The Journal of Supercomputing (2018) 74:6156–6183
https://doi.org/10.1007/s11227-018-2527-y

Privacy-preserving auditing scheme for shared data in public clouds

Libing Wu[1,2] · Jing Wang[1,2] · Sherali Zeadally[3] · Debiao He[1,2,4]

Published online: 13 August 2018

Abstract

Recent advances in cloud storage have enabled users to outsource large amounts of data to a remote cloud server in order to reduce storage and management costs, and share files among many users in a group. However, how to efficiently audit the integrity of shared data while maintaining data privacy and user identity anonymity, is still a critical issue. We propose a novel public auditing scheme for data stored in a remote cloud server and shared among users in a large group. In particular, the proposed scheme incorporates group signature, homomorphic message authentication code to create data block tags, so that it can support public auditing and provide user identity anonymity. Furthermore, we use the random masking technique in the proposed scheme to preserve data privacy from the third-party auditor. The correctness and security analyses demonstrate that the proposed scheme is correct and provably secure under a robust security model. The performance evaluation and experimental results show that the proposed scheme is efficient while maintaining the desirable security properties.

Keywords Identity anonymity · Privacy preservation · Cloud data · Public auditing

✉ Debiao He
hedebiao@163.com

Libing Wu
whuwlb@126.com

Jing Wang
cswjing@whu.edu.cn

Sherali Zeadally
szeadally@uky.edu

1 School of Computer Science, Wuhan University, Wuhan, China

2 State Key Laboratory of Cryptology, Beijing, China

3 College of Communication and Information, University of Kentucky, Lexington, USA

4 Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China

1 Introduction

As the amount of data continues to grow in today's world, cloud storage services are becoming increasingly important. Instead of storing data in a local storage system, outsourcing large amounts of data to a remote cloud server can greatly relieve the individuals and enterprises from the burden of data maintenance and management [1,2]. Cloud benefits such as elastic storage, high reliability, affordable management, location independence, on-demand self-service and ubiquitous network access [3] attract more and more users toward the cloud service for data storage and sharing, as shown in Fig. 1. According to Cisco [4], there will be 2 billion individuals who will use the cloud storage service by 2018 [5].

Despite the benefits of cloud storage which is managed by the cloud service providers (CSPs), the cloud storage service brings new threats and challenges to the security of user data in terms of integrity, availability, secure sharing and utilization. For instance, the cloud infrastructure may suffer from some inevitable failures and a wide range of external attacks, which can lead to the data corruption (e.g., some data stored on Amazon's cloud are destroyed permanently due to the crash disaster), but the CSP may hide this data loss in order to avoid a bad reputation. Even worse, a CSP may deliberately discard some sensitive data that are not frequently visited by users for financial reasons. Users have no idea if their outsourced data are stored correctly on the selected cloud server because they lose physical control of their data on the remote cloud servers. Hence, how to guarantee the data integrity on a remote cloud server becomes a key issue for cloud storage services.

To address the aforementioned security issue, various auditing schemes have been proposed in recent years. An auditing scheme is designed to help an auditor verify the integrity of outsourced data periodically without downloading the entire data file from the remote cloud server. In an auditing scheme, the original data file is considered to consist of n data blocks. It is typical for a user to create a corresponding signature or

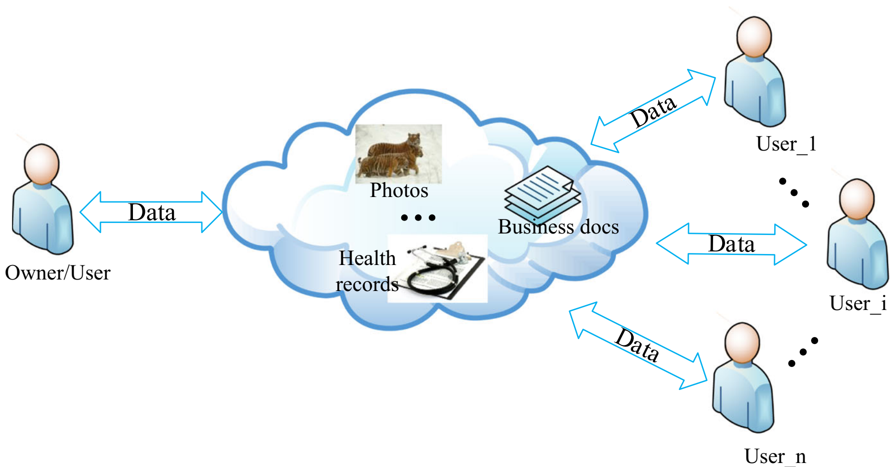

Fig. 1 The framework of cloud storage

tag for each data block, and then outsource all data signatures to a cloud server. During the auditing procedure, an authorized auditor can implement a challenge-protocol to verify the data integrity. Since the auditing proof responded by the cloud server is generated by aggregating the sampled data blocks and their corresponding signatures, it is essential for each data signature to be unforgeable. On the other hand, the value of aggregated data blocks, as part of auditing proof, should also be unforgeable.

Generally, the user is limited by the computational resources available, so that an authorized third party [known as third-party auditor (TPA)] is introduced to audit the data integrity instead of the original user. As TPA is a third party, it should not have any knowledge of the data content during the auditing procedure. In addition, several users can easily form a group for information sharing (e.g., an academic debate forum, where many researchers can upload files and share information with each other). In this case, the researchers in the group would not like to leak their private identity for security.

If a public auditing scheme cannot guarantee the user's identity privacy, it is possible for the TPA to know which blocks are more important, and which user is more important among the users in the current group after performing several audits. Then, the frequently audited data may suffer from more attacks and attract more attackers, such as address tracing. Thus, an auditing scheme for shared data must protect the identity anonymity of group users from the TPA and the cloud server.

However, absolute anonymity may bring some other integrity and security issues to the data owner or users. For example, a malicious data owner can damage or modify part of the shared data because of some reasons, while the property of absolute anonymity can provide a protective umbrella to the malicious user and help the user from taking any blame. In this situation, the property of absolute anonymity will lead to unpredictable financial or reputational losses. Therefore, both identity anonymity and traceability are important to a public auditing scheme for shared data.

In recent years, although many public auditing schemes for the shared data have been proposed, they still suffer from different security attacks or they incur high computation/communication overheads making them unsuitable for deployment in practical applications. Therefore, designing a secure and efficient public auditing scheme for shared data remains a significant challenge. In this work, we investigate how to achieve a secure and efficient public auditing scheme for shared data on a remote cloud server.

1.1 Our contributions

Based on the above description, we note that the unforgeability of auditing proofs, the anonymity of both shared data and user identity in cloud auditing are significant for group users. To address these security issues, we present a new efficient public auditing scheme for shared data on the remote cloud server based on Tzu-Hsin Ho's group signature scheme [6]. We summarize our research contributions as follows:

- First, we propose a new public auditing scheme for shared data with high efficiency. In the proposed scheme, we use the group signature and homomorphic message authentication code (MAC) to create signatures for data blocks of users in a group.

And we authorize a TPA to support public data auditing by sharing a secret key pair with it.

- Second, we analyze the security of the proposed scheme against signature forgery attack and proof forgery attack. Moreover, we prove that the proposed scheme can preserve data privacy, protect identity anonymity of users and provide conditional identity traceability.
- Finally, we evaluate the performance of the proposed scheme and demonstrate its efficiency.

1.2 Organization

The remainder of this paper is structured as follows: In Sect. 2, we briefly review some previous related works. In Sect. 3, we present some background related to the proposed scheme. In Sect. 4, we describe our proposed scheme. In Sect. 5, we analyze the proposed scheme in terms of its correctness and security. Then, in the following section, we evaluate the efficiency of the proposed scheme. Finally, we make some concluding remarks.

2 Related work

Data integrity verification is of vital importance when verifying the storage correctness of outsourced data on a remote cloud server. Ateniese et al. [7] first came up with the provable data possession (PDP) concept to avoid downloading entire files from a remote cloud server while auditing data integrity. In their scheme, homomorphic tags are utilized to aggregate the sample blocks and corresponding signatures in order to reduce communication costs. Then, Shacham and Waters proposed the compact proofs of retrievability (CPoR) concept [8] with a concrete construction based on the Boneh–Lynn–Shacham (BLS) signature [9], and the scheme is provable secure under the random oracle model [10]. Since then, several other public auditing schemes [11–15] have been proposed to meet diverse application requirements.

To reduce the management overhead of certificate in PKI-based auditing schemes [7,16,17], the identity-based cryptography has been adopted in many public auditing schemes [18,19]. For instance, Yu et al. [20] presented a generic construction of identity-based provable data possession (ID-PDP) scheme, and described a concrete ID-based auditing scheme as well. Wang et al. [21] proposed an identity-based proxy-oriented data uploading and auditing scheme, which is specialized for some business managers who are not able to upload or audit the outsourced data in person. Li et al. [22] presented a novel auditing scheme by using biometric as a fuzzy identity to address the complex key management issue.

Although some organizations require strong security in key management, the identity-based cryptography is not the best choice to adopt when checking the integrity of outsourced data because of the key escrow problem. It means that the private key of a user is revealed to a third party (refer in particular to the KGC), so that the KGC can impersonate the user to do some unexpectable things and damage the user's benefit.

Therefore, certificateless public auditing schemes have been proposed. Wang et al. [23] first presented a secure certificateless public auditing scheme for data integrity verification. Zhang et al. [24] are the first researchers to propose a certificateless public integrity verification scheme considering the security against a malicious TPA simultaneously. He et al. proposed a certificateless public auditing scheme for cloud-assisted wireless body area networks (WBANs) [25], which is proved to be more efficient than Wang et al.'s scheme [23].

