Information Sciences 472 (2019) 107–125

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Public auditing for shared cloud data with efficient and secure group management

Hui Tian[a,*], Fulin Nan[a], Hong Jiang[b], Chin-Chen Chang[c], Jianting Ning[d], Yongfeng Huang[e]

[a] College of Computer Science and Technology, National Huaqiao University, Xiamen 361021, China
[b] Department of Computer Science and Engineering, University of Texas at Arlington, TX 76010, USA
[c] Department of Information and Computer Science, Feng Chia University, Taichung 40724, Taiwan
[d] School of Information Technology, Deakin University, Victoria 3125, Australia
[e] Department of Electrical Engineering, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 11 May 2018
Revised 8 August 2018
Accepted 2 September 2018
Available online 11 September 2018

Keywords:
Cloud storage
Cloud security
Public auditing
Shared cloud data
Lazy revocation
Batch verification

ABSTRACT

With increasing popularity of collaboration in clouds, shared data auditing has become an important issue in cloud auditing field, and attracted extensive attention from the research community. However, none of the state of the arts can fully achieve all indispensable functional and security requirements. Thus, in this paper, we present a comprehensive public auditing scheme for shared data. Specifically, to preserve users' identity privacy, signatures on the challenged blocks are converted to the ones signed by the group manager during proof generation; to protect data privacy, a random masking is adopted to blind data proof; a modification record table is designed to record operation information to support identity traceability; we further extend the dynamic hash table to support shared-data dynamics, and present a batch auditing strategy. Moreover, we design a lazy-revocation based group management mechanism to achieve efficient group dynamics, which can resist collusion attacks while significantly reducing computational costs. We formally prove the security of our scheme, and evaluate its performance by comprehensive experiments and comparisons with the state-of-the-art ones. The results demonstrate that our scheme can effectively achieve secure auditing and outperforms the previous ones in computational overhead while maintaining relatively low communication costs.

1. Introduction

Cloud storage has attracted extensive attention from academic and industrial communities for its huge advantages of costs, performance and management over traditional local storage. As a result, a growing number of organizations and individuals have been migrating their data to the cloud storage that is managed and maintained by professional cloud service providers (CSPs) [6,16,21]. Despite its considerable advantages, there is no denying that the cloud storage also faces a series of security challenges, especially in terms of security and privacy [4,9,15,23,34]. One of the significant concerns is how to determine whether the CSP meets the legal expectations of users for data integrity [14,17,26], for which the reasons

* Corresponding author.
E-mail addresses: htian@hqu.edu.cn (H. Tian), flnan@hqu.edu.cn (F. Nan), hong.jiang@uta.edu (H. Jiang), alan3c@gmail.com (C.-C. Chang), jtning88@gmail.com (J. Ning), yfhuang@tsinghua.edu.cn (Y. Huang).

https://doi.org/10.1016/j.ins.2018.09.009
0020-0255/

Table 1
Comparison of communication costs.

Schemes	Data dynamics	Privacy preservation		Group dynamics		Collusion attack resistance	Traceability	Batch auditing
		Data privacy	Identity privacy	User enrollment	User revocation			
HHY15 [11]	◯	√	√	$\mathcal{K}$(III)	×	×	×	√
JCM16 [12]	◯	◯	×	◯	$\mathcal{T}$(IV)	√	√	◯
Oruta [30]	√	√	√	$\mathcal{K}$(I),$\mathcal{T}$(I)	$\mathcal{K}$(I),$\mathcal{T}$(I)	–	×	√
Panda [31]	√	×	×	◯	$\mathcal{T}$(III)	×	×	√
YYS16 [36]	◯	×	√	$\mathcal{K}$(II)	×	×	√	◯
SDIVIP2 [39]	◯	×	√	$\mathcal{K}$(IV),$\mathcal{T}$(II)	$\mathcal{K}$(IV),$\mathcal{T}$(II)	√	×	◯
YY14 [40]	◯	√	×	◯	$\mathcal{T}$(III)	√	×	√
YY15 [41]	◯	√	×	◯	$\mathcal{T}$(III)	×	×	√
PASCD	√	√	√	$\mathcal{K}$(III)	$\mathcal{T}$(V)	√	√	√

Note: "√" means "support"; "×" means "not support"; "–" means "no demand"; and "◯" means "not mentioned". PASCD is the acronym of the proposed scheme in this paper. $\mathcal{K}$(I) means that all users keys need to be regenerated; $\mathcal{K}$(II) means that there should be a key negotiation process between the group manager and the new user; $\mathcal{K}$(III) means that the new user needs to obtain his/her key pairs from the PKI; $\mathcal{K}$(IV) means that the shared keys should be recomputed by each user. In term of communication costs, $\mathcal{K}$(II) $\geq$ $\mathcal{K}$(I) $\approx$ $\mathcal{K}$(IV) $\approx$ $\mathcal{K}$(III); in term of computational costs, $\mathcal{K}$(I) $\approx$ $\mathcal{K}$(IV) $\geq$ $\mathcal{K}$(II) $\approx$ $\mathcal{K}$(III). $\mathcal{T}$(I) means that all block tags need to be regenerated; $\mathcal{T}$(II) means that all block tags need to be re-signed with the original user or a delegated user as the proxy; $\mathcal{T}$(III) means that the block tags generated by the revoked user need to be re-signed with the CSP as the proxy; $\mathcal{T}$(IV) means that the data blocks signed by the revoked user should be found out first and their tags should be regenerated; $\mathcal{T}$(V) means that a post-revocation authenticator is introduced to ensure that the data blocks signed by the revoked user and their audit metadata cannot be tampered with. In term of computational costs, $\mathcal{T}$(I) $\geq$ $\mathcal{T}$(II) $\geq$ $\mathcal{T}$(IV) $\geq$ $\mathcal{T}$(III) $\geq$ $\mathcal{T}$(V).

are twofold. First, due to losing local control of data, cloud users (data owners) can no longer verify the integrity of their data via traditional techniques popularly employed in local storage. Second, although the infrastructures of cloud storage are much more powerful and reliable than personal computing devices, they are more vulnerable to both internal and external security threats due to the open and shared nature of the cloud. More severe still is that some dishonest CSPs, having suffered Byzantine failures occasionally that corrupt data, may try to conceal the fact of data corruptions for their own self-interest. To address this concern, the cloud storage auditing technique, also called the cloud data auditing, whose purpose is to verify the integrity of the outsourced data remotely, is popularly employed [5,14,26]. Generally, there are two models for cloud data auditing, i.e., private auditing [8,25] and public auditing [1,28,33]. In the former, the verification is carried out between cloud users and CSPs, while in the latter an authorized third-party auditor (TPA) is introduced to perform the verification. In comparison, the latter can provide more dependable auditing results and remarkably reduce the users' burden [27,29,32]. Thus, it is believed to be more reasonable and practical, and has been popularly employed for cloud data auditing [14,21,26]. In this work, therefore, we focus on public auditing.

So far, many novel solutions for different cloud auditing requirements, such as, privacy-preserving auditing [10,32,38], dynamic data auditing [33,37,42] and multi-replica auditing [3,7,20], have been proposed. More recently, with the increasing popularity of collaboration and teamwork in the cloud, shared data auditing has become a new hot topic in the cloud data auditing field [12,31,41]. Differing from the personal cloud data possessed by only a single user, shared cloud data can be accessed and modified by all authorized users in a group, e.g., staff of an organization and collaborators working in a team. The collaborative and sharing nature of shared cloud data places greater demands on the desirable integrity auditing scheme. Besides privacy-preserving, support for data dynamics and batching verification, the shared-data auditing should also achieve the following security and function requirements:

- **Identity-privacy preservation**: The TPA, who is usually considered to be trusted but curious, might collect users' identity information in the auditing process to obtain some significant privacy information, such as the users who play pivotal roles and the data blocks that contain high-value data. Thus, it is indispensable to protect the identity privacy of users in the shared-data auditing [12,30,36].
- **Support for group dynamics**: In a data-sharing group, it is common for either some new users to join at some point or some existing users to leave (or be removed) at any time. Thus, in the shared-data auditing, it is important to support group dynamics, including user enrollment (i.e., adding a new user) [30,36,39] and user revocation (i.e., removing an existing user) [30,39,41]. To be specific, the user enrollment should be easily achieved without regenerating any audit metadata (e.g. block tags); and the user revocation should ensure that the data blocks signed by the revoked user and their audit metadata cannot be tampered with.
- **Identity traceability**: In practical applications, a few users of granted group membership might maliciously modify certain shared data for their own self-interest, which would largely destroy the usability of the shared data and even result in social and economic losses. Thus, it is significant to enable identity traceability in the shared-data auditing [12,36]. That is, the data owner (or group manager) should be able to trace all the modification operations in the collaborative work, and reveal the identities of misbehaved users if necessary.

As shown in Table 1, there have been some fruitful auditing schemes for shared data. However, none of the existing schemes could fully achieve all the security and performance requirements mentioned above. Particularly, they have varying degrees of performance and security issues in achieving group dynamics. For example, in Oruta [30], as a result of adopting

Table 2
Notations.

Symbols	Descriptions
$\mathbb{G}_1$, $\mathbb{G}_T$	Two multiplicative cyclic groups of a large prime order p.
H	A secure hash function such that $H: \{0,1\}^* \rightarrow \mathbb{Z}_p^*$.
e	A bilinear map function such that $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$.
$U = \{u_1, u_2, \ldots, u_w\}$	w users in the data-sharing group, where u_1 is the group manager.
U_R	The set of revoked users.
U_L	The set of all the legal users in the group.
$\mathcal{F} = \{m_1, m_2, \ldots, m_n\}$	A shared data file with n blocks.
pk_i, sk_i	The public key and the secret key of the ith group user.
σ_i	The tag for the ith data block.
ϑ	The file tag of the challenged file.
ID	The identifier of the challenged file.
$SIG(sk_1, ID)$	The signature on ID under the private key sk_1 of the group manager.
RU-blocks	The data blocks signed by the revoked users.
IDX	The index set of all the challenged blocks.
S	The set of random numbers for the challenged blocks.
Φ	The tag proof of the challenged blocks.
Ω_1	The data proof of the challenged blocks.
T	The tag proof of RU-blocks.
Ω_2	The data proof of RU-blocks.
Θ	The auxiliary auditing factor.
λ_i	The revocation parameter of the ith data block.
Λ	The post-revocation authenticator.
Ψ	The revocation message from the group manager to revoke a user.

the ring signature technique to generate block tags, when a user is added into or revoked from a group, all users' keys and all block tags have to be regenerated; in SDIVIP2 [39], each user adopts a key combined with the personal keys of all the users to generate block tags, so during both the user enrollment and revocation, the shared key should be recomputed by each user, and all block tags need to be re-signed by himself or with a delegated user as the proxy; in the schemes reported in Refs. [11,31,36,40,41] where each user adopts his/her own key for generating block tags, during the user revocation the block tags related to the revoked user must be re-signed with the CSP as the proxy [31,40,41]. In addition, some existing schemes (such as, [11,31,36,41]) can potentially suffer from collusion attacks, since the revoked user might share his/her private key with the CSP. In this case, the CSP could tamper with the data blocks previously signed by the revoked user. Things can get even worse if the CSP-based proxy re-signature technique is used to re-computed block tags during the user revocation [31,41]. The CSP would even obtain the private key of the delegated user, and can tamper with the data blocks related to both the revoked user and delegated user (Table 2).

