Certificateless Public Integrity Checking of Group Shared Data on Cloud Storage

Jiguo Li, Hao Yan, and Yichen Zhang

Abstract—Cloud storage service supplies people with an efficient method to share data within a group. The cloud server is not trustworthy, so lots of remote data possession checking (RDPC) protocols are proposed and thought to be an effective way to ensure the data integrity. However, most of RDPC protocols are based on the mechanism of traditional public key infrastructure (PKI), which has obvious security flaw and bears big burden of certificate management. To avoid this shortcoming, identity-based cryptography (IBC) is often chosen to be the basis of RDPC. Unfortunately, IBC has an inherent drawback of key escrow. To solve these problems, we utilize the technique of certificateless signature to present a new RDPC protocol for checking the integrity of data shared among a group. In our scheme, user's private key includes two parts: a partial key generated by the group manager and a secret value chosen by herself/himself. To ensure the right public keys are chosen during the data integrity checking, the public key of each user is associated with her unique identity, for example the name or telephone number. Thus, the certificate is not needed and the problem of key escrow is eliminated too. Meanwhile, the data integrity can still be audited by public verifier without downloading the whole data. In addition, our scheme also supports efficient user revocation from the group. The security of our scheme is reduced to the assumptions of computational Diffie-Hellman (CDH) and discrete logarithm (DL). Experiment results exhibit that the new protocol is very efficient and feasible.

Index Terms—Remote data checking, cloud storage, certificateless signature, data shared in group

1 Introduction

Cloud storage service offers user an efficient way to share data and work as a team. Once someone of the team uploads a file to the server, other members are able to access and modify the file by Internet. Many real applications such as Dropbox for Business [1] and TortoiseSVN [2] are used in many companies for their staff to work together. The most important problem of such applications is whether the cloud server provider (CSP) can ensure the data to be kept intact [3]. In fact, the CSP is not fully trustworthy and the failure of software or hardware is inevitable in some way, so serious accidents of the data corruption may occur at any time. Therefore, the user needs to audit the CSP to confirm the data on the cloud server is original.

To ensure the integrity of stored data, a great number of RDPC schemes are proposed [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32]. In these schemes, each data block generates an authentication tag which is bound with the block. By checking the correctness of the tags, the verifier is able to learn the status of the data. However, most of these schemes only focus on checking the integrity for personal data [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [29], [30], [31], [32], which is not valid under the situation of data shared in a group. When data is shared among multiple users, some new challenges appear which are not well solved in the RDPC schemes for personal data. For example, block tags may be generated by any group user, and different group user will output different tags even if the block is the same one. Moreover, when a group user updates a block, it should regenerate the tag again. When auditing the data integrity, all the authentication tags generated individually need to be aggregated and the information of all the generators for these tags will be involved in. It brings great complexity for the checking scheme. Furthermore, the group is dynamic, any group member may initiatively leave or be fired from the group at any time, so the user revocation is also an important problem that must be addressed. More specifically, once a user is revoked, he should not be allowed to access or modify the data and all his public/private keys are invalid. Under this situation, it is impossible to check the correctness of the tags made by revoked user. Thus, all the tags made by revoked user should be renewed by other normal user. The traditional method is to download the blocks signed by revoked user from the CSP, calculate the new tags and upload the new tags to the cloud again. It will increase heavy computation and communication cost for the normal user. Therefore, this task should be performed by the CSP rather than the normal user. How to design an efficient and secure method to outsource the task is a challenge issue. Besides, public verification is an

- *J. Li is with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China 350117, and also with State Key Laboratory of Cryptology, P.O. Box 5159, Beijing, 100878, China, Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing University of Posts and Telecommunications, China, College of Computer and Information, Hohai University, Nanjing, China 211100. E-mail: ljg1688@163.com.*
- *H. Yan is with the College of Computer and Information, Hohai University, Nanjing, Jiangsu 211100, China. E-mail: pxy_hao@163.com.*
- *Y. Zhang is with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, Fujian 350117, China, and with the College of Computer and Information, Hohai University, Nanjing, Jiangsu 211100, China. E-mail: zyc_718@163.com.*