Besides secure key management, the property of data privacy preservation is also important for public data integrity verification. Early on, most public auditing schemes suffered from security weakness owing to the linear combination of sampled data blocks when aggregating them as one part of the auditing proof, e.g., $\sum_{i \in L} v_i m_i$, where v_i is a random integer and m_i is the sampled block. For example, in scheme [26], Yang et al. analyzed the issue of data privacy in detail, and they pointed out that the scheme Panda [16] is vulnerable to the proof forgery attack and cannot keep the data privacy against a curious TPA. They further proposed an improved scheme to address the two weaknesses by using the random masking technique, that is, they added a random element to each aggregation (i.e., $\sum_{i \in L} v_i m_i + \eta$). Similarly, Xu et al. analyzed the security of Tang and Zhang's public auditing scheme [27] in [28]. And they pointed out that Tang and Zhang's scheme suffers from the same vulnerabilities as Panda. Recently, Li et al. [29] proposed a privacy-preserving scheme specialized for low-performance end devices in a cloud environment. Later, Shen et al. used a third-party entity in their public auditing scheme which preserves data privacy. Since the third party performs almost all the computation operations instead of the group users, their scheme is pretty efficient in computation cost [30]. However, the scheme suffers from communication costs than most previously proposed auditing schemes due to the additional communication overheads incurred between the user and the third party entity.

When it comes to the privacy preservation, identity anonymity is also an essential property for cloud auditing, especially for the case where many group users share their data on a remote cloud server. Luo et al. designed an efficient integrity auditing scheme for shared data to achieve secure user revocation, which uses the concept of Shamir Secret Sharing to address the weakness of proxy re-signatures [31]. Wang et al. proposed an integrity checking and sharing scheme for remote data in a cloud-based health Internet of Things (IoT) environment [32]. But both schemes do not consider the identity privacy. In 2015, He et al. proposed a public auditing scheme for shared data aimed at preserving identity privacy [33]. Their scheme converts each user's signature into the TPA's signature with a re-signed key, which perfectly protect the identity privacy of users from the TPA, but the scheme cannot provide identity traceability.

In the case of auditing shared data, Wang et al. proposed several public auditing schemes, such as *Panda* [16], *Oruta* [34], *Knox* [35]. Panda performs efficient user revocation without data privacy and user identity anonymity based on proxy re-signatures. *Oruta* utilizes the ring signature and homomorphic authenticators to achieve public auditing and identity anonymity. But for *Oruta*, the communication and computation cost increase linearly with the number of group users. To improve the efficiency, *Knox* proposed a scheme that use group signature. The scheme assures user identity anonymity from the TPA, and also supports the traceability of user identity by the group manager. However, the communication and computational efficiency of *Knox* should be further improved. Although many public auditing schemes with various functions and security features have been proposed in recent years, but none of them can satisfy all above security requirements (i.e., public auditing, data sharing, data privacy, user identity anonymity, data traceability and high performance). Therefore, it is important to design a more novel auditing scheme including all above practical features.

3 Background

3.1 Preliminaries

(1) Bilinear maps

Let G_1, G_2 and G_T denote three cycle groups of the same prime order q. Note that g_1 and g_2 are the generators of G_1 and G_2, respectively. An efficient computable bilinear map can be denoted as $e : G_1 \times G_2 \rightarrow G_T$ with the following two properties:

- Bilinearity: Given $\forall a, b \in Z_q$ and $\forall u \in G_1$, $\forall v \in G_2$, the equation $e(u^a, v^b) = e(u^a, v)^b = e(u, v^b)^a = e(u, v)^{ab}$ holds.
- Non-degenerate: The equation $e(g_1, g_2) \neq 1_{G_T}$ holds.

(2) Complexity assumptions

The security of the proposed scheme relies on the following three assumptions:

- *Decisional Diffie–Hellman (DDH) assumption*
  Given four elements $g, g^a, g^b, g^c \in G$, there is no polynomial algorithm to decide whether $g^c = g^{ab}$, where a, b, c are randomly chosen from the group Z_q.
- *Computational Diffie–Hellman (CDH) assumption*
  Given three elements $g, g^a, g^b \in G$, there is no polynomial algorithm to calculate the value of g^{ab} with non-negligible probability, where a, b are randomly chosen from group Z_q.
- *Elliptic curve discrete logarithm (ECDL) Assumption*

Given two element $g, g^a \in G$, it is computationally infeasible to compute the value of a, where a is randomly chosen from the group Z_q, and g is a random point in group G.

(3) Elliptic ElGamal encryption

The proposed scheme uses an efficient group signature to achieve data integrity auditing, while the Elliptic ElGamal encryption is embedded into this group signature to protect the private keys of group users. The encryption process can be described as follows:

Assume that G is an elliptic curve group of large prime order q, and g is a generator of G. Let a random element $x \in Z_q$ be an encryptor's private key, and $pk = g^x \in G$ be the public key. Given a message $m \in G$, the encryptor first selects a temporary key

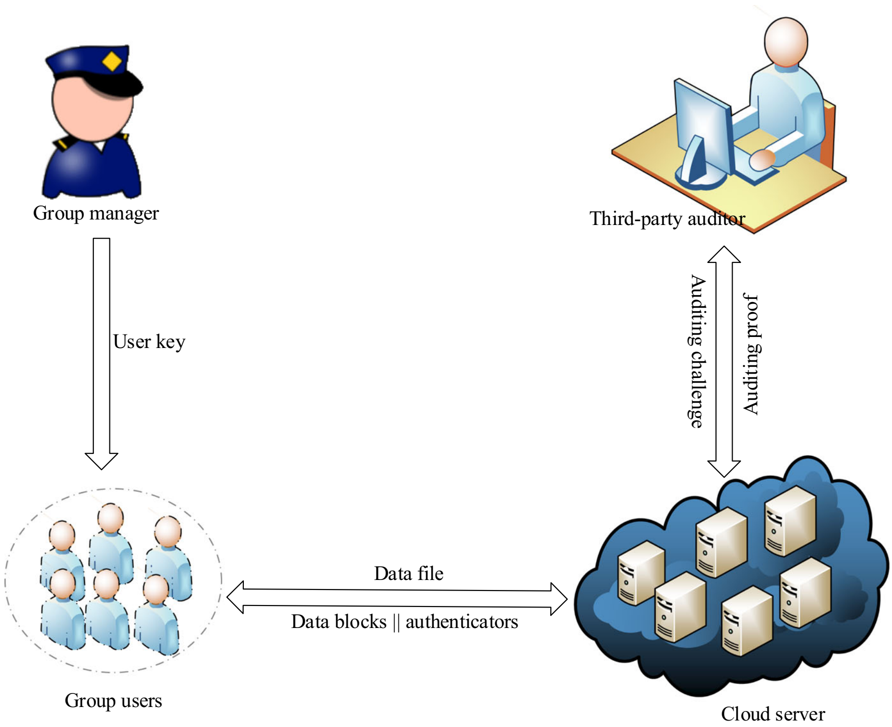

Fig. 2 The proposed system model

$y \in Z_q$ randomly, and then computes $C_1 = g^y \in G, C_2 = m \cdot pk^y$, and finally returns (C_1, C_2) as its ciphertext. To encrypt, a decryptor must know the decryptor's private key x, and computes C_2/C_1^x to get the plaintext m. The Elliptic ElGamal encryption is secure if the DDH assumption holds.

3.2 System model

The system model of the proposed scheme consists of four entities, as illustrated in Fig. 2:

- *Group manager* The group manager is a special group user who is trusted by all other group users and is responsible for generating a distinct signing key pair of each group user.
- *Group users* The group users are responsible for generating signatures of data blocks, and then outsourcing them to a cloud server.
- *Third-party auditor* The third-party auditor (TPA) is authorized by the group manager and users to check the storage correctness of shared data on a remote cloud server.
- *Cloud server* The cloud server which is managed by a semi-trusted cloud service provider(CSP) stores and manages large amount of data.

3.3 Design goals

To achieve a secure and efficient auditing process, the scheme should satisfy some basic requirements as follows:

(1) *Correctness* The auditor should be able to verify the integrity of shared data on the remote cloud server only if the data are stored correctly.
(2) *Public auditing* To reduce the computation overhead of group users, an authorized third-party auditor is allowed to check the data integrity of shared data.
(3) *Privacy preservation* Since the TPA is honest but curious, the data content should be kept secret from the TPA. That is, even an authorized TPA cannot get any knowledge of the data content from the auditing proofs.
(4) *Anonymity* Given an outsourced file, an adversary (as well as the TPA and other group users) cannot reveal the identity of file owners according to the signatures attached to each data block.
(5) *Traceability* The group manager can disclose the identity of any group user if that user uploads malicious files.
(6) *Efficiency* The auditor (e.g., the group user or TPA) can efficiently verify the data integrity without downloading the entire data file, and the computation cost is constant during the auditing process.

4 The proposed scheme

In this section, we first present an overview of the proposed scheme, and then we describe our design approach.

4.1 Overview

To address the design goals mentioned in previous auditing schemes for shared data on remote cloud, we propose a privacy-preserving efficient auditing scheme by utilizing the group signature and Homomorphic MACs presented by Tzu-Hsin Ho [6], Agrawal and Boneh [36], respectively. According to [6], the group signature evolved from the Elliptic ElGamal encryption algorithm, which satisfies semantic security under the DDH assumption. Homomorphic MACs is applied to create homomorphic signatures when developing the data auditing mechanism.

On the basis of this group signature, we can keep the identity anonymity of signers on the shared data blocks from TPA while checking the data integrity. Similar to other group signature schemes, the group manager which is considered to be the original data owner, can track the actual identity of each group signer if it is necessary. Moreover, the proposed scheme can keep the privacy of shared data through utilizing data masking technique. By making use of the short group signature and Homomorphic MACs, the proposed scheme can generate a constant-size public key and a constant-size signature which will help reduce communication and computation costs.