In view of the above shortcomings of the existing shared-data auditing schemes, in this paper, we seek to present an effective public auditing scheme to meet all the indispensable functional and security requirements mentioned above. Particularly, we propose an efficient and secure group management mechanism to support group dynamics. Specifically, in a data-sharing group, each valid user adopts his/her personal key to generate block tags, which enables new users to join the group expediently without affecting any other users or any existing shared data; to achieve efficient user revocation, we design a lazy-revocation strategy, which involves two operations. First, when a user is revoked, to prevent data blocks related to the revoked user (RU-blocks for short) from unauthorized modifications, a signature of the group manager on the aggregate value of all the RU-blocks' tags (called post-revocation authenticator) is introduced. Note that, thanks to the post-revocation authenticator, even if the revoked user shares the private key with the CSP, the CSP will not be able to tamper with any data blocks, suggesting that our strategy can resist the collusion attack effectively. Second, when the RU-blocks are modified by the other valid users, their tags would be regenerated accordingly. In other words, our lazy-revocation strategy does not need to do anything other than generating a post-revocation authenticator during the user revocation to protect RU-blocks, and postpones the tag regeneration of RU-blocks until they are modified by other users. Thus, compared with the existing schemes [12,30,31,39–41], our strategy can effectively reduce the computational overhead during the user revocation.

In addition, to preserve the identity privacy in the auditing process, we exploit the properties of bilinear maps to convert the signatures of different users on the challenged blocks to the ones signed by the group manager prior to generating the tag proof. To support identity traceability, we introduce a modification record table to record the operator, operation content, and operating time for each modified data block. Moreover, we further extend the dynamic hash table [27] to support shared-data dynamics, employ a random masking to protect data privacy in the auditing process, and achieve batch auditing by taking advantage of the aggregate signature technique.

In general, our contributions in this work can be summarized as follows:

1. We present a novel public auditing scheme for shared data, which can fully achieve all the necessary functional and security requirements, including data and identity privacy protection, data dynamics, group dynamics, identity traceability and batch verification.

2. We propose an efficient and secure group management mechanism to support group dynamics. Particularly, we present a lazy-revocation strategy to achieve user revocation in a secure and cost-effective manner.
3. We formally prove the security of the proposed scheme, and evaluate its performance by theoretical analysis and experimental comparisons with the state-of-the-art schemes. The experimental results demonstrate that the proposed scheme can efficiently achieve secure auditing for shared cloud data, and outperforms the previous ones in computational overhead while maintaining relatively low communication costs.

The rest of the paper is organized as follows: In Section 2, we review the related work concerning cloud data auditing, especially regarding the shared-data auditing. We introduce background and necessary preliminaries for our work in Section 3, and present our scheme in detail in Section 4. We formally prove and analyze the security of our scheme in Section 5, and conduct comprehensive performance evaluations by theoretical analysis and experimental comparisons with the state-of-the-art schemes in Section 6. Finally, Section 7 concludes the outcome of this work.

2. Related work

Nowadays, the cloud data auditing has been witnessed as the indispensable technique for remote integrity verification of cloud data. One of the earliest work is proof of retrievability (PoR) proposed by Juels et al. [13], where the data owner can not only verify the integrity of outsourced data but also ensure the retrievability using error-correcting codes. However, PoR is a typical private auditing scheme, and cannot support the verification conducted by a third party. At the same time, Ateniese et al. [1] first presented a model for provable data possession (PDP) and gave two provably-secure PDP schemes using RSA-based homomorphic authenticators, which support public verification for users' data at untrusted servers with sampling inspection. As mentioned above, in comparison with the private auditing, the public auditing can offer credible auditing results as well as greatly reduce users' burden for verification. Therefore, it is believed to be more reasonable and promising. In recent years, many public auditing schemes have been successively presented to address various concerns, such as data-privacy preservation and support for data dynamics.

Privacy protection, aiming at protecting the content of users' data from being obtained by unauthorized entities [17,18,22], is widely considered as an indispensable precondition of the public auditing [14,27,32]. To address this concern, a popular approach is to employ a random masking to blind the data proof [27,32,37], which has two implementation strategies. In the first strategy [27,37], the TPA first generates a mask number R with a random number r and a global parameter y as $R = y^r$, and sends R to the CSP together with the challenge; while responding to the challenge, the CSP computes the masked data proof of M as $M' = e(u, R)^M$, where e is a bilinear map and u is a global parameter. In the other strategy [32], the CSP calculates a mask number $R = y^r$, and blinds the data proof of M by computing $M' = M + rH(R)$, where r is a randomly chosen number, y is a global parameter and H is a hash function. In this paper, we also adopt the approach of random marking to protect the data privacy, but design a new implementation way, which will be described in detail in Section 4. Recently, Yu et al. [38] introduced zero-knowledge-proof technique to achieve privacy-preserving auditing, by which the CSP can convince the TPA to check the integrity of cloud data with neither the knowledge of data blocks nor their tags.

A practical auditing scheme should not disregard data dynamics, in that cloud data may be not only accessed but also updated frequently due to various application purposes [5,6,35]. To address this concern, authenticated data structures are widely introduced into auditing schemes. For example, Erway et al. [8] first incorporated a rank-based authenticated skip list into the PDP model to support data dynamics; Wang et al. [33] employed Merkle Hash Tree (MHT) to achieve public auditing for dynamic data; Zhu et al. [42] designed another authenticated data structure, called index hash table (IHT), to support data dynamics; Tian et al. [27] further presented a two-dimensional data structure, dynamic hash table (DHT), to achieve both the dynamic data updating and public verification effectively; It is worth mentioning that, in Oruta [30] and Panda [31], the authors leveraged IHT to achieve public verification for shared data. Given that DHT is proved to be more effective than IHT on data updating [27], we prefer to employ DHT to support dynamic updating of shared data in this work.

Besides the traditional requirements for cloud data auditing, the shared-data auditing should further fulfill the other three demands, namely, identity-privacy preservation, support for group dynamics and identity traceability. As shown in Table 1, the existing shared data auditing schemes [11,12,30,36,39], except for Panda [31], YY14 [40] and YY15 [41], can preserve the identity privacy effectively. In general, in accordance with the adopted signature techniques, they can be divided into four categories, i.e., the scheme utilizing the ring signature technique [30], the scheme employing the group signature technique [12], the scheme adopting shared signature keys [39], and the schemes using individual signature keys [11,31,36,40,41]. In the first three types of schemes, since it is impossible for the TPA to know the signers from the block tags, the identity privacy is naturally protected well in the auditing process. However, they would induce heavy communication and computational overheads in the user enrollment or/and revocation phases.

Specifically, in Oruta [30] that adopts the ring signature technique, in both the user enrollment and revocation phases, all users' keys and block tags must be regenerated. In JCM16 [12], the authors introduced an improved group signature technique, named as verifier-local-revocation group signature to achieve secure group management. However, the computational costs for user revocation are relatively high, because in order to revoke a user, the data blocks signed by the revoked user need to be found out first, and their tags must be regenerated. In SDIVIP2 [39], because the key combined with the personal keys of all users is employed to generate block tags, in the both user enrollment and revocation phases, the shared

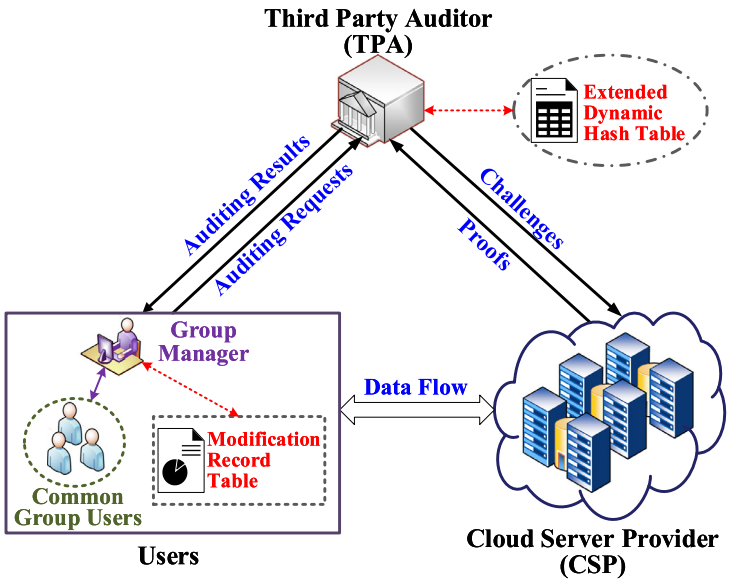

Fig. 1. System model for shared-data auditing.

key should be recomputed by each user, and all block tags need to be re-signed by himself or with a delegated user as the proxy.

In comparison with the former three types, the fourth type of schemes can achieve user revocation by regenerating the block tags related to the revoked user with the CSP as the proxy. Moreover, by virtue of the proxy re-signature technique, the tags of challenged blocks can be converted to the signatures of the group manager [40,41] or the user launching the challenge [11] to protect the identity privacy. However, some schemes adopting individual keys [11,31,36,41] can potentially suffer from the collusion attack, because the revoked user might share his/her private key with the CSP. Once the revoked user colludes with the CSP, the latter could tamper with the data blocks previously signed by the former. Furthermore, if the CSP-based proxy re-signature technique is used to re-computed block tags during the user revocation [31,41], the CSP could even obtain the private key of the delegated user, and thereby tamper with the data blocks related to both the revoked user and delegated user.

To avoid the malicious modification for shared data, it is crucial to enable identity traceability. So far, however, only a few schemes noticed the problem. In the schemes employing the group signature technique [12], the group manager could be authorized to open the group signatures to find out the operators. Moreover, to achieve the identity traceability, Yang et al. [36] introduced an identity-block list and an identity-key list to record the users and their modification operations. Inspired by Yang's work, we design a modification record table to track the users' operations on data blocks, which will be described in Section 3.

Although some fruitful public auditing schemes for shared data have been presented, none of them could fully achieve all the security and performance requirements mentioned above. Thus, in this paper, we seek to present a more comprehensive scheme for shared-data auditing. Particularly, we incorporate the homomorphic authentication technique based on individual keys and a lazy-revocation strategy to realize a novel membership management mechanism, which can achieve the enrollment and revocation of users in a more secure and cost-effective way.

3. Background and preliminaries

In this section, we describe the necessary background and preliminaries for our work. Specifically, we first present the public auditing model for shared cloud data and its design goals in Section 3.1, and introduce some cryptographic preliminaries in Section 3.2. Furthermore, we propose an extended dynamic hash table in Section 3.3 to support shared-data dynamics, and a modification record table in Section 3.4 to support identity traceability. Finally, the security assumptions are given in Section 3.5.

3.1. Problem statement

As illustrated in Fig. 1, we focus on the design of an effective public auditing scheme for shared data in this work, which involves the following three entities:

- **Users**, who are members of a sharing-data group, involve data owners, a group manager and a certain number of common users. The data owners are the group users who originally create cloud data and share it with all the users. The

group manager is a group user delegated by the data owners to manage the group, including adding a new user, revoking a user, and tracing users' operations. Each legal group user can access and update all shared data, and for each updated data block, he/she should create a digital signature, called block tag, to ensure its correctness.
- **Cloud Service Provider (CSP)**, manages and coordinates a lot of cloud servers to provide scalable and on-demand data storage services to group users.
- **Third Party Auditor (TPA)**, also called public auditor, is authorized to verify the correctness and integrity of shared cloud data on behalf of group users, and dedicated to provide reliable and credible auditing results.