Manuscript received 16 Apr. 2017; revised 22 Dec. 2017; accepted 2 Jan. 2018. Date of publication 8 Jan. 2018; date of current version 3 Feb. 2021. (Corresponding author: Jiguo Li.)
Digital Object Identifier no. 10.1109/TSC.2018.2789893

attractive feature of the data integrity checking work. That is, the integrity of shared data can be verified by not only the data owner but also everyone who is interested in the cloud data. It is very important for RDPC protocol to support public verification under current open environment.

Until now, lots of schemes [22], [23], [24], [25], [26], [27], [28] have been presented for the integrity verification of data shared in group. However, most of existing RDPC schemes [22], [23], [24], [25], [26], [28] are based on PKI. Although PKI is widely used and occupies an important position in public key cryptography, there are still some security threats in it. For example, the security of PKI is based on the trustworthy of certificate authority (CA), but it is not an easy work to ensure the trustworthiness of CA. Besides, the management of certificate such as distribution, storage, revocation and verification is also a big burden. To avoid these problems, some ID-based RDPC schemes [27], [28] are proposed. Unfortunately, ID-based RDPC schemes suffer from key escrow problem. Namely, the private key generator (PKG) generates all the private keys for the users. If PKG is untrusted, the scheme is not secure either. Thus, ID-based RDPC schemes may be restricted to small, closed settings. Compared with PKI and IBC, certficateless cryptography [33] solves the problems of certificate management and key escrow at the same time. To construct certificateless RDPC scheme is a good method for cloud data integrity checking.

1.1 Motivation and Contributions

In this paper, we mainly focus on the integrity checking for data shared within a group. Suppose there is a scenario that a software engineer starts an open source project and calls on volunteers from the world to join the project. They work as a temporary team. All the codes of the project are stored on certain cloud server so that all the team members upload and modify the source code by Internet. The team may be very big, so it should be set up and managed efficiently. The volunteers may leave the team at any time, so the problem of user revocation from the team should be considered. The most important thing is that there need some way to guarantee the integrity of source codes on cloud sever.

Motivated by such requirement, we propose a new RDPC scheme for data shared in a group. Different from previous work, our scheme is based on the certifcateless signature technique to avoid the problems of certificate management and key escrow. In our scheme, the group creator generates the partial key for each group user on behalf of key generation centre. Each user selects a secret value privately. The private key of each group user contains two parts: a partial key and a secret value. All the data blocks are signed by group user to get corresponding authentication tags. During the data verification, all the tags are aggregated to decrease the computation and communication cost. Based on CDH and DL assumptions, we prove the security of our scheme. Besides, our scheme supports public verification and efficient user revocation. We implement our scheme and perform some experiments. The experiment results indicate that our scheme has good efficiency.

1.2 Related Work

The first RDPC protocol for remote data checking was proposed by Deswarte et al. [4], in which a RSA-based hash function was utilized to generate the authentication tag of the data. Following it, a great number of provable data possession (PDP) [5] and proof of retrievability (POR) [29] schemes were proposed to solve the issue for data integrity verification.

Ateniese et al. [5] first presented PDP model and initially introduced the technique of probabilistic integrity checking for the remote data. However, the first PDP scheme was only suitable for static data. To meet dynamic operations of the data block, Ateniese et al. [6] proposed another scalable and efficient PDP scheme by symmetric encryption, which supported block appending, updating and deleting. Sebé et al. [7] presented a PDP protocol based on the hard problem of factoring large integers. Erway et al. [8] utilized the authenticated skip list to provide a fully dynamic PDP scheme, which supported data owner to insert, append, modify and delete data blocks.