Specifically, the proposed scheme uses eight algorithms: **SysInit**, **GroupSetup**, **Enroll**, **GroupSign**, **ProofGen**, **ProofVerify**, **Open** and **Revocation**. All public sys-

Table 1 Summary of some important notations

Notion	Description
g_i	Generator of group $G_i (i = 1, 2)$
thk	Secret key shared between the group and the authorized TPA
msk	Master secret key of the group, i.e., $msk = (d, u, t, \xi \in Z_q^*)$
gpk	Group public key, i.e., $gpk = (D, U, T, X, \eta)$, where $D = g_1^d$, $T = g_1^t$, $U = g_2^u$, $X = g_2^{\xi}$ and $\eta \in Z_q^*$
$usk[i]$	The i-th user's private signing key, i.e., $usk_i = (x_i, Z_i, \xi)$, where $Z_i \in G_1, x_i, \xi \in Z_q^*$
n	The number of data blocks
s	The number of slices in each data block
M	The file with n number of data blocks $m_1, \ldots, m_n$
$a_1 \cdots a_s$	Elements generated by pseudo-random generator PRG
$b_1 \cdots b_n$	Elements generated by pseudo-random function PRF
y_j	Temporary secret key of the j-th block selected by the group user
fn	Name of the file M
σ_j	Signature of the j-th data block computed by the group user, i.e., $\sigma_j = (C_{j1}, C_{j2}, \tau_j, w_j, \theta_j)$
$\gamma_1 \cdots \gamma_s$	Elements from group Z_q^* used to blind the data block
$\|\|$	Message concatenation operation
pf	Auditing proof, i.e., $pf = \{\Theta, \vec{\mu}, \Omega, \vec{\Upsilon}\}$

tem parameters are generated by taking a security parameter as input with the **SysInit** algorithm. The **GroupSetup** algorithm is executed by the group manager to generate the group key pairs, where the group private key is kept secret and only the manager knows it, while the public key is shared among group users and the authorized TPA. With the **Enroll** algorithm, a valid group user can be assigned a distinct key pair by the group manager, which is used to sign the data blocks in the **GroupSign** algorithm. The **ProofGen** and **ProofVerify** algorithms are executed to check the storage correctness of shared data on the remote cloud. As for the **Open** algorithm, the group manager is able to recognize the identity of the signer who uploads illegal files. Furthermore, the group manager can revoke that malicious user by following the **Revocation** algorithm.

4.2 Detailed design

Some important notations are shown in Table 1. We then describe the detailed design of our proposed scheme as follows:

SysInit(λ) Taking as input the security parameter λ, the system outputs three cyclic groups G_1, G_2, G_T of λ-bit prime order q, and let $e : G_1 \times G_2 \longrightarrow G_T$ be an efficient bilinear map. g_1 and g_2 denote the generators of G_1 and G_2, respectively. It is worth noting that there is no efficiently computable isomorphism from G_1 to G_2. There are two hash functions $h : \{0, 1\}^* \longrightarrow Z_q$ and $H : \{0, 1\}^* \longrightarrow G_1$, which are considered as random oracles in the proof of security. Then, the system chooses a pseudo-random generator PRG: $K_{prg} \longrightarrow Z_q^k$ and a pseudo-random function PRF: $K_{prf} \times I \longrightarrow Z_q$, where K_{prg} and K_{prf} are the set of secret keys for PRG and PRF, respectively, and I is the index table of data blocks. Finally, the system publishes these public parameters $params = \langle q, G_1, G_2, G_T, e, g_1, g_2, h, H, PRG, PRF \rangle$

GroupSetup$(params)$ Given the public parameters $params$, the group manager executes the algorithm to get group secret keys and public keys as follows:

- Randomly selects a key pair $thk = (a, b)$, where $a \in K_{prg}$ and $b \in K_{prf}$, and share the key pair $thk = (a, b)$ with the group user and the TPA via a secure channel.
- Choose four random elements $d, u, t, \xi \in Z_q^*$ as group master key, i.e., $msk = (d, u, t, \xi)$, where d, u are used to enroll new group users, t is used to trace the signatures of group users, and ξ is used to sign data blocks. Then, the group manager sets $D = g_1^d, U = g_2^u, T = g_1^t, X = g_2^\xi$ and $\eta \in Z_q^*$ as the group public key, i.e., $gpk = (D, U, T, X, \eta)$.
- Keeps the group master key $msk = (d, u, t, \xi)$ as secret, and publishes the public key $gpk = (D, U, T, X, \eta)$ among group users and the authorized TPA.

Enroll(i, msk) Given the index of a new group user i and a group master key msk, the group manager runs the algorithm to distribute a private key for group users as follows:

- Randomly selects an element $x_i \in Z_q^*$, and sets $Z_i = g_1^{(d-x_i)(ux_i)^{-1}} = g_1^{z_i} \in G_1$, where $z_i \in Z_q^*$. Actually, we can get the equation $d = x_i + z_i u x_i$.
- Re-selects another value x_i if the value of x_i is a distinct secret. Otherwise, the group manager records the key pair (x_i, Z_i, ξ) as i-th user's signing key in a table called, ***Tab***.
- Sends the secret key $usk[i] = (x_i, Z_i, \xi)$ to i-th user through a secure channel.

GroupSign$(gpk, thk, usk[i], M)$ Given the group public key $gpk = (D, U, T, \eta)$, the group private key thk, the user's private key $usk[i] = (x_i, Z_i)$ and the file $M = \{m_1, m_2, \ldots, m_n\} \in \{0, 1\}^*$, where $m_j = \{m_{j1}, m_{j2}, \ldots, m_{js}\}$, the user i executes the following steps to calculate corresponding signatures for each data block m_j.

- Selects a random element $y_j \in Z_q^*$, and computes $Q_j = e(U, T)^{y_j}$, $C_{j1} = g_1^{y_j}$ and $C_{j2} = Z_i^{x_i} \cdot T^{y_j}$, respectively.
- Computes $\vec{A} = \{a_1, a_2, \ldots, a_s\} \longleftarrow PRG(a)$ and $\vec{B} = \{b_1, b_2, \ldots, b_n\}$, where $b_j \longleftarrow PRF(b, j)$. The group manager then calculates $\pi_j = \sum_{l=1}^{s} a_l m_{jl} + b_j$.
- Computes $\tau_j = \eta^{\pi_j} h(C_{j1}, C_{j2}, Q_j)$ and $\omega_j = y_j \cdot \tau_j + x_i$.
- Computes a tag for the data block as $\theta_j = [H(fn \parallel j) g_1^{\pi_j}]^\xi$, where fn is a distinct random string assigned by the user i, as a unique file identifier of a file. We note that fn can be known by the TPA for auditing, because the file identifier does not contain the real content of data.
- Outsources the file M with its signatures $\sigma_j = (C_{j1}, C_{j2}, \tau_j, \omega_j, \theta_j)_{1 \le j \le n}$ to the remote cloud server.

ProofGen(fn, σ_j, n) This algorithm is an interactive protocol executed between the remote cloud server and the user/TPA for data integrity checking.

If the user cannot check the data integrity in person, he/she can delegate the auditing task to an authorized TPA by sending an auditing request.

Step 1 When receiving the auditing request, the TPA generates an auditing message and sends it to remote cloud server as follows:

- Randomly selects a c-element subset L from set $[1, n]$, where $j \in L$ denotes the index of a block sampled to check in the auditing process and c is the number of sampled data blocks to be audited.
- For each $j \in L$, generates the small integer $v_j \in Z_q^*$.
- Sends the auditing message $chal = \{(j, v_j), fn\}_{j \in L}$ to the remote cloud server.

Step 2 Upon receiving the auditing message $chal$ from the user or TPA, the cloud server generates the corresponding proof of data possession with the sampled data blocks as follows:

- Randomly selects a set of elements as $\gamma_1, \gamma_2, \ldots, \gamma_s$, and computes $\mu_l = \sum_{j \in L} v_j m_{jl} + \gamma_l$ for $1 \le l \le s$.
- Computes the aggregation of block tags as $\Theta = \prod_{j \in L} \theta_j^{v_j}$.
- Sends $pf = \{\Theta, \vec{\mu}, \Omega, \vec{\Upsilon}\}$ as the auditing proof, where $\vec{\mu} = \{\mu_l\}_{1 \le l \le s}$, $\vec{\Upsilon} = \{\delta_l, \eta_l\}_{1 \le l \le s}$, $\delta_l = g_1^{-\gamma_l}$, $\eta_l = \eta^{-\gamma_l}$, and $\Omega = \{\varphi_j\}_{j \in L}$, $\varphi_j = (C_{j1}, C_{j2}, Q_j, \tau_j, w_j)$, where $Q_j = \frac{e(C_{j2}, U) \cdot e(g_1, g_2)^{w_j}}{e(D \cdot C_{j1}^{\tau_j}, g_2)}$ can be precomputed by the cloud server to save the computation cost.

ProofVerify(pf, gpk, thk) Given that the auditing proof pf, group public key gpk and the private key thk are only shared among group members and the authorized TPA, the user/TPA can execute the following steps to check the correctness of data storage:

- Checks the following equation

$$\prod_{j \in L} Q_j \stackrel{?}{=} \frac{e(\prod_{j \in L} C_{j2}, U) \cdot e(g_1, g_2)^{\sum_{j \in L} w_j}}{e(D^c \cdot \prod_{j \in L} C_{j1}^{\tau_j}, g_2)} \tag{1}$$

  If it does not hold, the TPA aborts the procedure and outputs "0"; otherwise, it continues to do the following steps for checking Eqs. (2) and (3):
- Computes $\vec{A} = \{a_1, a_2, \ldots, a_s\} \longleftarrow PRG(a)$ and $b_j \longleftarrow PRF(b, j)$ for $j \in L$ by taking the private key thk as input.
- Computes $\varpi = \sum_{l=1}^{s} a_l \mu_l + \sum_{j \in L} v_j b_j$.

$$\prod_{j \in L} \tau_j^{v_j} \stackrel{?}{=} \eta^{\varpi} \cdot \prod_{l=1}^{s} \eta_l^{a_l} \cdot \prod_{j \in L} h(C_{j1}, C_{j2}, Q_j)^{v_j} \tag{2}$$

$$e(\Theta, g_2) \stackrel{?}{=} e\left(\prod_{j \in L} H(fn \parallel j)^{v_j} \cdot \prod_{l=1}^{s} \delta_l^{a_l} \cdot g_1^{\varpi}, X\right) \tag{3}$$

If all above three equations hold, the proof shows that the user's data are stored correctly on the remote cloud server and outputs "1"; otherwise, the data file is corrupted.

- Sends the auditing result to the user if the verifier is an authorized TPA.

Open(t, M, σ) Given the group private key t, data file M and the corresponding signature set σ, the group manager can trace a signature back to the actual group user if he/she uploads illegal files.