The users can enjoy high-performance services of data outsourcing and maintenance provided by the CSP. However, due to losing local possession of data, they have a strong desire to check the correctness and integrity of their cloud data regularly. To obtain a believable result as well as reduce the users' burden for verification, the TPA is often introduced to verify the integrity of cloud data. However, the TPA is not expected to be able to learn privacy information regarding the users and their data.

As in the existing public auditing schemes, the TPA is considered to be credible but curious [26]. That is to say, the TPA can provide a credible result by independent auditing, but may be curious about the cloud data and identity information of users. The CSP is considered to be semi-honest. In other words, the CSP can provide scalable and on-demand data outsourcing services for users in the normal cases, but may conceal data corruption incidents intentionally for his/her own interests. To hide the fact of data corruption, the CSP may further launch the following attacks to the TPA:

- *Forge attack*. The CSP attempts to forge the data blocks and corresponding tags to pass the verification.
- *Replacing attack*. The CSP attempts to replace a corrupted block and its tag with another block and its corresponding tag to pass the verification.
- *Replay attack*. The CSP attempts to deceive the TPA by using the proofs generated previously.
- *Collusion attack*. The CSP may collude with the revoked users to obtain their secret key, and accordingly be able to tamper with the related data blocks and their tags.

To enable secure and efficient public auditing for shared cloud data, our scheme aims to fulfill the following security and functional requirements:

1. *Public auditing*: any authorized TPA is able to verify the correctness and integrity of shared cloud data.
2. *Blockless verification*: the TPA can verify the integrity of shared cloud data without retrieving any data blocks.
3. *Auditing correctness*: the TPA should be able to verify the integrity of shared cloud data correctly.
4. *Privacy preserving*: from the auditing proofs, the TPA can obtain neither any actual content of shared data nor any identity information of group users.
5. *Support for group dynamics*: both the user enrollment and revocation should be achieved in a secure and efficient manner.
6. *Support for identity traceability*: the group manager should be able to trace the information about data updating.
7. *Batch auditing*: the TPA should be able to handle multiple auditing tasks from various groups in a fast and cost-efficient manner.
8. *Lightweight*: the public verification should be performed with the minimum communication and computational costs.

3.2. Preliminaries

To make our paper self-contained, we would like to start with a quick review of the involved cryptographic background in our proposed scheme.

3.2.1. Bilinear map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of a large prime order p. A bilinear map is a map function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties: A) **Computability**: there exists an efficient computable algorithm for solving the function e; B) **Bilinearity**: $\forall h_1, h_2, h_3 \in \mathbb{G}$, and $\forall u, v \in \mathbb{Z}_p^*$, $e(h_1^u, h_2^v) = e(h_1, h_2)^{u \cdot v}$, and $e(h_1, h_2 \cdot h_3) = e(h_2 \cdot h_3, h_1) = e(h_1, h_2) \cdot e(h_1, h_3)$; C) **Nondegeneracy**: let g be a generator of $\mathbb{G}$, $e(g, g) \neq 1$.

3.2.2. Homomorphic verifiable authenticator

Homomorphic verifiable authenticator (HVA), is popularly adopted as a building block for public auditing [27,32,36], which enables a public auditor to check the integrity of cloud data without downloading any original data. In general, digital signatures (e.g. RSA-based signature and BLS signature) can be used to generate HVAs for public verification. Therefore, HVAs for public auditing are also considered as homomorphic verifiable signatures, also called homomorphic verifiable tags (HVTs). Besides unforgeability, HVTs satisfy the following properties [19,30,31]:

- **Blockless verifiability**. Using the proof constructed by HVTs, the TPA can verify the integrity of the data blocks without knowing their actual data content.
- **Homomorphism**. Let $\mathbb{G}$ and $\mathbb{H}$ be multiplicative groups of a large prime order p, "$\oplus$" and "$\otimes$" be operations in $\mathbb{G}$ and $\mathbb{H}$. If a map function $f : \mathbb{G} \to \mathbb{H}$ satisfies homomorphism, then $\forall g_1, g_2 \in \mathbb{G}$, $f(g_1 \oplus g_2) = f(g_1) \otimes f(g_2)$.
- **Non-malleability**. Let σ_1 and σ_2 be signatures on blocks m_1 and m_2 respectively, α_1 and α_2 be two random numbers in $\mathbb{Z}_p^*$. For the given block, $m' = \alpha_1 m_1 + \alpha_2 m_2$, a user, who does not know the private key sk, cannot generate a legitimate signature σ' on m' by combining σ_1 and σ_2.

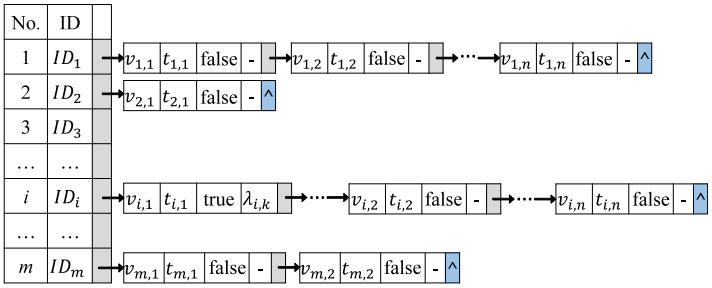

Fig. 2. Extended dynamic hash table.

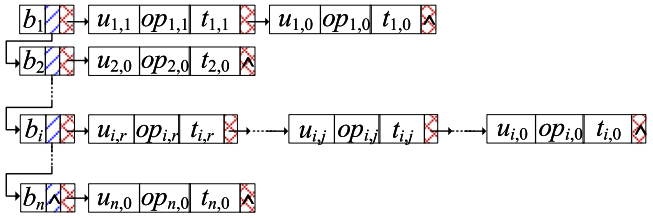

Fig. 3. Modification record table.

3.2.3. Lazy revocation

There are two user revocation models in multi-user systems, i.e., active revocation model and lazy revocation model [2]. In the former, all cryptographic information should be updated immediately once a user is revoked. This is expensive and might even cause disruptions of the system's normal operations. In the latter, the cryptographic information stays unchanged when a user is revoked, and is re-computed only when the corresponding data is updated once again [24]. Inspired by the notion, in this paper we design a lazy revocation strategy to achieve efficient revocation in a data-sharing group while ensuring the security of integrity auditing. The key idea is that, during the user revocation, nothing other than generating a post-revocation authenticator to protect the blocks related to the revoked user, needs to be done, and the tag regenerations of the related blocks are postponed until they are modified by the other users.

3.3. Extended dynamic hash table

To support data dynamics, authenticated data structures are popularly introduced into auditing schemes [27,36,42]. In our previous work [27], we presented a two-dimensional data structure, dynamic hash table (DHT), to achieve both the dynamic data updating and public verification effectively. Particularly, it is proved to be more effective than index hash table (IHT) that is adopted in the typical shared-data auditing schemes like Oruta [30] and Panda [31]. Therefore, we prefer to employ DHT to support dynamic updating of shared data in this work.

However, to accommodate the lazy revocation strategy, we improve this data structure, and give an extended dynamic hash table (EDHT), as illustrated in Fig. 2. Like in the original DHT, in an EDHT, all file elements are constructed as an array, and each file element consists of the index number (No_i) of the file (e.g. F_i), the File identifier (ID_i) and a pointer indicating its first block element; all the block elements of each file are organized using a linked list with the corresponding file element as the header node. The main difference is that, besides the current version of the block $v_{i,j}$, its time stamp $t_{i,j}$ and a pointer indicating the next node, each block element (e.g. the jth block of the ith file $m_{i,j}$) still contains two new fields, namely, revocation flag and revocation parameter. The revocation flag of each block element indicates whether the corresponding data block is last modified by a revoked user. That is, if the flag is true, it means that the current block was last modified by a revoked user; otherwise, the current block was handled by a valid user. The revocation parameter is valid, if and only if the revocation flag is set as true. The revocation parameter is produced by the group manager during the generation of post-revocation authenticator, and would be employed in the tag regeneration phase and possibly in the auditing process, which will be described in detail in the next section.

3.4. Modification record table

To achieve the identity traceability, we design a modification record table (MRT) to track users' operations on each data block. As illustrated in Fig 3, the MRT is a two-dimensional data structure maintained by the group manager, which records all the modified blocks and related operation information since the latest successful verification of data usability. It involves two types of metadata items, i.e., block items and operation items. All the block items are constructed as a linked list, which makes it easy to add new elements or delete existing ones. Each block item contains its block identifier (b_i), a pointer

indicating the item of the next modified block, and a pointer indicating its first operation item. All the operation items for each modified block are organized as a linked stack, of which the first element is the latest operation record. For the ith block, each operation item O_j consists of the identifier of the user who modified the data block (denoted by $u_{i,j}$), the specific operation $op_{i,j}$ (such as insertion, update and deletion), and the timestamp $t_{i,j}$.

In practice, the group manager could maintain a MRT for each data file. When a user modified one or some blocks, the group manager should update the corresponding MRT to record all the operations. For a data file, if the usability of its current version has been verified successfully, its MRT can be emptied. However, if some data are maliciously modified, the group manager could find out the alleged users by virtue of the MRT.

3.5. Security assumptions

The security of the proposed scheme is based on the following assumptions.

Definition 1 (Computational Diffe–Hellman (CDH) Assumption)**.** Let $\mathbb{G}$ be a cyclic group of prime order p, for a randomly chosen generator g and random numbers $a, b \in \mathbb{Z}_p^*$, given $(g, g^a, g^b) \in \mathbb{G}$, it is computationally intractable to compute the value g^{ab}. That is, for any probabilistic polynomial-time adversary $\mathcal{A}$, the probability of solving the CDH problem is negligible, namely,

$$Pr\Big(\mathcal{A}_{CDH}\big(g, g^a, g^b \in \mathbb{G}\big) \to g^{ab} \in \mathbb{G} : \forall a, b \underset{R}{\in} \mathbb{Z}_p^*\Big) \le \varepsilon.$$

Definition 2 (Discrete Logarithm (DL) Assumption)**.** Let $\mathbb{G}$ be a cyclic group of prime order p with a generator g, for a given $h \in \mathbb{G}$, it is computationally intractable to compute the value $a \in \mathbb{Z}_p^*$, such that $h = g^a$. In other words, for any probabilistic polynomial-time adversary $\mathcal{A}$, the probability of solving the DL problem is negligible, namely,

$$Pr\big(\mathcal{A}_{DL}(g, h \in \mathbb{G}) \to a \in \mathbb{Z}_p^*, \text{s.t. } h = g^a\big) \le \varepsilon.$$

4. The proposed scheme

In this section, we will present our public auditing scheme for shared data, which consists of the dynamic verification protocol with privacy protection described in Section 4.1, the group management mechanism detailed in Section 4.2, and the batch verification protocol introduced in Section 4.3.

