-Riyami SS, Paterson KG (2003) Certificateless public key cryptography. In: International conference on the theory and application of cryptology and information security, Springer, pp 452–473

[2] Ateniese G, Burns R, Curtmola R, et al (2007) Provable data possession at untrusted stores. In: Proceedings of the 14th ACM conference on Computer and communications security, pp 598–609

[3] Ateniese G, Di Pietro R, Mancini LV, et al (2008) Scalable and efficient provable data possession. In: Proceedings of the 4th international conference on Security and privacy in communication netowrks, pp 1–10

[4] Erway CC, Küpçü A, Papamanthou C, et al (2015) Dynamic provable data possession. ACM Transactions on Information and System Security (TISSEC) 17(4):1–29

[5] Gudeme JR, Pasupuleti SK, Kandukuri R (2021) Certificateless multi-replica public integrity auditing scheme for dynamic shared data in cloud storage. Computers & Security 103:102,176

[6] He D, Zeadally S, Wu L (2015) Certificateless public auditing scheme for cloud-assisted wireless body area networks. IEEE Systems Journal 12(1):64–73

[7] He K, Chen J, Yuan Q, et al (2019) Dynamic group-oriented provable data possession in the cloud. IEEE Transactions on Dependable and Secure Computing 18(3):1394–1408

[8] Ji Y, Shao B, Chang J, et al (2022) Flexible identity-based remote data integrity checking for cloud storage with privacy preserving property. Cluster Computing 25(1):337–349

[9] Li J, Yan H, Zhang Y (2018) Certificateless public integrity checking of group shared data on cloud storage. IEEE Transactions on Services Computing 14(1):71–81

[10] Li J, Yan H, Zhang Y (2020) Identity-based privacy preserving remote data integrity checking for cloud storage. IEEE Systems Journal 15(1):577–585

[11] Li Y, Yu Y, Yang B, et al (2018) Privacy preserving cloud data auditing with efficient key update. Future Generation Computer Systems 78:789–798

[12] Liu X, Zhang Y, Wang B, et al (2012) Mona: Secure multi-owner data sharing for dynamic groups in the cloud. IEEE transactions on parallel and distributed systems 24(6):1182–1191

[13] Luo Y, Xu M, Huang K, et al (2018) Efficient auditing for shared data in the cloud with secure user revocation and computations outsourcing. Computers & Security 73:492–506

[14] Shen W, Qin J, Yu J, et al (2018) Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. IEEE Transactions on Information Forensics and Security 14(2):331–346

[15] Tian H, Chen Y, Chang CC, et al (2015) Dynamic-hash-table based public auditing for secure cloud storage. IEEE Transactions on Services Computing 10(5):701–714

[16] Tian H, Nan F, Jiang H, et al (2019) Public auditing for shared cloud data with efficient and secure group management. Information Sciences 472:107–125

[17] Tian J, Jing X (2020) Cloud data integrity verification scheme for associated tags. Computers & Security 95:101,847

[18] Wang B, Li B, Li H (2012) Knox: privacy-preserving auditing for shared data with large groups in the cloud. In: International conference on applied cryptography and network security, Springer, pp 507–525

[19] Wang B, Li B, Li H (2013) Panda: Public auditing for shared data with efficient user revocation in the cloud. IEEE Transactions on services computing 8(1):92–106

[20] Wang B, Li B, Li H, et al (2013) Certificateless public auditing for data integrity in the cloud. In: 2013 IEEE conference on communications and network security (CNS), IEEE, pp 136–144

[21] Wang B, Li B, Li H (2014) Oruta: Privacy-preserving public auditing for shared data in the cloud. IEEE transactions on cloud computing 2(1):43–56

[22] Wang C, Chow SS, Wang Q, et al (2011) Privacy-preserving public auditing for secure cloud storage. IEEE transactions on computers 62(2):362–375

[23] Wang Q, Wang C, Ren K, et al (2010) Enabling public auditability and data dynamics for storage security in cloud computing. IEEE transactions on parallel and distributed systems 22(5):847–859

[24] Xu Z, He D, Vijayakumar P, et al (2021) Certificateless public auditing scheme with data privacy and dynamics in group user model of cloud-assisted medical wsns. IEEE Journal of Biomedical and Health Informatics

[25] Yan H, Liu Y, Zhang Z, et al (2021) Efficient privacy-preserving certificateless public auditing of data in cloud storage. Security and Communication Networks 2021

[26] Yu Y, Au MH, Ateniese G, et al (2016) Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Transactions on Information

Forensics and Security 12(4):767–778

[27] Zhang Y, Yu J, Hao R, et al (2018) Enabling efficient user revocation in identity-based cloud storage auditing for shared big data. IEEE Transactions on Dependable and Secure computing 17(3):608–619

[28] Zhou L, Fu A, Yang G, et al (2020) Efficient certificateless multi-copy integrity auditing scheme supporting data dynamics. IEEE Transactions on Dependable and Secure Computing

[29] Zhu Y, Ahn GJ, Hu H, et al (2011) Dynamic audit services for outsourced storages in clouds. IEEE transactions on services computing 6(2):227–238

[30] Zhu Y, Hu H, Ahn GJ, et al (2011) Collaborative integrity verification in hybrid clouds. In: 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), IEEE, pp 191–200