- Verifies if the signature set σ is a valid signature on file M through Eqs. (1) and (2).
- For arbitrary $j \in [1, n]$, compute

$$Z_i^{x_i} = C_{j2}/C_{j1}^t \tag{4}$$

Since the value of $Z_i^{x_i}$ denotes the multiplication of two parts of *usk[i]*, the group manager can reveal the actual identity of the user through traversing the table that can map $Z_i^{x_i}$ to a user's identity.

4.3 Support user revocation

Initially, the group manager publishes a revocation list, called ***RList***, to record the identity of those revoked/departed users. When a group user i always uploads junk files, the group manager has to expel the user from this group by adding x_i into ***RList***, where x_i is a part of the secret key that belongs to the revoked user i. Then, the group manager deletes the tuple (x_i, Z_i, ξ) from ***Tab***.

Once a user has been revoked, the cloud server does not need to respond to future requests from this user for security reasons. To distinguish whether the user is a revoked one, the cloud server can do a revocation test as follows:

Given a part of signatures $(C_{j1}, C_{j2}, \tau_j, w_j)$, for each record $x \in$ ***RList***, the cloud server can first compute the value of $\widetilde{Q}_j = \frac{e(C_{j2},U)\cdot e(g_1,g_2)^{w_j}}{e(D\cdot C_{j1}^{\tau_j},g_2)}$, then it checks the correctness of the following equation:

$$e(C_{j2}, T) \cdot e(g_1^x, g_2) \stackrel{?}{=} e(D, g_2) \cdot \widetilde{Q}_j \tag{5}$$

If the cloud server proves that Eq. (5) holds when $x = x_{i'}$, it can know the current data file and the corresponding signatures are provided by a revoked user i' so that it can refuse the user's storage request to save space. According to the specific policies of different groups, the group manager can choose to destroy the privacy of data uploaded by the revoked user, or resign the data by utilizing the proxy re-signing signature as referred in [35].

5 Correctness and security analysis

In this section, we present a detailed analysis of the proposed scheme according to the security requirements aforementioned in *Design goals*. The correctness of the scheme can be verified through a straightforward calculation based on the properties of a bilinear map. The following theorems support the security analysis of the proposed scheme.

5.1 Correctness analysis

Theorem 1 *The proposed scheme satisfies the correctness, i.e., the TPA can check the data integrity in the ProofVerify procedure as long as all entities follow the proposed scheme honestly.*

Proof The group user i can honestly generate a set of signatures $\{\sigma_j\}_{1\leq j\leq n}$ on file M with his private key $usk[i]$, which is generated by the honest group manager. Moreover, the cloud server correctly stores the user's data and follows the proposed scheme to generate the corresponding auditing proof $pf = \{\Theta, \mu_1, \mu_2, \ldots, \mu_s, \Omega, \vec{\Upsilon}\}$. Given the group public key $gpk = (D, U, T, \eta)$, the correctness of Eqs. (1), (2) and (3) can be proved as follows:

$$\begin{aligned}
& e\left(\prod_{j\in L} C_{j2}, U\right) \cdot e(g_1, g_2)^{\sum_{j\in L} w_j} \\
&= \prod_{j\in L}[e(Z_i^{x_i} \cdot T^{y_j}, U) \cdot e(g_1, g_2)^{y_j\cdot\tau_j + x_i}] \\
&= \prod_{j\in L}[e(g_1, g_2)^{x_i z_i u} \cdot e(T, U)^{y_j} \cdot e(g_1, g_2)^{y_j\cdot\tau_j + x_i}] \\
&= \prod_{j\in L}[e(g_1, g_2)^{x_i z_i u + y_j\cdot\tau_j + x_i} \cdot Q_j] \\
&= \prod_{j\in L}[e(g_1, g_2)^{d + y_j\cdot\tau_j} \cdot Q_j] \\
&= \prod_{j\in L}[e(D \cdot C_{j1}^{\tau_j}, g_2) \cdot Q_j] \\
&= e\left(D^c \cdot \prod_{j\in L} C_{j1}^{\tau_j}, g_2\right) \cdot \prod_{j\in L} Q_j
\end{aligned}$$

The above formula transformation shows that Eq. (1) holds.

Equation (2) can be proved as follows:

$$
\begin{aligned}
\prod_{j\in L}\tau_j^{v_j} &= \prod_{j\in L}[\eta^{\pi_j}h(C_{j1},C_{j2},Q_j)]^{v_j}\\
&= \eta^{\sum_{j\in L}\pi_j v_j}\cdot\prod_{j\in L}h(C_{j1},C_{j2},Q_j)^{v_j}\\
&= \eta^{\sum_{j\in L}(\sum_{l=1}^{s}a_l m_{jl}+b_j)v_j}\cdot\prod_{j\in L}h(C_{j1},C_{j2},Q_j)^{v_j}\\
&= \eta^{\sum_{l=1}^{s}a_l(u_l-\gamma_l)+\sum_{j\in L}b_j v_j}\cdot\prod_{j\in L}h(C_{j1},C_{j2},Q_j)^{v_j}\\
&= \eta^{\varpi}\cdot\prod_{l=1}^{s}\eta_l^{a_l}\cdot\prod_{j\in L}h(C_{j1},C_{j2},Q_j)^{v_j}
\end{aligned}
$$

From the above two equations, we note that the group signatures for batch auditing satisfy the correctness criterion.

$$
\begin{aligned}
e(\Theta,g_2) &= e\left(\prod_{j\in L}\theta_j^{v_j},g_2\right)\\
&= e\left(\prod_{j\in L}[H(fn\parallel j)g_1^{\pi_j}]^{\xi v_j},g_2\right)\\
&= e\left(\prod_{j\in L}H(fn\parallel j)^{v_j}\cdot\prod_{j\in L}g_1^{\pi_j v_j},g_2^{\xi}\right)\\
&= e\left(\prod_{j\in L}H(fn\parallel j)^{v_j}\cdot\prod_{l=1}^{s}g_1^{a_l\mu_l-a_l\gamma_l+\sum_{j\in L}b_j v_j},X\right)\\
&= e\left(\prod_{j\in L}H(fn\parallel j)^{v_j}\cdot\prod_{l=1}^{s}g_1^{-a_l\gamma_l}\cdot g_1^{\sum_{l=1}^{s}a_l\mu_l+\sum_{j\in L}b_j v_j},X\right)\\
&= e\left(\prod_{j\in L}H(fn\parallel j)^{v_j}\cdot\prod_{l=1}^{s}\delta_l^{a_l}\cdot g_1^{\varpi},X\right)
\end{aligned}
$$

Thus, Eq. (3) also satisfies the correctness criterion. It proves that, as long as the data maintains integrity, the auditing proof generated over the original data can pass the TPA's verification.

Moreover, for Eq. (4) in the **Open** procedure, we have

$$\begin{aligned} C_{j2}/C_{j1}^{t} &= \left(Z_i^{x_i} \cdot T^{y_j}\right)/C_{j1}^{t} \\ &= \left(Z_i^{x_i} \cdot g_1^{y_j t}\right)/g_1^{y_j t} \\ &= Z_i^{x_i} \end{aligned}$$

Thus, all valid signatures can be opened correctly by the group manager.

The correctness of the value $\widetilde{Q}_j$ is based on the proof of Eq. (1), and Eq. (5) can be proved as follows:

$$\begin{aligned} & e(C_{j2}, U) \cdot e(g_1^x, g_2) \\ &\quad = e\left(Z_i^x \cdot T^{y_j}, U\right) \cdot e\left(g_1^x, g_2\right) \\ &\quad = e\left(g_1^{z_i x}, g_2^u\right) \cdot e\left(T^{y_j}, U\right) \cdot e\left(g_1^x, g_2\right) \\ &\quad = e(g_1, g_2)^{z_i x u + x} \cdot e(T^{y_j}, U) \\ &\quad = e(D, g_2) \cdot \widetilde{Q}_j \end{aligned}$$

□

5.2 Security analysis

Theorem 2 *It is computationally infeasible for a semi-trusted cloud server or an external adversary to forge a valid signature for any data block if the CDH assumption holds.*

Proof To prove unforgeability, we first assume that $\mathcal{F}$, which can adaptively choose data blocks and identities, is able to generate a forged group signature with the advantage of ε, and it can only execute at most q_H hash queries and q_s sign queries within time qt. Then ,we can construct a challenger $\mathcal{C}$ that can break the CDH assumption within time qt' and advantage ε', where

$$\begin{aligned} qt' &\leq qt + (q_H + q_s + 1) \cdot T_{G_1} \\ \varepsilon' &\geq \frac{\varepsilon}{e \cdot (1 + q_s)} \end{aligned}$$

Let us assume that $\mathcal{C}$ has broken the homomorphic MAC algorithm. That is, given a message m_j, $\mathcal{C}$ can generate a valid homomorphic MAC π_j. Then, it further implements a challenge-response game to break the group signature for algorithm $\mathcal{F}$ as follows:

Game1 Given $(g_1, g_1^{\alpha}, g_1^{\beta})$ as a challenge, $\mathcal{F}$ uses it to set a target public key $PK = g_1^{\alpha}$. Let Q_{Hash}, Q_{sign} denote the hash queries and sign queries, respectively, for which $\mathcal{C}$ is supposed to return valid responses. we note that the result of hash queries in this game is $H(fn \parallel j)g_1^{\pi_j}$ on the query of (j, m_j), where π_j is the homomorphic MAC of block m_j.

Q_{Hash} Given a coin $coin \in \{0, 1\}$, the probability for $\mathcal{C}$ to select a coin by $coin = 0$ is $Pr[0] = \frac{q_s}{1+q_s}$. $\mathcal{C}$ maintains a list L_H to record the hash queries. For each hash query (j, m_j) from $\mathcal{F}$, $\mathcal{C}$ first checks whether the entry has been in L_H, if so, outputs corresponding result to $\mathcal{F}$ directly. Otherwise, $\mathcal{C}$ first flips the coin to select the value of $coin$. If $coin = 0$, it outputs the result as $h_j^* = g_1^{\beta}$ to $\mathcal{F}$. Otherwise, it chooses a random element $r \in Z_q^*$, and outputs the result as $h_j = g_1^r$ to $\mathcal{F}$, and records the result in L_H, where r is a distinct value for the current query with index j.