4.1. Public auditing for shared data

Let $\mathbb{G}_1, \mathbb{G}_T$ be two multiplicative cyclic groups of a large prime order p, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a bilinear map. g, g_1 and μ are the generators of $\mathbb{G}_1$. H is a secure hash function such that $H : \{0,1\}^* \to \mathbb{Z}_p^*$. Assume that the shared data file $\mathcal{F}$ is divided into n blocks, namely, $\mathcal{F} = \{m_1, m_2, \ldots, m_n\}$; there are w users (denoted by $U = \{u_1, u_2, \ldots, u_w\}$) in the data-sharing group, where u_1 is the data owner and also serves as the group manager, and the others are the common users. Our public auditing protocol, as illustrated in Fig. 4. involves five algorithms, namely, ***KeyGen, TagGen, Challenge, ProofGen***, and ***Verify***.

***KeyGen* (Key Generation)**. Each user (e.g. $u_i, i = 1, 2, \ldots, w$) first selects a random number $a_i \in \mathbb{Z}_p^*$; for the group manager u_1, his/her secret key is $sk_1 = a_1$ and public key is $pk_1 = y_1$, where $y_1 = g^{a_1}$; for each common user u_i $(i = 2, 3, \ldots, w)$, his/her secret key is $sk_i = a_i$ and public key is $pk_i = y_i$, where $y_i = y_1^{1/a_i}$.

***TagGen* (Tag Generation)**. For each block m_i $(i = 1, 2, \ldots, n)$, the user, who modified the block, generates a signature σ_i with his/her secret key $sk_j = a_j$ $(j = 1, 2, \ldots, w)$, which can be described as follows:

$$\sigma_i = \left(\mu^{H(v_i \| t_i)} g_1^{m_i}\right)^{a_j}, \tag{1}$$

where v_i and t_i are the version number and timestamp of the given block, respectively. This signature, often called block tag, serves as a metadata object, and should be uploaded to the CSP along with the block. Note that, although all HVAs based on public-key cryptography (such as RSA-HVA and BLS-HVA) can be employed in our scheme, we prefer to choose BLS-HVA for a shorter length of each block tag [27,31,33]. Moreover, for each modification on a data block, the group manager would add the operation information into the MRT, and ask the TPA to update the version information (i.e., v_i and t_i) of the block in the EDHT. It is worth mentioning that, to ensure the integrity of the unique file identifier ID, the group manager would also compute a file tag $\vartheta = ID \| SIG(sk_1, ID)$, where $SIG(sk_1, ID)$ is the signature on ID under the private key sk_1, and send it to the CSP.

Challenge. The TPA first retrieves the file tag ϑ, and verifies the signature $SIG(sk, ID)$ using the public key pk_1 of the group manager. If the verification fails, the TPA directly quits the verification by emitting FALSE; otherwise, it generates the challenge information as follows: (1) generate a challenge block set $IDX = \{idx_i | 1 \le i \le c, c \le n\}$ by randomly choosing c different data blocks from the ones that were not modified by the revoked users, and a set of random numbers $S = \{s_i | i \in IDX\}$, where $s_i \in \mathbb{Z}_p^*$; (2) Assume that D is the index set of all the RU-blocks (i.e., data blocks modified by the revoked users), for each $i \in D$, calculate $\beta_i = \lambda_i / \eta$, where $\eta \in \mathbb{Z}_p^*$ is a random number, and λ_i is the revocation parameter of the ith block; (3) Let the set of all the legal users in the group be U_L and the set of all the revoked users be U_R, for each $i \in U_L$, calculate $\gamma_i = y_i^{\omega} = g^{\omega \cdot a_1 / a_i}$, and for each $i \in U_R$, calculate $\tau_i = y_i^{\omega/\varepsilon} = g^{\omega/\varepsilon \cdot a_1 / a_i}$, where $\varepsilon \in \mathbb{Z}_p^*$ and $\omega \in \mathbb{Z}_p^*$ are two random numbers.

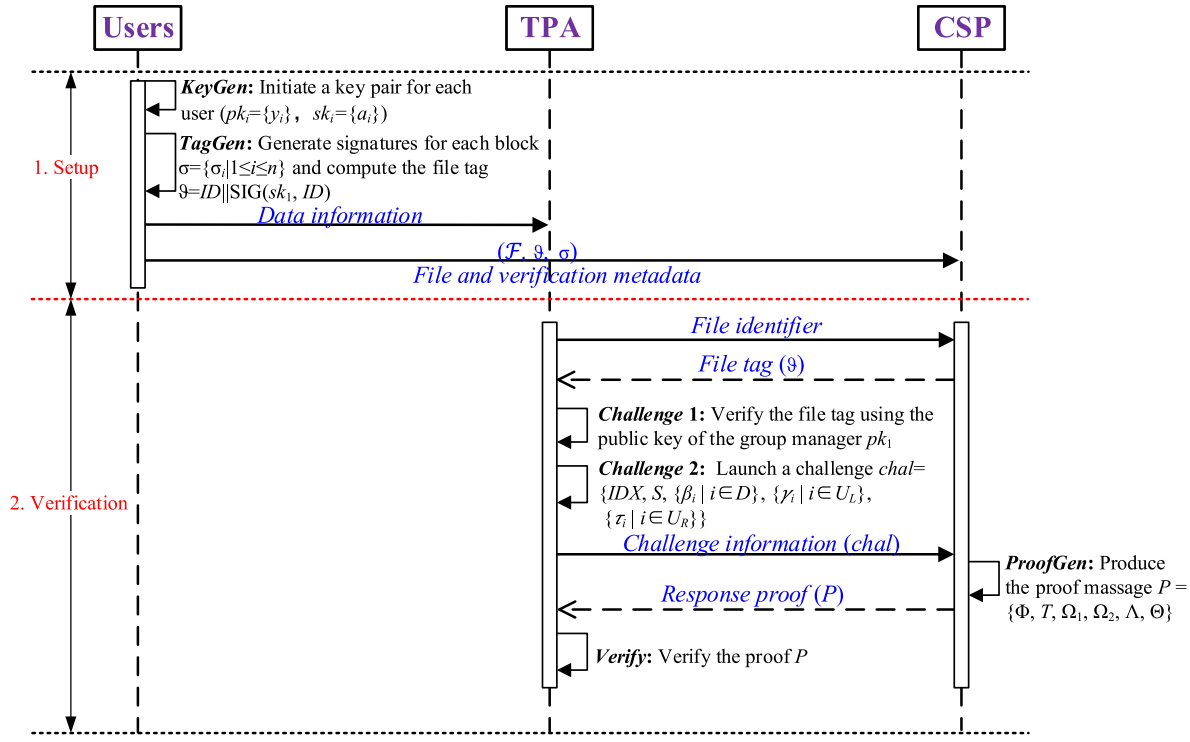

Fig. 4. Workflow of the auditing process of the proposed scheme.

Finally, the TPA launches a challenge by sending the challenge information $chal = \{IDX, S, \{\beta_i | i \in D\}, \{\gamma_i | i \in U_L\}, \{\tau_i | i \in U_R\}\}$ to the CSP.

***ProofGen* (Proof Generation)**. Once receiving the challenge from the TPA, the CSP would produce a set of response proofs for data storage correctness, which includes the proofs of challenged blocks, the proofs of all the RU-blocks, the post-revocation authenticator, and an auxiliary auditing factor.

For the challenged blocks, the CSP calculates the tag proof Φ as

$$\Phi = \prod_{i \in IDX} \left(e\left(\sigma_i, \gamma_j\right)\right)^{s_i}, \tag{2}$$

where $j \in U_L$ and the ith block was last handled by the jth user. Further, the CSP calculates the data proof Ω_1 as

$$\Omega_1 = \sum_{i \in IDX} m_i \cdot s_i + r, \tag{3}$$

where $r \in \mathbb{Z}_p^*$ is a random mask, and used to blind the data proof to protect the data privacy.

For all the RU-blocks (if existing), the CSP calculates the tag proof T as

$$T = \prod_{i \in D} \left(e\left(\sigma_i, \tau_j\right)\right)^{\beta_i}, \tag{4}$$

where $j \in U_R$ and the ith block was last handled by the jth revoked user. Further, the CSP calculates the data proof Ω_2 as

$$\Omega_2 = \sum_{i \in IDX} m_i \cdot \beta_i + r. \tag{5}$$

The post-revocation authenticator Λ is produced by the group manager during the user revocation, and maintained by the CSP. We will describe its detail in the next subsection.

Moreover, the CSP calculates the auxiliary auditing factor as

$$\Theta = e(g_1, y_1)^{-r}. \tag{6}$$

Finally, the CSP sends $P = \{\Phi, T, \Omega_1, \Omega_2, \Lambda, \Theta\}$ back to the TPA as the response.

Verify. The verification may involve two sub-steps. First, the TPA checks whether U_R is empty. If $U_R = \emptyset$, meaning that there is no revoked user or all the RU-blocks have been modified again by the other valid users, then the TPA can directly

verify the correctness of the data file; otherwise, he/she verifies the post-revocation authenticator by checking the following equation:

$$\Lambda \stackrel{?}{=} T^{\eta\cdot\varepsilon/\omega}. \tag{7}$$

If Eq. (7) does not hold, the TPA terminates the verification and outputs FALSE, meaning that some tags of the RU-blocks have been tampered with; otherwise, he/she can go on with the verification. The correctness of Eq. (7) will be demonstrated in the next subsection. To verify the correctness of the data file, the TPA checks the following equation:

$$\Phi \cdot T^{\eta\cdot\varepsilon} \stackrel{?}{=} e\left(\mu^{\sum_{i\in IDX} B_i s_i + \sum_{k\in D} B_k\lambda_k}\cdot g_1^{\Omega}, y_1\right)^{\omega}\cdot \Theta^{\omega\cdot(\eta+1)}, \tag{8}$$

where $B_i = H(v_i\|t_i)$, and $\Omega = \Omega_1 + \Omega_2\cdot\eta$. If Eq. (8) holds, it outputs TRUE; otherwise, FALSE.

The correctness of the above verification equation can be demonstrated as follows.

$$\begin{aligned}
\Phi &= \prod_{i\in IDX} e\left(\sigma_i, \gamma_j\right)^{s_i} = \prod_{i\in IDX} e\left(\left(\mu^{B_i}\cdot g_1^{m_i}\right)^{a_j}, g^{\omega\cdot a_1/a_j}\right)^{s_i}\\
&= \prod_{i\in IDX} e\left(\mu^{B_i}\cdot g_1^{m_i}, y_1\right)^{\omega s_i} = e\left(\prod_{i\in IDX}\mu^{B_i s_i}\cdot g_1^{\sum_{i\in IDX} m_i s_i}, y_1\right)^{\omega}\\
&= e\left(\mu^{\sum_{i\in IDX} B_i s_i}\cdot g_1^{\Omega_1 - r}\right)^{\omega},\\
T^{\eta\cdot\varepsilon} &= \prod_{k\in D} e\left(\sigma_k, \tau_j\right)^{\beta_k\cdot\eta\cdot\varepsilon} = \prod_{k\in D} e\left(\left(\mu^{B_k}\cdot g_1^{m_k}\right)^{a_j}, g^{\frac{\omega}{\varepsilon}\cdot\frac{a_1}{a_j}}\right)^{\lambda_k\varepsilon}\\
&= \prod_{k\in D} e\left(\mu^{B_k}\cdot g_1^{m_k}, y_1\right)^{\lambda_k\omega} = e\left(\prod_{k\in D}\mu^{B_k\lambda_k}\cdot g_1^{\sum_{k\in D} m_k\lambda_k}, y_1\right)^{\omega}\\
&= e\left(\mu^{\sum_{k\in D} B_k\lambda_k}\cdot g_1^{(\Omega_2 - r)\cdot\eta}, y_1\right)^{\omega}.
\end{aligned}$$

Further, we have

$$\begin{aligned}
&\Phi\cdot T^{\eta\cdot\varepsilon}\\
&= e\left(\mu^{\sum_{i\in IDX} B_i s_i}\cdot g_1^{\Omega_1 - r}\right)^{\omega}\cdot e\left(\mu^{\sum_{k\in D} B_k\lambda_k}\cdot g_1^{(\Omega_2 - r)\cdot\eta}, y_1\right)^{\omega}\\
&= e\left(\mu^{\sum_{i\in IDX} B_i s_i}\cdot\mu^{\sum_{k\in D} B_k\lambda_k}\cdot g_1^{\Omega_1 + \Omega_2\cdot\eta - (1+\eta)\cdot r}, y_1\right)^{\omega}\\
&= e\left(\mu^{\sum_{i\in IDX} B_i s_i}\cdot\mu^{\sum_{k\in D} B_k\lambda_k}\cdot g_1^{\Omega}, y_1\right)\cdot e(g_1, y_1)^{-\omega\cdot(1+\eta)\cdot r}\\
&= e\left(\mu^{\sum_{i\in IDX} B_i s_i + \sum_{k\in D} B_k\lambda_k}\cdot g_1^{\Omega}, y_1\right)^{\omega}\cdot\Theta^{\omega\cdot(\eta+1)}.
\end{aligned}$$

Moreover, from the above derivation, we can get that,

$$\Phi = \prod_{i\in IDX} e\left(\mu^{B_i}\cdot g_1^{m_i}, y_1\right)^{\omega\cdot s_i}. \tag{9}$$

$$T = \prod_{k\in D} e\left(\mu^{B_k}\cdot g_1^{m_k}, y_1\right)^{\frac{\omega}{\varepsilon}\cdot\beta_k}. \tag{10}$$

That is, from Φ and T, the TPA cannot learn any information about who handled which blocks, since the verification only involves the public key of the group manager. Thus, our scheme can effectively protect the identity privacy.

4.2. Group management

The group management involves two aspects, i.e., the user enrollment (i.e., adding a new user) and user revocation (i.e., removing an existing user).

In the proposed scheme, the user enrollment is relatively simple, because each legal user has his/her own key pair. Specifically, if a new user needs to be added, the group manager first adds him/her into the user list and notifies the CSP

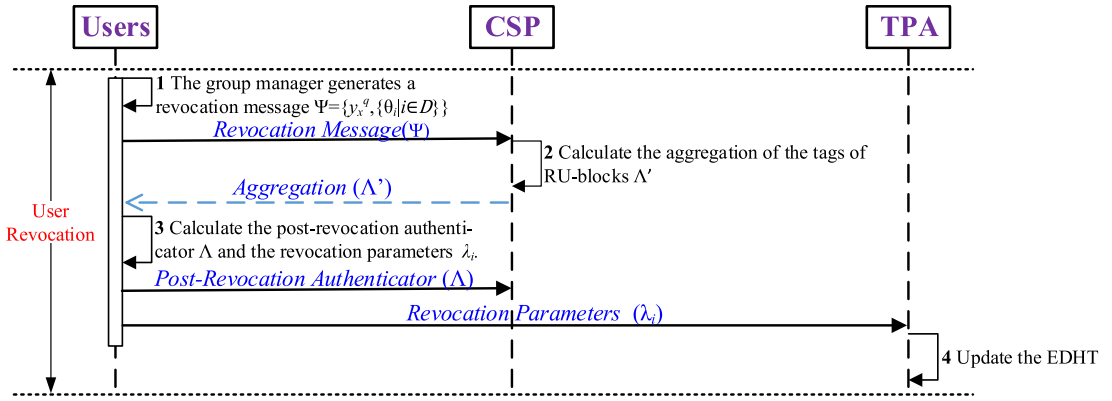

Fig. 5. Workflow of the user revocation.

to add him/her into the access control list. Moreover, the new user can perform the ***KeyGen*** algorithm to obtain his/her key pair.

To achieve efficient user revocation, we present a novel lazy revocation strategy, as illustrated in Fig. 5. If one or some users need to be revoked, the group manager generates a post-revocation authenticator to protect the RU-blocks, and the regenerations of their tags are postponed until these blocks are modified by the other users. Of course, the group manager should also add the revoked users into the set U_R, and notify the CSP to remove them from the access control list. Specifically, assume that the user u_x needs to be revoked, the group manager informs the CSP of the user revocation by sending a message Ψ as follows:

$$\Psi = \{y_x^q, \{\theta | i \in D\}\}, \tag{11}$$

where D is the index set of the blocks related to u_x, $q \in \mathbb{Z}_p^*$ and $\theta_i \in \mathbb{Z}_p^* (i \in D)$ are random numbers.

Upon receiving the revocation message Ψ, the CSP first calculates the aggregation of the tags related to the revoked user u_x as

$$\Lambda' = \prod_{i \in D} e\left(\sigma_i, y_x^q\right)^{\theta_i}. \tag{12}$$

Further, the CSP sends Λ' back to the group manager. After getting Λ', the group manager can generate the post-revocation authenticator as

$$\Lambda = \Lambda'^{\rho}, \tag{13}$$

where $\rho \in \mathbb{Z}_p^*$ is a random number. Next, for each RU-block, the group manager calculates a revocation parameter as

$$\lambda_i = q \cdot \rho \cdot \theta_i, \ i \in D. \tag{14}$$

Moreover, the group manager should send $(R, \{\lambda_i | i \in D\})$, where R is the index set of the RU-blocks, to the TPA to update the revocation flags and parameters of the RU-blocks in the EDHT.

Note that if there are multiple users to be revoked, namely, $|U_R| \geq 2$, the post-revocation authenticator is the product of the authenticators of all the revoked users, i.e.,

$$\Lambda = \prod_{i \in U_R} \Lambda_i, \tag{15}$$

where Λ_i is the post-revocation authenticator for the ith revoked user, and calculated according to Eq. (13). In addition, if $U_R = \emptyset$, then $\Lambda = 1$.

When a data block m_i handled by a revoked user $u_x (x \in U_R)$ is modified again by another legal user, the CSP would update the post-revocation authenticator as

$$\Lambda = \Lambda / e(\sigma_i, y_x)^{\lambda_i}. \tag{16}$$

Apparently, prior to the calculation, the CSP should first retrieve the revocation parameter λ_i from the TPA. Moreover, the tag of m_i would be regenerated by the newly handling user. Accordingly, the TPA should set the revocation flag of m_i as FALSE, and clear the revocation parameter. If all the blocks related to the user u_x have been modified again by the other valid users, then u_x should be removed from U_R.

Now, we can formally prove the correctness of Eq. (7) as follows.

$$\begin{aligned} T^{\eta\cdot\frac{\varepsilon}{\omega}} &= \prod_{i\in D} e\left(\sigma_i, \tau_j\right)^{\beta_i\cdot\eta\cdot\frac{\varepsilon}{\omega}} = \prod_{i\in D} e\left(\sigma_i, g^{\frac{\omega}{\varepsilon}\cdot\frac{a_1}{a_j}}\right)^{\lambda_i\cdot\frac{\varepsilon}{\omega}} \\ &= \prod_{i\in D} e\left(\sigma_i, y_j^{\frac{\omega}{\varepsilon}}\right)^{q\cdot\rho\cdot\theta_i\cdot\frac{\varepsilon}{\omega}} = \prod_{i\in D} e\left(\sigma_i, y_j\right)^{q\cdot\rho\cdot\theta_i} \\ &= \prod_{i\in D} e\left(\sigma_i, y_j^{q}\right)^{\rho\cdot\theta_i} = \Lambda'^{\rho} = \Lambda \quad (j\in U_R). \end{aligned}$$

4.3. Batch verification

In the shared-data auditing scenario, the core of the batch auditing is how to concurrently handle multiple verification tasks from different groups. According to the homomorphism of HVAs, we can easily extend our scheme to achieve batch verification efficiently.

Assume that there are k verification tasks from k different groups. For each task, the TPA launches a challenge by performing the ***challenge*** algorithm. Assume that the challenge for the ith group $(i = 1, 2, \ldots, k)$ is $chal_i = \{IDX_i, S_i, \{\beta_{i,j}|j\in D_i\}, \{\gamma_{i,j}|j\in U_{L,i}\}, \{\tau_{i,j}|j\in U_{R,i}\}\}$. Note that, to assure the aggregability of the proofs, the random numbers ε, ω and η in each task are respectively identical with the ones in the other tasks. Upon receiving these challenges, for each group (e.g. the ith one, $i = 1, 2, \ldots, k$), the CSP calculates the tag proof Φ_i and data proof $\Omega_{1,i}$ for the challenged blocks, the tag proof $_i$ and the data proof $\Omega_{2,i}$ for all the RU-blocks, and the auxiliary auditing factor Θ_i, according to Eqs. (2)–(5). Then, for the tag proofs and the auxiliary auditing factor, the TPA respectively calculates their aggregations as follows.

$$\Phi_{\mathfrak{B}} = \prod_{i=1}^{k} \Phi_i, \tag{17}$$

$$T_{\mathfrak{B}} = \prod_{i=1}^{k} T_i, \tag{18}$$

$$\Theta_{\mathfrak{B}} = \prod_{i=1}^{k} \Theta_i. \tag{19}$$

In addition, the CSP calculate the aggregation of all the post-revocation authenticators as

$$\Lambda_{\mathfrak{B}} = \prod_{i=1}^{k} \Lambda_i. \tag{20}$$

At last, the CSP sends $P_{\mathfrak{B}} = \{\Phi_{\mathfrak{B}},\ T_{\mathfrak{B}},\ \{\Omega_{1,i}, \Omega_{2,i}|i = 1, 2, \ldots, k\},\ \Lambda_{\mathfrak{B}},\ \Theta_{\mathfrak{B}}\}$ back to TPA as the response.

Upon receiving the response proofs, the TPA first verifies the post-revocation authenticators by checking the following equation:

$$\Lambda_{\mathfrak{B}} \stackrel{?}{=} T_{\mathfrak{B}}^{\eta\cdot\varepsilon/\omega}. \tag{21}$$

If Eq. (21) does not hold, the TPA terminates the verification and outputs FALSE; otherwise, he/she goes on with the verification. To verify the correctness of all the data, the TPA checks the following equation:

$$\Phi_{\mathfrak{B}}\cdot T_{\mathfrak{B}}^{\eta\cdot\varepsilon} \stackrel{?}{=} \prod_{i=1}^{k} e\left(\mu^{\mathcal{H}_i}\cdot g_1^{\Omega_i}, y_{i,1}\right)^{\omega}\cdot\Theta_{\mathfrak{B}}^{\omega\cdot(\eta+1)}, \tag{22}$$

where

$$\mathcal{H}_i = \sum_{l\in IDX_i} B_{i,l}s_{i,l} + \sum_{d\in D_i} B_{i,d}\lambda_{i,d}, B_{i,x} = H(v_{i,x}\|t_{i,x}). \tag{23}$$

and $\Omega_i = \Omega_{1,i} + \Omega_{2,i}\cdot\eta$. If Eq. (22) holds, it outputs TRUE; otherwise, FALSE.