Q_{sign} $\mathcal{C}$ maintains a list L_S to record the sign queries. If a sign query (j, m_j) has been recorded in L_S, $\mathcal{C}$ outputs the existed result to $\mathcal{F}$. Otherwise, $\mathcal{C}$ flips the coin, if $coin = 0$, it implies a failure and it aborts this game. Otherwise, by searching the hash query list L_H (we assume that a hash query has been issued on this block), it outputs a signature $\theta_j = (g_1^{\alpha})^r = h_j^{\alpha} = g_1^{r\cdot\alpha}$ to $\mathcal{F}$.

Forgery $\mathcal{F}$ generates a forgery (j^*, m_j^*, θ^*). In accordance with the above random oracle queries, if $coin \neq 0$, $\mathcal{C}$ fails to guess the target (j^*, m_j^*) that $\mathcal{F}$ wants to implement a forgery attack. Otherwise, with the result of the hash query on (j^*, m_j^*), $\mathcal{C}$ can compute a signature $\theta_j^* = (h^*)^{\alpha} = g_1^{\alpha\cdot\beta}$, which is considered as the result of the CDH problem.

Thus, $\mathcal{C}$ can break the CDH assumption if $\mathcal{F}$ can generate a forgery successfully. Additionally, the probability that $\mathcal{C}$ wins the game is $Pr[0]^{q_s} \cdot (1 - Pr[0]) \cdot \varepsilon = \frac{\varepsilon}{e\cdot(1+q_s)}$, where $e = \lim_{q_s \to +\infty}(1 + \frac{1}{q_s})^{q_s}$ is a constant. For each hash query or sign query of $\mathcal{F}$, $\mathcal{C}$ requires one exponentiation operation on G_1. For the last forgery, it requires an exponentiation operation as well. So, the whole time for $\mathcal{C}$ to break the CDH assumption is at most $qt' = qt + (q_H + q_s + 1) \cdot T_{G_1}$, where T_{G_1} is the running time of one exponentiation operation.

However, it is computationally infeasible to compute a result for CDH problem if the CDH assumption holds. It is also worth mentioning that the probability for an algorithm $\mathcal{F}$ to break a homomorphic MAC on any data block is $1/q$ [36], which is a significant challenge. Therefore, a semi-trusted cloud server or an external adversary cannot forge a signature under the proposed scheme. □

Theorem 3 *It is computationally infeasible for a semi-trusted cloud server or an external adversary to generate an auditing proof with corrupted data that can pass the verification of an auditor under the proposed auditing scheme.*

Proof Based on Theorem 2, it is rather hard for a semi-trusted cloud server to forge a group signature on a data block. Besides, to prove the unforgeability of auditing proof, we also define a challenge-response game (named *Game2*) between the TPA and a semi-trusted cloud server. If the cloud server wins *Game2* by generating a forged auditing proof with corrupted data block, and enabling the proof to pass the TPA's verification, we can break the ECDL assumption on group G_1. Similar to the game defined in previous works [37,38], *Game2* can be described as follows:

Game2 The TPA generates an auditing challenge $chal = \{j, v_j\}_{j\in L}$ on shared data M and sends it to the cloud server. If all these data blocks are not corrupted, the correct auditing proof is $pf = \{\Theta, \vec{\mu}, \Omega, \vec{\Upsilon}\}$. If we assume that some sampled data blocks have been corrupted, the cloud server has to generate a forged proof $pf' = \{\Theta, \vec{\mu}', \Omega, \vec{\Upsilon}\}$, where $\vec{\mu}' = \{\mu_l'\}_{1\leq l\leq s}$, $\mu_l' = \sum_{j\in L} v_j m_{jl}' + \gamma_l$. Since

Theorem 2 has proved that the signatures of data blocks cannot be forged, the value $\{\Theta, \Omega\}$ can not be a forgery. In addition, as some data blocks are corrupted, there is at least one element in $\{\Delta\mu_l\}_{1\le l\le s}$, where $\Delta\mu_l = \mu_l - \mu_l{}'$.

According to Eq. (3), a correct auditing proof satisfies

$$e(\Theta, g_2) = e\left(\prod_{j\in L} H(fn \parallel j)^{v_j} \cdot \prod_{l=1}^{s} \delta_l^{a_l} \cdot g_1^{\varpi}, X\right)$$

Now, we assume that if the forged auditing proof can also pass the TPA's verification, then it satisfies

$$e(\Theta, g_2) = e\left(\prod_{j\in L} H(fn \parallel j)^{v_j} \cdot \prod_{l=1}^{s} \delta_l^{a_l} \cdot g_1^{\varpi'}, X\right)$$

where $\varpi' = \sum_{l=1}^{s} a_l\mu_l{}' + \sum_{j\in L} v_j b_j$.

From the above two equations, we can get

$$\begin{aligned} &g_1^{\varpi'} = g_1^{\varpi} \Rightarrow g_1^{\sum_{l=1}^{s} a_l\mu_l{}' + \sum_{j\in L} v_j b_j} = g_1^{\sum_{l=1}^{s} a_l\mu_l + \sum_{j\in L} v_j b_j} \\ &g_1^{\sum_{l=1}^{s} a_l\mu_l{}'} = g_1^{\sum_{l=1}^{s} a_l\mu_l} \Rightarrow g_1^{\sum_{l=1}^{s} a_l\Delta\mu_l} = 1 \\ &1 = \prod_{l=1}^{s} (g_1^{a_l})^{\Delta\mu_l} \end{aligned} \tag{6}$$

Give two elements $h, g \in G_1$, because G_1 is a cyclic group, there always exists a number x that satisfies $h = g^x$. Then, we show how to compute the value of x as follows:

Let $g_1^{a_l} = g^{\kappa_l} h^{\rho_l}$, where κ_l, ρ_l is randomly chosen from Z_q. Then, Eq. (5) can be transformed into

$$\begin{aligned} 1 &= \prod_{l=1}^{s} (g_1^{a_l})^{\Delta\mu_l} = \prod_{l=1}^{s} (g^{\kappa_l} h^{\rho_l})^{\Delta\mu_l} \\ &= g^{\sum_{l=1}^{s} \kappa_l\cdot\Delta\mu_l} h^{\sum_{l=1}^{s} \rho_l\cdot\Delta\mu_l} \\ &= g^{\sum_{l=1}^{s} \kappa_l\cdot\Delta\mu_l + x\cdot\sum_{l=1}^{s} \rho_l\cdot\Delta\mu_l} \end{aligned}$$

We can get the following result:

$$x = -\frac{\sum_{l=1}^{s} \kappa_l \cdot \Delta\mu_l}{\sum_{l=1}^{s} \rho_l \cdot \Delta\mu_l}$$

As it is defined that at least one element is nonzero, and ρ_l is a random element of Z_q, so the probability for $\sum_{l=1}^{s} \rho_l \cdot \Delta\mu_l = 0$ is $1/q$, which can be neglected. Thus, the value obtained in Eq. (5) is meaningful, and we therefore can conclude that if the cloud server wins *Game2* by passing the verification with a forged proof, we can solve the ECDL problem with the probability of $1 - 1/q$, which contradicts the ECDL assumption. Therefore, the semi-trusted cloud server cannot forge an auditing proof to pass the TPA's verification when the data blocks are not stored correctly under the proposed scheme. □

Theorem 4 *The TPA cannot get any knowledge of the original data from the auditing proof under the proposed scheme if the ECDL assumption holds.*

Proof The authorized but curious TPA can only get three kinds of information: a shared key pair $a \in K_{prg}, b \in K_{prf}$, the auditing challenge $\{j, v_j\}_{j \in L}$ and the corresponding proof $pf = \{\Theta, \vec{\mu}, \Omega, \vec{\Upsilon}\}$. As the shared key pair and auditing challenge do not contain any message about the original data, we can only prove that the auditing proof does not leak the data privacy based on the ECDL assumption as follows:

For the element $\Omega = \{C_{j1}, C_{j2}, Q_j, \tau_j, w_j\}_{j \in L} \in pf$, only the elements $\tau_j = \eta^{\pi_j} \cdot h(C_j 1, C_{j2}, Q_j)$ and $w_j = y_j \cdot \tau_j + x_i$ refer to the original data, where $\pi_j = \sum_{l=1}^{s} a_l m_{jl} + b_j$. Although the TPA can get the value of a_l, b_j and $h(C_j 1, C_{j2}, Q_j)$, it still cannot compute the value of π_j if the ECDL assumption holds. Thus, it is not able to get m_{jl} from τ_j. As the TPA cannot get the original data from τ_j, it therefore cannot obtain the value of m_{jl} from w_j, which is further blinded by secret keys x_i and y_j.

For the element $\vec{\mu} = \{\mu_l\}_{1 \le l \le s} \in pf$, we have $\mu_l = \sum_{j \in L} v_j m_{jl} + \gamma_l$. From the previous work (i.e., [25,30,39]), the TPA can recover the original data from $\sum_{j \in L} v_j m_{jl}$, while we blind it with a random element $\gamma_l \in Z_q$ in the proposed scheme. Although the TPA can get $\vec{\Upsilon} = \{\delta_l, \eta_l\}_{1 \le l \le s}$, it cannot get the value of γ_l from $g_1^{\gamma_l}$ or η^{γ_l} if the ECDL assumption holds.