The correctness of Eq. (21) can be demonstrated as follows.

$$\begin{aligned} T_{\mathfrak{B}}^{\eta\cdot\frac{\varepsilon}{\omega}} &= \prod_{i=1}^{k}\prod_{l\in D_i} e\left(\sigma_{i,l}, \tau_{i,j}\right)^{\beta_{i,l}\cdot\eta\cdot\frac{\varepsilon}{\omega}} \\ &= \prod_{i=1}^{k}\prod_{l\in D_i} e\left(\sigma_{i,l}, g^{\frac{\omega}{\varepsilon}\cdot\frac{a_{i,1}}{a_{i,j}}}\right)^{\lambda_{i,l}\cdot\frac{\varepsilon}{\omega}} \end{aligned}$$

$$= \prod_{i=1}^{k} \prod_{l \in D_i} e\left(\sigma_{i,l}, y_{i,j}\right)^{q_i \cdot \rho_i \cdot \theta_{i,j}}$$
$$= \prod_{i=1}^{k} \prod_{l \in D_i} e\left(\sigma_{i,l}, y_{i,j}^{q_i}\right)^{\rho_i \cdot \theta_{i,j}}$$
$$= \prod_{i=1}^{k} \Lambda_i'^{\rho_i} = \prod_{i=1}^{k} \Lambda_i = \Lambda_{\mathfrak{B}}.$$

The correctness of Eq. (22) can be demonstrated as follows.

$$\begin{aligned}
\Phi_{\mathfrak{B}} &= \prod_{i=1}^{k} \prod_{l \in IDX} e\left(\sigma_{i,l}, \gamma_{i,j}\right)^{s_{i,l}} \\
&= \prod_{i=1}^{k} \prod_{l \in IDX} e\left(\left(\mu^{B_{i,l}} \cdot g_1^{m_{i,l}}\right)^{a_{i,j}}, g^{\omega \cdot \frac{a_{i,1}}{a_{i,j}}}\right)^{s_{i,l}} \\
&= \prod_{i=1}^{k} \prod_{l \in IDX} e\left(\mu^{B_{i,l}} \cdot g_1^{m_{i,l}}, y_{i,1}\right)^{\omega \cdot s_{i,l}} \\
&= \prod_{i=1}^{k} e\left(\prod_{l \in IDX} \mu^{B_{i,l} \cdot s_{i,l}} \cdot g_1^{\sum_{l \in IDX} m_{i,l} \cdot s_{i,l}}, y_{i,1}\right)^{\omega} \\
&= \prod_{i=1}^{k} e\left(\prod_{l \in IDX} \mu^{\sum_{l \in IDX} B_{i,l} \cdot s_{i,l}} \cdot g_1^{\Omega_{1,i} - r_i}, y_{i,1}\right)^{\omega}.
\end{aligned}$$

$$\begin{aligned}
T_{\mathfrak{B}}^{\eta \cdot \varepsilon} &= \prod_{i=1}^{k} \prod_{d \in D} e\left(\sigma_{i,d}, \tau_{i,j}\right)^{\beta_{i,d} \cdot \eta \cdot \varepsilon} \\
&= \prod_{i=1}^{k} \prod_{d \in D} e\left(\left(\mu^{B_{i,d}} \cdot g_1^{m_{i,d}}\right)^{a_{i,j}}, g^{\frac{\omega}{\varepsilon} \cdot \frac{a_{i,1}}{a_{i,j}}}\right)^{\lambda_{i,d} \cdot \varepsilon} \\
&= \prod_{i=1}^{k} \prod_{d \in D} e\left(\mu^{B_{i,d}} \cdot g_1^{m_{i,d}}, y_{i,1}\right)^{\omega \cdot \lambda_{i,d}} \\
&= \prod_{i=1}^{k} e\left(\prod_{d \in D} \mu^{B_{i,d} \cdot \lambda_{i,d}} \cdot g_1^{\sum_{d \in D} m_{i,d} \cdot \lambda_{i,d}}, y_{i,1}\right)^{\omega} \\
&= \prod_{i=1}^{k} e\left(\prod_{d \in D} \mu^{\sum_{d \in D} B_{i,d} \cdot \lambda_{i,d}} \cdot g_1^{(\Omega_{2,i} - r_i) \cdot \eta}, y_{i,1}\right)^{\omega}.
\end{aligned}$$

Further, we have

$$\begin{aligned}
&\Phi_{\mathfrak{B}} \cdot T_{\mathfrak{B}}^{\eta \cdot \varepsilon} \\
&= \prod_{i=1}^{k} e\left(\prod_{l \in IDX} \mu^{\sum_{l \in IDX} B_{i,l} \cdot s_{i,l}} \cdot g_1^{\Omega_{1,i} - r_i}, y_{i,1}\right)^{\omega} \\
&\quad \cdot \prod_{i=1}^{k} e\left(\prod_{d \in D} \mu^{\sum_{d \in D} B_{i,d} \cdot \lambda_{i,d}} \cdot g_1^{(\Omega_{2,i} - r_i) \cdot \eta}, y_{i,1}\right)^{\omega} \\
&= \prod_{i=1}^{k} e\left(\mu^{\sum_{l \in IDX} B_{i,l} \cdot s_{i,l} + \sum_{d \in D} B_{i,d} \cdot \lambda_{i,d}} \cdot g_1^{\Omega_{1,i} + \Omega_{2,i} \cdot \eta - (1+\eta) \cdot r_i}\right)^{\omega} \\
&= \prod_{i=1}^{k} \left\{ e\left(\mu^{\mathcal{H}_i} \cdot g_1^{\Omega_i}, y_{i,1}\right)^{\omega} \cdot e(g_1, y_{i,1})^{-\omega \cdot (1+\eta) \cdot r_i} \right\}
\end{aligned}$$

$$= \prod_{i=1}^{k} \left\{ e\left(\mu^{\mathcal{H}_i} \cdot g_1^{\Omega_i}, y_{i,1}\right)^{\omega} \cdot \Theta_i^{\omega\cdot(1+\eta)} \right\}$$
$$= \prod_{i=1}^{k} e\left(\mu^{\mathcal{H}_i} \cdot g_1^{\Omega_i}, y_{i,1}\right)^{\omega} \cdot \Theta_{\mathfrak{B}}^{\omega\cdot(1+\eta)}.$$

5. Security analysis

Theorem 1 (Unforgeability of the post-revocation authenticator)**.** *In the proposed scheme, it is computationally infeasible for the CSP to forge a valid post-revocation authenticator.*

Proof. As mentioned in the lazy revocation strategy, if a user u_x has been revoked, the group manager should generate the post-revocation authenticator as

$$\Lambda = \Lambda'^{\rho} = \left(\prod_{i\in D} e\left(\sigma_i, y_x^q\right)^{\theta_i} \right)^{\rho},$$

where ρ is unknown to the CSP. If the CSP wants to forge a post-revocation authenticator, he/she should first know ρ. However, according to the DL assumption, it is computationally intractable to compute the value $\rho \in \mathbb{Z}_p^*$, such that $\Lambda = \Lambda'^{\rho}$. That is, the CSP cannot forge a valid post-revocation authenticator. □

Theorem 2 (Immunity of forging attacks)**.** *The proposed scheme can effectively resist forging attacks. That is, it is infeasible for the CSP to forge valid proofs to pass the verification, even if a revoked user colludes with the CSP.*

Proof. To response the challenge $chal = \{IDX, S, \{\beta_i | i \in D\}, \{\gamma_i | i \in U_L\}, \{\tau_i | i \in U_R\}\}$, the CSP should provide a proof message $P = \{\Phi, T, \Omega_1, \Omega_2, \Lambda, \Theta\}$. If Θ is incorrect, Eq. (8) does not hold, even if the other proofs are right. Thus, the CSP cannot give a wrong Θ. In fact, it is also meaningless for the CSP to provide a fake Θ. As mentioned in Theorem 1, the post-revocation authenticator cannot also be forged. Moreover, it has been proved that BLS-HVAs are unforgeable [27,33] if the private key of the user is not known, so Φ, the aggregation of all the tags for the challenged blocks, cannot be forged too. In this sense, we only need to prove the unforgeability of the tag proof of the RU-blocks T, the data proof of the challenged blocks Ω_1 and the data proof of the RU-blocks Ω_2.

1. Unforgeability of T
   To prove this, we design a game as follows: to response the challenge, the CSP provides a forged proof message $P' = \{\Phi, T', \Omega_1, \Omega_2, \Lambda, \Theta\}$, where
   $$T = \prod_{i\in D} e\left(\sigma_i, \tau_j\right)^{\beta_i} \neq T' = \prod_{i\in D} e\left(\sigma_i', \tau_j\right)^{\beta_i}. \tag{24}$$
   Eq. (24) suggests that $\exists i \in D,\ \sigma_i \neq \sigma_i'$. If the CSP still passes the verification, he/she wins; otherwise, he/she fails. Suppose that the CSP wins the game, then,
   $$\Lambda = T'^{\eta\cdot\frac{\varepsilon}{\omega}} = \prod_{i\in D} e\left(\sigma_i', \tau_j\right)^{\lambda_i\cdot\frac{\varepsilon}{\omega}}. \tag{25}$$
   For the correct proofs, we have
   $$\Lambda = T^{\eta\cdot\frac{\varepsilon}{\omega}} = \prod_{i\in D} e\left(\sigma_i, \tau_j\right)^{\lambda_i\cdot\frac{\varepsilon}{\omega}}. \tag{26}$$
   According to the properties of bilinear maps, we can learn that $\sigma_i = \sigma_i', \forall i \in D$, which apparently contradicts the assumption. That is, T is unforgeable.
2. Unforgeability of Ω_1 and Ω_2
   We design another game as follows: to response the challenge, the CSP provides a forged proof message $P'' = \{\Phi, T, \Omega_1', \Omega_2', \Lambda, \Theta\}$, where
   $$\Omega = \Omega_1 + \Omega_2 \cdot \eta \neq \Omega' = \Omega_1' + \Omega_2' \cdot \eta$$
   $$\Rightarrow \sum_{i\in IDX} m_i s_i + \sum_{j\in D} m_j \lambda_j \neq \sum_{i\in IDX} m_i' s_i + \sum_{j\in D} m_j' \lambda_j.$$

Table 3
Comparison of communication costs .

Schemes	Challenge	Response
HHY15 [11]	$O(c)$	$O(1)$
Panda [31]	$O(c)$	$O(n_u + c)$
YYS16 [36]	$O(c)$	$O(1)$
YY15 [41]	$O(c)$	$O(1)$
PASCD	$O(c + c_r) + O(n_u)$	$O(1)$

Note: s is the number of segments that a data block is divided into, c is the number of the challenged blocks, c_r is the number of RU-blocks, n_u is the number of the valid users.

If the CSP still passes the verification, he/she wins; otherwise, he/she fails. Suppose that the CSP wins the game, then,

$$e\left(\mu^{\sum_{i\in IDX} B_i s_i + \sum_{j\in D} B_j \lambda_j} \cdot g_1^{\Omega'}, y_1\right) \cdot \Theta^{\omega\cdot(\eta+1)}$$
$$= e\left(\mu^{\sum_{i\in IDX} B_i s_i + \sum_{j\in D} B_j \lambda_j} \cdot g_1^{\sum_{i\in IDX} m_i' s_i + \sum_{j\in D} m_j' \lambda_j}, y_1\right) \cdot \Theta^{\omega\cdot(\eta+1)}.$$

For the correct proofs, we have

$$e\left(\mu^{\sum_{i\in IDX} B_i s_i + \sum_{j\in D} B_j \lambda_j} \cdot g_1^{\Omega}, y_1\right) \cdot \Theta^{\omega\cdot(\eta+1)}$$
$$= e\left(\mu^{\sum_{i\in IDX} B_i s_i + \sum_{j\in D} B_j \lambda_j} \cdot g_1^{\sum_{i\in IDX} m_i s_i + \sum_{j\in D} m_j \lambda_j}, y_1\right) \cdot \Theta^{\omega\cdot(\eta+1)}.$$

According to the properties of bilinear maps, we can learn that

$$\sum_{i\in IDX} m_i s_i + \sum_{j\in D} m_j \lambda_j = \sum_{i\in IDX} m_i' s_i + \sum_{j\in D} m_j' \lambda_j,$$

which contradicts the assumption. That is, Ω_1 and Ω_2 are unforgeable.

In a word, the CSP cannot forge any valid proofs to pass the verification. □

6. Performance evaluation

In this section, we will evaluate the performance of our scheme (PASCD) and compare it with the state-of-the-art schemes [11,31,36,41] using individual signature keys.

6.1. Communication costs

Table 3 shows the communication costs of PASCD and the other state-of-the-art schemes (i.e., HHY15 [11], Panda [31], YYS16 [36], YY15 [41]) in the challenge and response phases, from which we can learn that, for the challenge, the communication costs of the existing schemes are $O(c)$, while the ones of PASCD are $O(c + c_r) + O(n_u)$, a slightly larger than $O(c)$. The reasons for this are twofold. First, to resist the collusion attack, PASCD verifies the integrity of the tags of all the RU-blocks, of which the communication cost is $O(c_r)$. Second, to achieve the identity privacy, some additional information (i.e., $\{\gamma_i | i \in U_L\}$, $\{\tau_i | i \in U_R\}$) should be sent to the CSP for hiding the actual operators of the data blocks, of which the communication cost is $O(n_u)$. Thus, we prefer to trade slightly larger communication costs for security and privacy. Moreover, in the response phase, the communication costs of YY15, HHY15, YYS16 and PASCD is $O(1)$, while the ones of Panda are $O(n_u) + O(c)$, because it needs to generate the data proof and tag proof for each user as well as provide the identifiers of all the challenged blocks and the identifiers of their signer. In this sense, Panda cannot preserve the identity privacy for the group users.

6.2. Computational costs

In this section, we further evaluate the computational costs of PASCD and compare them with Panda [31] and YY15 [41] (we do not include HHY15 [11] and YYS16 [36], since they cannot provide mechanisms for user revocation). We implement the prototype systems of PASCD, Panda and YY15 based on the C++ version of the Pairing-Based Cryptography (PBC) Library (0.5.14). The algorithms involved in the schemes are evaluated on an HP workstation with an Intel Core i5-6500 CPU at 3.20 GHz, 2 × 4GB DDR4 2133 MHz RAM, and 7200 RPM 2TB Serial ATA drive with a 64 MB buffer. We employ an MNT d159 curve, which has a 160-bit group order. That is, the length of the prime order p in the experiments is 160 bits. Moreover, in all experiments, the size of each data block is 4KB. All the statistical results are the averages of 20 trials.

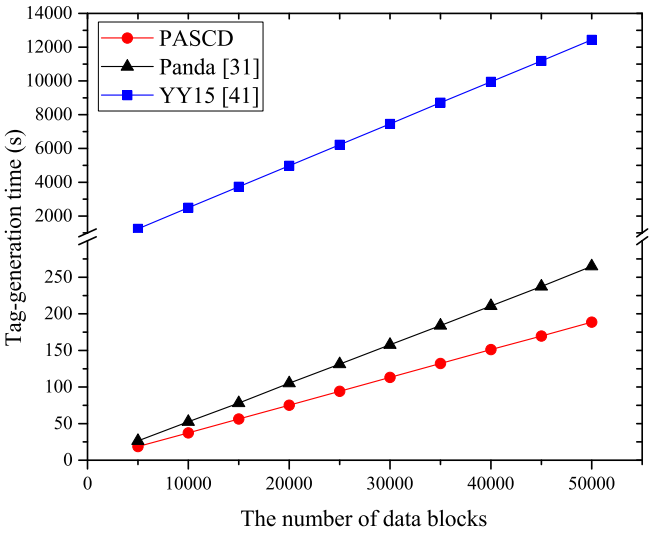

Fig. 6. Time of tag generation for different numbers of blocks.

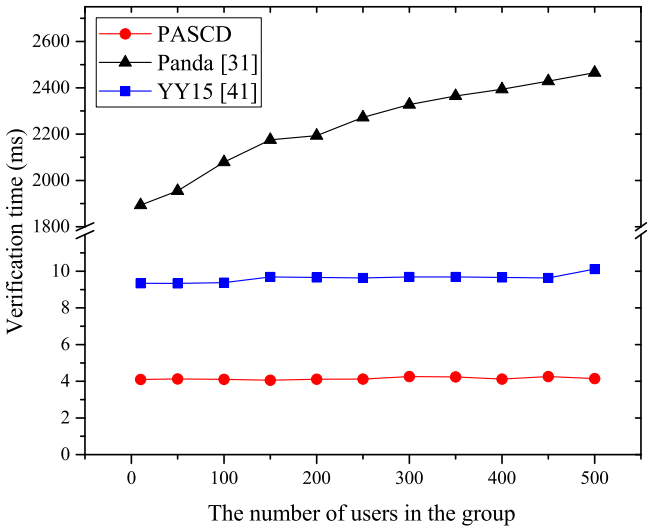

Fig. 7. Verification time for various numbers of users in the group.

6.2.1. Computational costs for generating tags

Fig. 6 shows the experimental results of the tag-generation time for the different numbers of data blocks with the same size, from which we can learn that: (1) in all the three schemes, the time for generating tags is proportional to the block number; and (2) for the same number of data blocks, PASCD spends less time than Panda, and two orders of magnitude less than YY15. That is to say, the computational costs of the user in PASCD are the fewest.

6.2.2. Computational costs for verification

Fig. 7 shows the experimental results of the verification time for various numbers of users in the group, from which we can learn that the verification time in PASCD and YY15 is independent of the user number, while the verification time of Panda is proportional to the user number. Moreover, for the same number of users in the group, the verification time of PASCD is less than the half of that of YY15, and two orders of magnitude less than Panda.

Fig. 8 shows the experimental results of the verification time for the different numbers of RU-blocks, from which we can learn that the verification time in Panda and YY15 is irrelevant with the number of RU-blocks, while the verification time of PASCD increases slightly with the number of RU-blocks. Moreover, for the same number of RU-blocks, the verification time of PASCD is apparently less than YY15, and two orders of magnitude less than Panda.

In a word, compared with Panda and YY15, PASCD reduces the computational costs significantly in the verification phase.

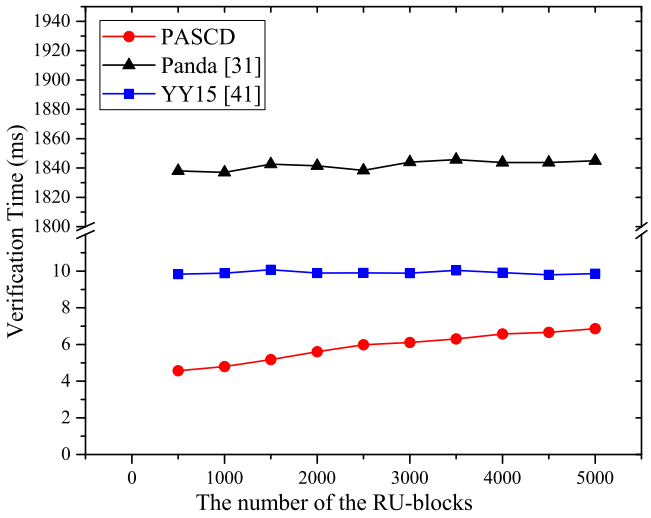

Fig. 8. Verification time for the different numbers of RU-blocks.

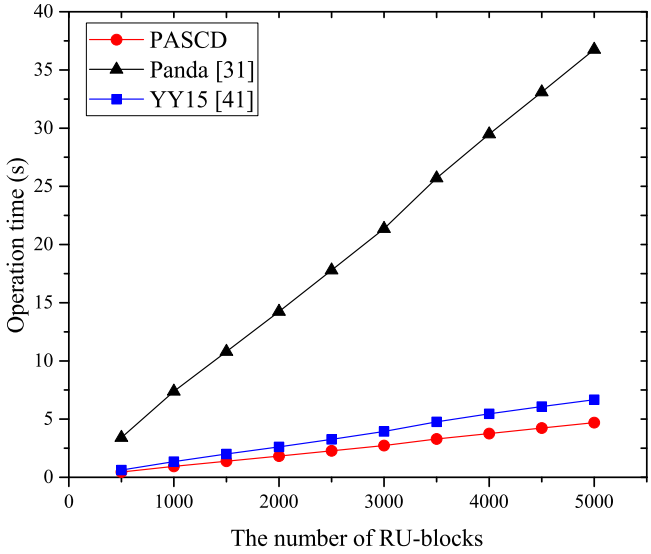

Fig. 9. The operation time for the different numbers of RU-blocks in the user revocation phase.

6.2.3. Computational costs for user revocation

Fig. 9 shows the experimental results of operation time for the different numbers of RU-blocks in the user revocation phase, from which we can learn that: 1) in all the three schemes, the operation time for user revocation is proportional to the numbers of RU-blocks; and 2) for the same number of RU-blocks, PASCD spends less time than Panda and YY15, which indicates that the proposed lazy revocation strategy not only provides excellent security but also outperforms the revocation approaches in Panda and YY15 in computational overhead.

6.2.4. Computational costs for batch verification

We also evaluate the performance of PASCD in the batch auditing scenario and compare it with Panda and YY15. The experimental results, as shown in Fig. 10, suggests that: (1) all the three schemes can simultaneously handle different verifications from multiple-groups; and (2) for the same number of auditing tasks, the average verification time per task in PASCD is less than that in YY15, and three orders of magnitude less than that in Panda. That is, the batch auditing protocol in PASCD is more efficient than those in Panda and YY15.

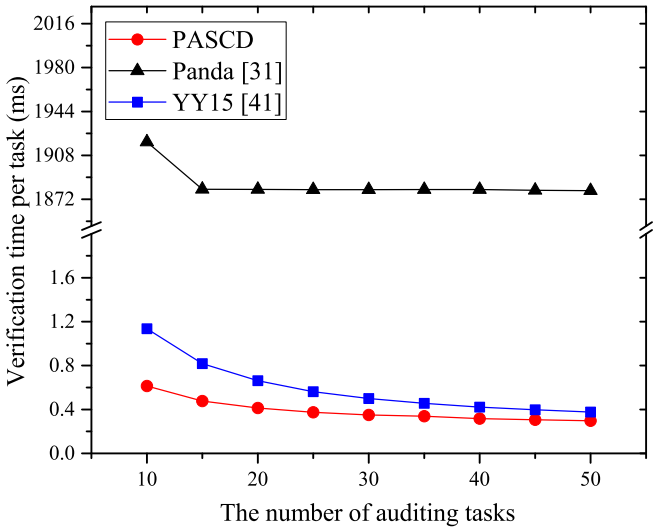

Fig. 10. The verification time per task for the different numbers of auditing tasks.