For the element $\Theta \in pf$, the formula $\Theta = \prod_{j \in L} \theta_j^{v_j}$ can be transformed as follows:

$$
\begin{aligned}
\Theta &= \prod_{j \in L} H(fn \parallel j)^{v_j \xi} \cdot \prod_{j \in L} g_1^{\pi_j v_j \xi} \\
&= \prod_{j \in L} H(fn \parallel j)^{v_j \xi} \cdot \prod_{j \in L} g_1^{(\sum_{l=1}^{s} a_l m_{jl} + b_j) v_j \xi} \\
&= \prod_{j \in L} H(fn \parallel j)^{v_j \xi} \cdot \prod_{j \in L} g_1^{b_j v_j \xi} \cdot \prod_{j \in L} g_1^{(\sum_{l=1}^{s} a_l m_{jl}) v_j \xi} \\
&= \prod_{j \in L} H(fn \parallel j)^{v_j \xi} \cdot \prod_{j \in L} g_1^{b_j v_j \xi} \cdot g_1^{\xi \sum_{j \in L} (v_j \sum_{l=1}^{s} a_l m_{jl})}
\end{aligned}
$$

From the above equation, the value of $g_1^{\xi \sum_{j \in L} (v_j \sum_{l=1}^{s} a_l m_{jl})}$ is blinded by $\prod_{j \in L} H(fn \parallel j)^{v_j \xi}$ and $g_1^{b_j v_j \xi}$. Given $X = g_2^{\xi}$, H_j, v_j and b_j, it is impossible for the TPA to get the value of their product based on the CDH assumption, especially when there is no efficiently computable isomorphism from g_1 to g_2. Besides, the TPA still cannot get the value of $\sum_{l=1}^{s} a_l m_{jl}$ from $g_1^{\xi \sum_{j \in L} (v_j \sum_{l=1}^{s} a_l m_{jl})}$ based on the ECDL assumption. Therefore, the data privacy is guaranteed under the proposed scheme. □

Theorem 5 *The TPA and cloud server cannot know the group user's identity under the proposed scheme if ElGamal encryption is secure.*

Proof Since the proposed auditing scheme is developed from a simple-yet-efficient group signature scheme [6], and the corresponding block signature $(\Theta, \vec{\mu}, \vec{\Upsilon})$ does not contain any identity information, so that the proof of anonymity is followed with that described in [6]. Here, we show that if an algorithm $\mathcal{F}$ can disclose the identity anonymity of the proposed auditing scheme, there exists an algorithm $\mathcal{B}$ that can break ElGamal encryption. $\mathcal{B}$ first selects a random element $t \in Z_q^*$ to compute $T = g_1^t$ as a public key of ElGamal encryption. Then, $\mathcal{B}$ computes $D = g_1^d$, $U = g_2^u$, where d, u is randomly selected from Z_q^* as a group private key gsk. For each user i, $\mathcal{B}$ computes $Z_i = g_1^{z_i} = g_1^{(d-x_i)(ux_i)^{-1}}$, where x_i, z_i are two random elements in Z_q^*. Finally, $\mathcal{B}$ stores the distinct value of (x_i, Z_i) as i-th user's private key, and outputs the group public key $gpk = (D, U, T, X, \eta)$ to $\mathcal{F}$, where $\eta \in Z_q^*$.

Let us assume that $\mathcal{F}$ can query $\mathcal{B}$ about the random hash oracle $h(\cdot)$ at any time, and $\mathcal{B}$ holds a list *List* about these answers. If $\mathcal{F}$ gives a repeated query, $\mathcal{B}$ can directly respond to it with the existing answer in $List$. Otherwise, responds with a random element chosen from Z_q and stores the answer in *List* for repeated queries.

Now, $\mathcal{F}$ sends two indices, i_0 and i_1, and a homomorphic MAC π to $\mathcal{B}$ as an anonymity challenge. Then, $\mathcal{B}$ sends two values $Z_{i_0}^{x_{i_0}}$ and $Z_{i_1}^{x_{i_1}}$ to $\mathcal{C}$ as an indistinguishability challenge, where $\mathcal{C}$ denotes an ElGamal encryption challenger. Upon receiving the indistinguishability challenge, $\mathcal{C}$ chooses one of two values to encrypt, i.e., $C_{j1} = g_1^{y_j}$, $C_{j2} = Z_{i_b}^{x_{i_b}} \cdot T^{y_j}$, where b is a bit from $\{0, 1\}$, $y_j \in Z_q^*$.

$\mathcal{B}$ needs to determine which value is encrypted by $\mathcal{C}$ as follows: It chooses two elements $\tau_j, w_j \in Z_q$ for computing $Q_j = \frac{e(C_{j2},U)\cdot e(g_1,g_2)^{w_j}}{e(D\cdot C_{j1}^{\tau_j},g_2)}$, where τ_j is a multiple of η^{π_j}. For the tuple $(\eta, \tau_j, C_{j1}, C_{j2}, Q_j)$, if the random hash oracle $h(\cdot)$ has set some other value instead of c, there is a collision to one-way hash function, which implies a failure.

Since the ciphertext of $Z_{i_b}^{x_{i_b}}$ is embedded into a valid signature σ of the data block by group user i_b, $\mathcal{B}$ can respond to its indistinguishability challenge by calling $\mathcal{F}$'s response. In other words, if $\mathcal{F}$ can break the identity anonymity of the proposed scheme, $\mathcal{B}$ can break the ElGamal encryption, which contradicts with the semantic security of the ElGamal encryption.

Therefore, the identity of any group user is anonymous under the proposed auditing scheme. □

Theorem 6 *The group manager can trace the identity of the group user by using the signatures on each data block.*

Proof Given a data block m, the corresponding signature should be $(C_1, C_2, \tau, w, \theta)$ if the user i follows the $GroupSign$ procedure in a right manner. As the group manager owns the group private key t, it can computes the value of $Z_i^{x_i} = C_2/C_1^t$. By checking the table ***Tab***, the group manager can get the multiplication of each group user's private key, and then reveal the identity of the group user who signed on the data block m. □

Table 2 Security comparisons of our proposed scheme and related schemes

	Wang et al.'s scheme [35]	Yuan et al.'s scheme [40]	Fu et al.'s scheme [41]	Our proposed scheme
SR_1	✓	✓	✓	✓
SR_2	✓	×	✓	✓
SR_3	×	×	×	✓
SR_4	✓	×	✓	✓
SR_5	✓	×	✓	✓
SR_6	×	✓	✓	✓

SR_1: The requirement of public auditing
SR_2: The requirement of authorized auditing
SR_3: The requirement of data privacy
SR_4: The requirement of identity privacy
SR_5: The requirement of identity traceability
SR_6: The requirement of user revocation
✓: The requirement is satisfied
×: The requirement is not satisfied

5.3 Security comparison

In order to highlight the security benefits of our proposed scheme, we compare it with a few recently proposed approaches, such as Wang et al.'s scheme [35], Yuan et al.'s scheme [40] and Fu et al's scheme [41]. For sake of simplicity, let SR_1, SR_2, SR_3, SR_4, SR_5 and SR_6 represent the security requirements in terms of public auditing, authorized auditing, data privacy, identity privacy, identity traceability and user revocation.

Table 2 shows that, none of the previously proposed related schemes protect data privacy, because the curious TPA can recover the data content from the value of $\mu = \sum_{j \in L} v_j m_j$. Due to the usage of group signature, the scheme [35,41] and our proposed scheme can achieve identity privacy and traceability, but the scheme [35]cannot provide user revocation. Although Yuan et al.'s scheme [40] can provide user revocation, it cannot provide identity privacy and traceability for shared data in the cloud.

6 Performance evaluation

In this section, we first analyze the performance of our proposed scheme in terms of computation and communication costs based on an asymmetric bilinear map groups of Type-3 [42]. Then, we demonstrate the efficiency by implementing practical experiments. We also compare the performance of our proposed scheme with another recent scheme [41]. According to [43] and Table 3 referred in [44], for most mathematical operations, the bilinear map groups of type-3 is more efficient than other bilinear map groups of Type-1 when they are at 80-bit security level, so that the following comparison results are based on the bilinear map groups of Type-3 at the same security level.

6.1 Performance analysis

For the sake of simplicity, we employ some notations to describe the mathematical operations used in Fu et al.'s scheme [41] and ours. They are defined as: $E_{G_1}, E_{G_2}, E_{G_T}$ and E_{Z_q} denote one exponentiation operation in group G_1, G_2, G_T and Z_q, respectively, and $M_{G_1}, M_{G_T}, M_{Z_q}$ denote one multiplication operation in group G_1, G_T and Z_q, respectively. D_{Z_q}, D_{G_T} represents one division operation in group Z_q and G_T; A_{Z_q} denotes one addition operation in group Z_q. Hs, hs denote one hash-to-point operation on group G_1 and one hash-to-integer operation in group Z_q, respectively. $P_{1,2}$ represents one bilinear pairing operation. Assume that the number of sampled blocks to verify is c, and each block is split into s sectors. (The advantages of partitioning data blocks are given in [39].) Since the data block in Fu et al.'s scheme is not split into s sectors, we set $s = 1$ to get fair comparison results, and so that we can compare Fu et al's scheme with our proposed scheme.

(1) Analysis on communication cost:

Since uploading the data file and the corresponding verification information is a one-time operation, therefore we do not analyze the communication cost between each group user and the remote cloud server. But the communication cost incurred by the auditing challenge and proof cannot be ignored.

For every **ProofGen** phase, the auditing challenge $chal = \{(j, v_j), fn\}$ costs $c(|j| + |q|) + |fn|$ bits, where $|j|$ is the size of a block index j, $|q|$ is the size of an element chosen from group Z_q, and $|fn|$ is the size of file identifier. For the auditing proof $pf = \{\Theta, \vec{\mu}, \Omega, \vec{\Upsilon}\}$, it consumes $(2c + 1)|G_1| + c|G_T| + (2c + 2s)|q|$ bits, where $|G_1|$, $|G_T|$ denote the sizes of an element in group G_1 and G_T, respectively. Thus, the total communication cost is $(2c+1)|G_1|+c|G_T|+(3c+2)|q|+c|j|+|fn|$ bits due to $s = 1$. The total communication cost of Fu et al.'s scheme [41] is $|G_1| + (16c + 1)|q| + c(|j| + |id_j|)$ bits, so that the communication cost of their scheme is higher than ours by $(14c - 1)|q| - 2c|G_1| - c|G_T| + (c - 1)|id_j|$ bits (where $|fn|$ is equal to $|id_j|$ and $c \geq 100$ in practical experiments, and the representation sizes of those mentioned elements can be found in [44]). Our proposed scheme is therefore much more efficient than Fu et al.'s scheme in terms of communication cost.

(2) Analysis on computation cost:

For our proposed scheme, the computation cost is mainly caused by the **ProofGen** phase and the **ProofVerify** phase. To create an auditing proof, the computational operations of the cloud are $(c+s)E_{G_1} + cM_{G_1} + sE_{Z_q} + csM_{Z_q} + (cs+s)A_{Z_q}$. After receiving the cloud's proof, the TPA has to execute $E_{G_T} + (c + 1)M_{G_T} + D_{G_T} + (2c + s + 2)E_{G_1} + (3c + s + 3)M_{G_1} + (2c + s + 1)E_{Z_q} + (3c + 2s + 2)M_{Z_q} + (c + s)A_{Z_q} + 4P_{1,2} + c(hs + Hs)$ computational operations to finish the auditing proof verification, where $s = 1$.

For Fu et al.'s scheme [41], the cloud only needs $c(E_{G_1} + M_{G_1} + M_{Z_q} + A_{Z_q})$ operations to generate an auditing proof. To verify it, there are $(2c + 1)E_{G_1} + (21c + 2)M_{G_1} + (22c+2)E_{Z_q} + (11c+2)M_{Z_q} + (3c)D_{Z_q} + (7c)A_{Z_q} + 2P_{1,2} + c(2hs + Hs)$

operations for TPA to execute. As we can see, there are many exponentiation operations and division operations in Z_q for Fu et al.'s scheme during the auditing phase which cause it to incur high computation overheads.

Based on the comparison, at the cloud's side, the computation cost of our proposed scheme is higher than that in Fu et al.'s by $E_{G_1} + E_{Z_q} + A_{Z_q}$, which is used to blind the sampled blocks in order to protect data privacy. As a matter of fact, it is insignificant to include the additional operations for a cloud because of its strong computing capability. However, at the TPA's side, the computation cost of our proposed scheme is a little lower than that of Fu et al.'s by $(20c)E_{Z_q} + (18c-2)M_{Z_q} + (3c)D_{Z_q} + (6c-1)A_{Z_q} - D_{G_T} - (c+1)M_{G_T} - E_{G_T} - 2E_G - 2P_{1,2} + c \cdot hs$. Therefore, our proposed scheme is lightly more efficient than Fu et al.'s scheme in theory, and we demonstrate it in the following practical experiments.

(3) Analysis on signature size:

The efficiency of our proposed scheme is based on the efficiency of group signature, which only contains five elements for each data block, e.g., $\sigma_j = (C_{j1}, C_{j2}, \tau_j, \omega_j, \theta_j)$. Thus, the signature size is $n(3|G_1| + 2|Z_q|)$ if there are n data blocks in a file. While for Fu et al.'s scheme [41], there are fifteen elements consisting of a data block's signature, and it costs $n(|G_1| + 14|Z_q|)$ bits to create the signatures for a whole file. Thus, on the premise of $s = 1$, the signature size of Fu et al.'s scheme is larger than ours by $n(12|Z_q| - 2|G_1|)$ with the same file. In fact, by referring to [39], the value of s is greater, the number of blocks n is lower which means the less bandwidth we need to transfer the signatures to a remote cloud server and also need less time we need to generate the signatures for a file.

6.2 Experimental results

In this subsection, we evaluate the performance of our proposed scheme from the view of practical experiments. We use the free Mircal Library, written in C, to implement the cryptographic operations on Windows system with an Intel(R) Core(TM) i7-6700 CPU at 3.40 GHz and equipped with 8GB of RAM. To achieve the bilinear operations (referred to the R-ate pairing [45]), the elliptic curve we selected is a MNT curve at 80-bit security level, whose base field size is 159 bits, and the embedding degree is 6. Therefore, the size of element chosen from Z_q^* is 160 bits. Let us assume that the number of sampled blocks to be audited is c, and the number of corrupted data blocks in the shared data is t. Let n be the number of corrupted blocks in the sampled data blocks, the probability P for an auditor to detect the corrupted blocks can be computed as: $P = P(X \geq 1) = 1 - P(X = 0) = 1 - \frac{C_{n-t}^c}{C_n^c} = 1 - \frac{(n-t)(n-t-1)\cdots(n-t-c+1)}{n(n-1)\cdots(n-c+1)}$, which can be converted to the inequality: $1 - (\frac{n-t}{n})^c \leq P \leq (\frac{n-t-c+1}{n-c+1})^c$. For instance, if 1% data blocks are corrupted, the auditor can detect the corruption with a probability of 95% with 300 sampled blocks, and 99% with 460 sampled blocks.

Similar to the performance analysis, we also compare our proposed scheme with [41] in terms of computation cost and communication cost. In our experiments, the size of each sector in our proposed scheme (s sectors consist of a data block) is set to be 160 bits, so that the size of each block is about $160 \times s$ bits. For simplicity and fair

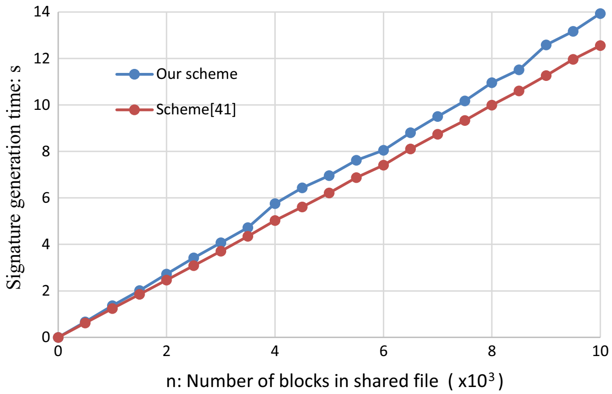

Fig. 3 Time of signature generation with different number of blocks

comparison, we still set the number of sector as $s = 1$. All our experimental results are the average of 50 trials.

6.2.1 Experiments on computation cost

Firstly, to demonstrate the efficiency of signature generation, we evaluate the execution time of the **GroupSign** phases in our proposed scheme and Fu et al.'s scheme [41]. Figure 3 shows the experimental results. The total number of data blocks is set to be $n = 10000$, and the execution time of generating signatures is nearly linear with the number of data blocks. The result shows that Fu et al.'s scheme is slightly more efficient than ours at the data user side, but the operations for generating signatures on blocks can be viewed as a one-time phase, rather than a frequent phase (e.g., auditing phase). Therefore, our proposed scheme is acceptable for practical applications as well because it is also efficient to some degree.

As the auditing efficiency is mainly affected by proof generation and proof verification, we evaluate the execution time of the **ProofGen** phase on the cloud's side and the **ProofVerify** phase on the TPA's side, respectively. As shown in Fig. 4, the execution time of our scheme for generating auditing proofs is slightly higher than Fu et al.'s scheme because of the additional exponential operations, which are used to blind the original data blocks for improving auditing security. As a matter of fact, the extra computation cost can be negligible in real applications because of the powerful computational capability of the cloud server. For the execution time of proof verification shown in Fig. 5, we note that the efficiency of our proposed scheme is higher than that of Fu et al.'s scheme. In particular, to generate the auditing proofs, it takes around 0.010s and 0.011s for our proposed scheme and the other scheme, respectively, when the number of block is $c = 100$. The execution time increase to around 0.056s (with our scheme) and 0.057s (with the other scheme) when the number of block increases to $c = 500$. To verify the auditing proofs when $c = 100$ and $c = 500$, our proposed scheme only needs 0.098s and 0.432s, respectively, whereas Fu et al.'s scheme needs 0.155s and 0.733s, respectively. Moreover, to satisfy the detection probability

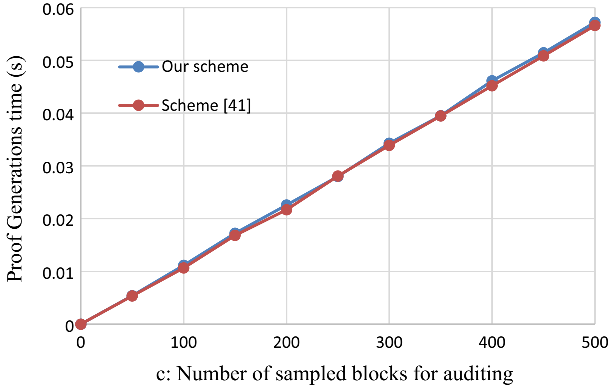

Fig. 4 Time of auditing proof generation with different numbers of blocks

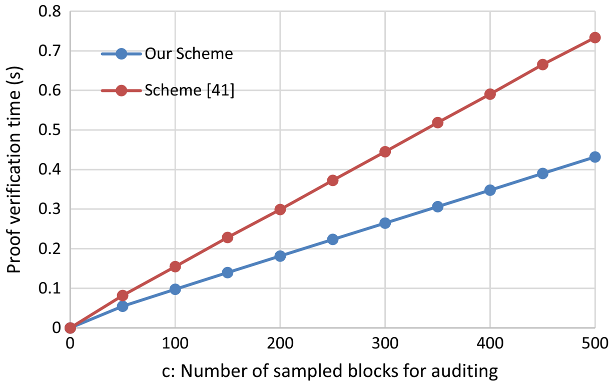

Fig. 5 Time of auditing proof verification with different block numbers

of corruption, we also test the execution time of proof verification when the number of sampled blocks is 300 or 460. The experimental results show that it takes 0.445s and 0.671s for Fu et al's scheme when $c = 300$ and $c = 460$. To finish the same auditing tasks, it only takes 0.265s and 0.392s for our proposed scheme, both of which improve the efficiency by about 41%. Therefore, our proposed scheme is more efficient in proof auditing while improving the security of that scheme at the same time.