7. Conclusion

The cloud storage auditing, as an indispensable technique for cloud storage security, has been attracting more and more attention. More recently, with the increasing popularity of collaboration and teamwork in the cloud, shared data auditing has become a new hot topic in this field. Although there have been some fruitful auditing schemes for shared data, none could fully achieve all the security and performance requirements. Therefore, in this paper, we are motivated to present a new public auditing scheme for shared data. To be specific, to achieve public auditing, we adopt the BLS signature technique to generate homomorphic authenticators; to preserve the identity privacy of users, the signatures of different users on the challenged blocks are converted to the ones signed by the group manager during the proof generation exploiting the properties of bilinear maps; to protect data privacy, the random masking is employed to blind the data proofs; to support identity traceability, a modification record table is designed to record the operation information for each modified data block; and the proposed scheme extends dynamic hash table to support shared-data dynamics, and utilizes the aggregate signature technique to achieve batch auditing. In addition, to achieve efficient group dynamics, a novel group management mechanism is designed. Particularly, we present a lazy-revocation strategy to achieve user revocation in a secure and cost-effective manner. The key idea is that, during the user revocation, the group manager only needs to generate a post-revocation authenticator to protect the blocks related to the revoked user, and the tag regenerations of the related blocks are postponed until they are modified by the other users. Compared with the existing schemes, the lazy strategy cannot only resist the collusion attack but also reduce the computational costs during user revocation. We formally prove the security of the proposed scheme, and evaluate the performance by detailed experiments and comparisons with the existing ones. The results demonstrate that the proposed scheme can effectively achieve secure auditing for shared cloud data, and outperforms the previous ones in computational overhead while maintaining relatively low communication costs.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China under Grant Nos. U1405254 and U1536115, Natural Science Foundation of Fujian Province of China under Grant No. 2018J01093, Program for New Century Excellent Talents in Fujian Province University under Grant No. MJK2016-23, and Program for Outstanding Youth Scientific and Technological Talents in Fujian Province University under Grant No. MJK2015-54.

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, Provable data possession at untrusted stores, in: Proceedings of 14th ACM Conference on Computer and Communications (ACM CCS '07), Alexandria, Virginia, USA, 2007, pp. 598–609.
[2] M. Backes, C. Cachin, A. Oprea, Lazy revocation in cryptographic file systems, in: Proceedings of the 3rd IEEE International Security in Storage Workshop (SISW '05), 2005, p. 11.
[3] A.F. Barsoum, M.A. Hasan, Provable multicopy dynamic data possession in cloud computing systems, IEEE Trans. Inf. Forensics Secur. 10 (3) (2015) 485–497.
[4] Z. Cai, H. Yan, P. Li, Z.-a. Huang, C. Gao, Towards secure and flexible EHR sharing in mobile health cloud under static assumptions, Cluster Comput. 20 (3) (2017) 2415–2422.
[5] X. Chen, J. Li, X. Huang, J. Ma, W. Lou, New publicly verifiable databases with efficient updates, IEEE Trans. Dependable Secure Comput. 12 (5) (2015) 546–556.

[6] X. Chen, J. Li, J. Weng, J. Ma, W. Lou, Verifiable computation over large database with incremental updates, IEEE Trans. Comput. 65 (10) (2016) 3184–3195.
[7] R. Curtmola, O. Khan, R. Burns, G. Ateniese, MR-PDP: multiple-replica provable data possession, in: Proceedings of the 28th International Conference on Distributed Computing Systems, 2008, pp. 411–420.
[8] C. Erway, A. Küpçü, C. Papamanthou, R. Tamassia, Dynamic provable data possession, in: Proceedings of the 16th ACM Conference on Computer and Communications (ACM CCS '09), New York, NY, USA, 2009, pp. 213–222.
[9] C. Gao, S. Lv, Y. Wei, Z. Wang, Z. Liu, X. Cheng, M-SSE: an effective searchable symmetric encryption with enhanced security for mobile devices, IEEE Access 6 (2018) 38860–38869, doi:10.1109/ACCESS.2018.2852329.
[10] Z. Hao, S. Zhong, N. Yu, A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability, IEEE Trans. Knowl. Data Eng. 23 (9) (2011) 1432–1437.
[11] K. He, C. Huang, K. Yang, J. Shi, Identity-preserving public auditing for shared cloud data, in: Proceedings of the 23rd IEEE International Symposium on Quality of Service (IWQoS '15), 2015, pp. 159–164.
[12] T. Jiang, X. Chen, J. Ma, Public integrity auditing for shared dynamic cloud data with group user revocation, IEEE Trans. Comput. 65 (8) (2016) 2363–2373.
[13] A. Juels, B.S. Kaliski Jr., Pors: proofs of retrievability for large files, in: Proceedings of the 14th ACM Conference on Computer and Communications Security (ACM CCS '07), Alexandria, Virginia, USA, 2007, pp. 584–597.
[14] M. Kolhar, M.M. Abu-Alhaj, S.M.A. El-atty, Cloud data auditing techniques with a focus on privacy and security, IEEE Secur. Privacy 15 (1) (2017) 42–51.
[15] J. Li, X. Chen, S.S.M. Chow, Q. Huang, D.S. Wong, Z. Liu, Multi-authority fine-grained access control with accountability and its application in cloud, J. Netw. Comput. Appl. 112 (2018) 89–96.
[16] J. Li, X. Huang, J. Li, X. Chen, Y. Xiang, Securely outsourcing attribute-based encryption with checkability, IEEE Trans. Parallel Distrib. Syst. 25 (8) (2014) 2201–2210.
[17] J. Li, Z. Liu, X. Chen, F. Xhafa, X. Tan, D.S. Wong, L-EncDB: a lightweight framework for privacy-Preserving data queries in cloud computing, Knowl. Based Syst. 79 (2015) 18–26.
[18] T. Li, J. Li, Z. Liu, P. Li, C. Jia, Differentially private naive bayes learning over multiple data sources, Inf. Sci. 444 (2018) 89–104.
[19] Q. Lin, H. Yan, Z. Huang, W. Chen, J. Shen, Y. Tang, An ID-based linearly homomorphic signature scheme and its application in blockchain, IEEE Access 6 (2018) 20632–20640.
[20] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, J. Chen, MuR-DPA: top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud, IEEE Trans. Comput. 64 (9) (2015) 2609–2622.
[21] C. Liu, C. Yang, X. Zhang, J. Chen, External integrity verification for outsourced big data in cloud and IoT: a big picture, Future Gener. Comput. Syst. 49 (2015) 58–67.
[22] Z. Liu, Y. Huang, J. Li, X. Cheng, C. Shen, Divoram: towards a practical oblivious RAM with variable block size, Inf. Sci. 447 (2018) 1–11.
[23] K. Ren, C. Wang, Q. Wang, Security challenges for the public cloud, IEEE Internet Comput. 16 (1) (2012) 69–73.
[24] R.L. Rivest, K. Fu, K.E. Fu, Group Sharing and Random Access in Cryptographic Storage File Systems, Massachusetts Institute of Technology, 1999 Doctoral dissertation.
[25] F. Sebé, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, J.J. Quisquater, Efficient remote data possession checking in critical information infrastructures, IEEE Trans. Knowl. Data Eng. 20 (8) (2008) 1034–1038.
[26] M. Sookhak, A. Gani, H. Talebian, A. Akhunzada, S.U. Khan, R. Buyya, A.Y. Zomaya, Remote data auditing in cloud computing environments: a survey, taxonomy, and open issues, ACM Comput. Surv. 47 (4) (2015) 65:1–65:34.
[27] H. Tian, Y. Chen, C.C. Chang, H. Jiang, Y. Huang, Y. Chen, J. Liu, Dynamic-hash-table based public auditing for secure cloud storage, IEEE Trans. Serv. Comput. 10 (5) (2017) 701–714.
[28] H. Tian, Z. Chen, C.-C. Chang, Y. Huang, T. Wang, Z.-a. Huang, Y. Cai, Y. Chen, Public audit for operation behavior logs with error locating in cloud storage, Soft Comput. (2018) 1–14.
[29] H. Tian, Z. Chen, C.-C. Chang, M. Kuribayashi, Y. Huang, Y. Cai, Y. Chen, T. Wang, Enabling public auditability for operation behaviors in cloud storage, Soft Comput. 21 (8) (2017) 2175–2187.
[30] B. Wang, B. Li, H. Li, Oruta: privacy-preserving public auditing for shared data in the cloud, IEEE Trans. Cloud Comput. 2 (1) (2014) 43–56.
[31] B. Wang, B. Li, H. Li, Panda: public auditing for shared data with efficient user revocation in the cloud, IEEE Trans. Serv. Comput. 8 (1) (2015) 92–106.
[32] C. Wang, S.S.M. Chow, Q. Wang, K. Ren, W. Lou, Privacy-preserving public auditing for secure cloud storage, IEEE Trans. Comput. 62 (2) (2013) 362–375.
[33] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, Enabling public auditability and data dynamics for storage security in cloud computing, IEEE Trans. Parallel Distrib. Syst. 22 (5) (2011) 847–859.
[34] Z. Wu, L. Tian, P. Li, T. Wu, M. Jiang, C. Wu, Generating stable biometric keys for flexible cloud computing authentication using finger vein, Inf. Sci. 433–434 (2018) 431–447.
[35] J. Xu, L. Wei, Y. Zhang, A. Wang, F. Zhou, C.-z. Gao, Dynamic fully homomorphic encryption-Based merkle tree for lightweight streaming authenticated data structures, J. Netw. Comput. Appl. 107 (2018) 113–124.
[36] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, R. Hao, Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability, J. Syst. Softw. 113 (2016) 130–139.
[37] K. Yang, X. Jia, An efficient and secure dynamic auditing protocol for data storage in cloud computing, IEEE Trans. Parallel Distrib. Syst. 24 (9) (2013) 1717–1726.
[38] Y. Yu, M.H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, G. Min, Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage, IEEE Trans. Inf. Forensics Secur. 12 (4) (2017) 767–778.
[39] Y. Yu, J. Ni, Q. Xia, X. Wang, H. Yang, X. Zhang, SDIVIP2: shared data integrity verification with identity privacy preserving in mobile clouds, Concurrency Comput. 28 (10) (2016) 2877–2888.
[40] J. Yuan, S. Yu, Efficient public integrity checking for cloud data sharing with multi-user modification, in: Proceedings of IEEE Conference on Computer Communications (IEEE INFOCOM '14), 2014, pp. 2121–2129.
[41] J. Yuan, S. Yu, Public integrity auditing for dynamic data sharing with multiuser modification, IEEE Trans. Inf. Forensics Secur. 10 (8) (2015) 1717–1726.
[42] Y. Zhu, G.J. Ahn, H. Hu, S.S. Yau, H.G. An, C.J. Hu, Dynamic audit services for outsourced storages in clouds, IEEE Trans. Serv. Comput. 6 (2) (2013) 227–238.