6.2.2 Experiments on communication cost

When the data owner/users outsource(s) their data file and signatures on blocks to the cloud server, the communication cost must be considered. As we analyzed in the subsection on signature size earlier, our proposed scheme incurs lower communication cost than that in scheme [41]. Here, we would not compare the communication cost of this procedure because it can be considered as a one-time process for an outsourced file. For the auditing challenge message $chal$, since the extra communication cost produced by fn is not more than 4 bytes in our experiments, the communication cost

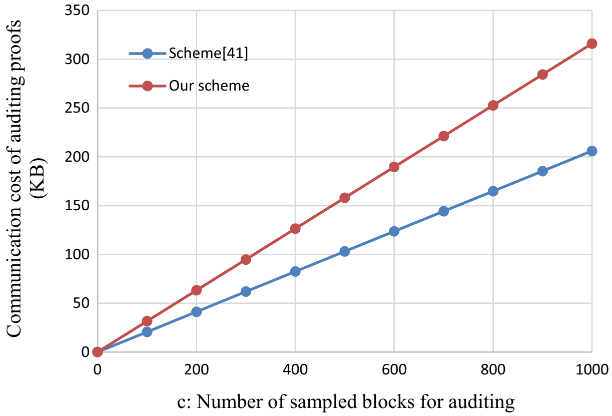

Fig. 6 Time of auditing proof verification with different numbers of blocks

Table 3 Comparisons of computation and communication costs for $c = 300$ and $c = 460$

	$c = 300$		$c = 460$	
	[41]	Ours	[41]	Ours
Proof generation(s)	0.034	0.035	0.050	0.052
Proof verification(s)	0.445	0.265	0.671	0.392
Challenge message (KB)	7.031	7.035	10.781	10.785
Auditing proof (KB)	97.71	54.75	134.61	84.05

in our proposed is very close to than in Fu et al's scheme. As for the communication costs (shown in Fig. 6) produced by the auditing proof, it increases when the number of sampled blocks grows in the auditing procedure. Fortunately, the number of blocks $c = 460$ is enough for implementing the auditing tasks. Additionally, Fig. 6 demonstrates that our proposed scheme incurs lower communication than Fu et al.'s scheme [41]. To present the costs clearly, we list the computation and communication costs for $c = 300$ and $c = 460$, respectively, in Table 3.

7 Conclusion

Cloud storage has emerged as a promising solution to the management problem of massive amount of data generated in the big data era. To save costs, an application can share the data among a large number of users working in the same group. To guarantee the data integrity, many public auditing schemes have been proposed to check the data storage correctness. However, it remains a significant challenge to design an efficient public auditing scheme for shared data while simultaneously preserving identity privacy.

In this paper, we propose an efficient privacy-preserving public auditing scheme for shared data on a remote cloud server. More concretely, we utilize efficient group signature and homomorphic authenticators to compute the verification information for each data block, so that an authorized TPA can efficiently verify the integrity of data, but cannot know the identity of signer on the sampled blocks. Moreover, with the properties of group signature, the group manager can reveal and revoke any group user if he/she behaves badly. We also utilize the random masking technique to keep data privacy against the TPA. Our security analysis demonstrates that the proposed scheme satisfies all the security requirements for public data auditing. Moreover, the performance evaluation shows that the proposed scheme achieves the property of auditing efficiency as well.

As a part of our future work, we will design a more efficient public auditing scheme for shared data with constant verification time, while achieving both high security and optimal performance.

Acknowledgements We greatly appreciate the invaluable suggestions provided by the anonymous reviewers and the associate editor. The work was supported in part by the National Natural Science Foundation of China under (Nos. 61501333, 61772377, 61572379, 61572370), and in part by the Natural Science Foundation of Hubei Province of China (Nos. 2017CFA007, 2015CFA068).

References

1. Selvaraj A, Sundararajan S (2015) Survey on public auditability to ensure data integrity in cloud storage. Int J Comput Appl 37(3–4):102–110
2. Garg N, Bawa S (2016) Comparative analysis of cloud data integrity auditing protocols. J Netw Comput Appl 66:17–32
3. El-Dein RE, Youssef B, ElGamal S (2016) Content auditing in the cloud environment. Data Min Knowl Eng 8(10):311–317
4. Cisco Global Cloud Index Cisco (2014) Forecast and methodology, 2013–2018. Cited on, page 23
5. Xue L, Ni J, Li Y, Shen J (2017) Provable data transfer from provable data possession and deletion in cloud storage. Comput Stand Interfaces 54:46–54
6. Ho T-H, Yen L-H, Tseng C-C (2015) Simple-yet-efficient construction and revocation of group signatures. Int J Found Comput Sci 26(5):611–624
7. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D (2007) Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. ACM, pp 598–609
8. Shacham H, Waters B (2008) Compact proofs of retrievability. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer, pp 90–107
9. Boneh D, Lynn B, Shacham H (2004) Short signatures from the weil pairing. J Cryptol 17(4):297–319
10. Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. ACM, pp 62–73
11. Wang C, Chow SSM, Wang Q, Ren K, Lou W (2013) Privacy-preserving public auditing for secure cloud storage. IEEE Trans Comput 62(2):362–375
12. Liu C, Chen J, Yang LT, Zhang X, Yang C, Ranjan R, Kotagiri R (2014) Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates. IEEE Trans Parallel Distrib Syst 25(9):2234–2244
13. Zhang J, Li P, Mao J (2016) IPad: ID-based public auditing for the outsourced data in the standard model. Clust Comput 19(1):127–138
14. Yang G, Jia Y, Shen W, Qianqian S, Zhangjie F, Hao R (2016) Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability. J Syst Softw 113:130–139
15. Kim D, Jeong IR (2017) Certificateless public auditing protocol with constant verification time. Secur Commun Netw 5:1–14

16. Wang B, Li B, Li H (2015) Panda: public auditing for shared data with efficient user revocation in the cloud. IEEE Trans Serv Comput 8(1):92–106
17. Xu Z, Wu L, Khan MK, Choo K-KR, He D (2017) A secure and efficient public auditing scheme using RSA algorithm for cloud storage. J Supercomput 73:1–25
18. Wang H (2015) Identity-based distributed provable data possession in multicloud storage. IEEE Trans Serv Comput 8(2):328–340
19. Zhang J, Dong Q (2016) Efficient ID-based public auditing for the outsourced data in cloud storage. Inf Sci 343:1–14
20. Yu Y, Zhang Y, Mu Y, Susilo W, Liu H (2015) Provably secure identity based provable data possession. In: International Conference on Provable Security. Springer, pp 310–325
21. Wang H, He D, Tang S (2016) Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud. IEEE Trans Inf Forensics Secur 11(6):1165–1176
22. Li Y, Yu Y, Min G, Susilo W, Ni J, Choo K-KR (2017) Fuzzy identity-based data integrity auditing for reliable cloud storage systems. IEEE Trans Dependable Secure Comput. https://doi.org/10.1109/TDSC.2017.2662216
23. Wang B, Li B, Li H, Li F (2013) Certificateless public auditing for data integrity in the cloud. In: IEEE Conference on Communications and Network Security (CNS). IEEE, pp 136–144
24. Zhang Y, Chunxiang X, Shui Y, Li H, Zhang X (2015) SCLPV: secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors. IEEE Trans Comput Soc Syst 2(4):159–170
25. He D, Zeadally S, Wu L (2018) Certificateless public auditing scheme for cloud-assisted wireless body area networks. IEEE Syst J 12(1):64–73
26. Yang T, Yu B, Wang H, Li J, Lv Z (2015) Cryptanalysis and improvement of panda–public auditing for shared data in cloud and internet of things. Multimed Tools Appl 76:1–18
27. Tang CM, Zhang XJ (2015) A new publicly verifiable data possession on remote storage. J Supercomput 1–15. https://doi.org/10.1007/s11227-015-1556-z
28. Xu Z, Wu L, He D, Khan MK (2017) Security analysis of a publicly verifiable data possession scheme for remote storage. J Supercomput 73(11):4923–4930
29. Li J, Zhang L, Liu JK, Qian H, Dong Z (2016) Privacy-preserving public auditing protocol for low-performance end devices in cloud. IEEE Trans Inf Forensics Secur 11(11):2572–2583
30. Shen W, Jia Y, Xia H, Zhang H, Xiuqing L, Hao R (2017) Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium. J Netw Comput Appl 82:56–64
31. Luo Y, Xu M, Fu S, Wang D, Deng J (2015) Efficient integrity auditing for shared data in the cloud with secure user revocation. In: Trustcom/BigDataSE/ISPA, 2015 IEEE, vol 1, pp 434–442. IEEE
32. Wang H, Li K, Ota K, Shen J (2016) Remote data integrity checking and sharing in cloud-based health internet of things. IEICE Trans Inf Syst 99(8):1966–1973
33. He K, Huang C, Yang K, Shi J (2015) Identity-preserving public auditing for shared cloud data. In: IEEE 23rd International Symposium on Quality of Service (IWQoS). IEEE, pp 159–164
34. Wang B, Li B, Li H (2014) Oruta: privacy-preserving public auditing for shared data in the cloud. IEEE Trans Cloud Comput 2(1):43–56
35. Wang B, Li B, Li H (2012) Knox: privacy-preserving auditing for shared data with large groups in the cloud. In: Bao F, Samarati P, Zhou J (eds) Applied cryptography and network security. Springer, Berlin, pp 507–525
36. Agrawal S, Boneh D (2009) Homomorphic MACs: Mac-based integrity for network coding. In: ACNS, vol 9. Springer, pp 292–305
37. Wu L, Wang J, Kumar N, He D (2017) Secure public data auditing scheme for cloud storage in smart city. Pers Ubiquitous Comput 21(5):949–962
38. Wang H (2013) Proxy provable data possession in public clouds. IEEE Trans Serv Comput 6(4):551–559
39. Li A, Tan S, Jia Y (2016) A method for achieving provable data integrity in cloud computing. J Supercomput 1–17. https://doi.org/10.1007/s11227-015-1598-2
40. Yuan J, Yu S (2014) Efficient public integrity checking for cloud data sharing with multi-user modification. In: INFOCOM, 2014 Proceedings IEEE. IEEE, pp 2121–2129
41. Fu A, Yu S, Zhang Y, Wang H, Huang C (2017) NPP: a new privacy-aware public auditing scheme for cloud data sharing with group users. IEEE Trans Big Data
42. Boyen X (2008) The uber-assumption family. In: International Conference on Pairing-Based Cryptography. Springer, pp 39–56

43. MIRACL Cryptographic Library: Multiprecision Integer and Rational Arithmetic C/C++ Library(MIRACL)
44. Park JH, Lee DH (2016) An efficient IBE scheme with tight security reduction in the random oracle model. Des Codes Cryptog 79(1):63–85
45. Lee E, Lee H-S, Park C-M (2009) Efficient and generalized pairing computation on Abelian varieties. IEEE Trans Inf Theory 55(4):1793–1803