Based on the technique of random masking and the homomorphic linear authenticator, Wang et al. [9] presented a public verification PDP scheme with property of privacy-preserving. To support the public auditability and data dynamics, Wang et al. [10] utilized merkle hash tree (MHT) to present a dynamic scheme for cloud data checking. The scheme was fully dynamic and allowed anyone to verify the file integrity with public keys. MHT was also used in schemes [11], [12] to implement data dynamic. However, due to the computation complexity of the MHT, this scheme caused heavy computation cost and communication cost. To overcome this shortcoming, Yang and Jia [13] introduced a linear index table to support data dynamic. Yan et al. [14] further optimized the implementation of linear index table and provided an efficient RDPC scheme. Feng et al. [15] presented a public remote integrity checking scheme, which could protect the user identity on file level to reduce the storage and communication cost.

Zhu et al. [16] provided a cooperative PDP scheme for the multi-cloud setting, in which the data blocks were stored on different cloud servers. To improve the security, Wang [17] proposed another identity-based PDP scheme for multi-cloud setting without certificate management. Recently, Wang et al. [18] presented an incentive and unconditionally anonymous identity-based public PDP scheme. In order to reduce the computation cost of data owner, Wang et al. [19] presented a proxy-oriented PDP scheme which moved the work of tag generation from data owner to proxy. To address the problem of key escrow and certificate management, two PDP scheme based on certificateless [20] and certificate-based cryptography [21] were proposed respectively.

All the schemes mentioned above focused on the integrity verification for personal data. In 2012, Wang et al. [22] proposed a protocol for checking the integrity of data shared in a group. They utilized the technique of group signature to generate each authentication tag so as to preserve the tag generator's privacy. Wang et al. [23] proposed another PDP scheme for group data which supported the group user's joining and leaving. Based on broadcast encryption and group signature techniques, Liu et al. [24] provided a PDP scheme for group data. To improve the efficiency, Wang et al. [25] presented another scheme based on ring signature technique. However, these two schemes didn't solve the problem of user revocation. To address this issue, Wang et al. [26] used proxy re-signature technique to propose a new scheme with user revocation. Yu et al. [27] presented a

PDP scheme without paring, which also supported dynamic group. Yuan and Yu [28] proposed a PDP scheme based on polynomial-based authentication tags, which aimed to solve the problem of multi-user modification for blocks. All these schemes rely on traditional PKI mechanism which has security risk and big burden for certificate management. Besides, schemes of [27], [28] also suffer from the problem of key escrow. Thus, there still exists big limitation for the schemes to be used in real applications.

PoR [29] is another direction for auditing the integrity of remote data on cloud server. To enhance the efficiency, Shacham and Waters [30] proposed two compact PoR schemes based on technique of short signature [34]. Furthermore, lots of PoR schemes [31], [32] were presented for higher efficiency or better security.

With the development of cloud computing, how to share the data from the cloud server attracts more concern. In order to ensure the security and privacy, and obtain flexibly fine-grained file access control, attribute based encryption (ABE) scheme [35], [36], [37], [38], [39], [40], [41], [42] as a new cryptographic primitive was presented and applied in cloud storage system. In ABE scheme, an encryptor associates ciphertext with a set of attributes. The authority issues the users' different private keys that are associated with access policy on attributes. Li et al. [35], [36] modeled collusion attack executed by existing users with revoked users and constructed efficient CP-ABE schemes with user revocation. Recently Wang et al. [37] provided an anonymous distributed fine-grained access control scheme with verifiably outsourced decryption in public cloud. In ABE scheme, sensitive documents should be encrypted prior to outsourcing for privacy requirements, which hinders efficient query processing like keyword-based document retrieval. In order to address this issue, Li et al. [38], [39] presented ABE schemes with keyword search function. ABE schemes may have sensitive information and leak the privacy of the encryptor because access policy is sent to the decryptor along with the ciphertext. ABE scheme with hidden access policy [40] and privacy preserving ABE scheme [41] can overcome the above issue. To ensure cloud service provider correctly stores ciphertext, some ABE schemes [42], [43] with efficient verifiable outsourced decryption were proposed. Recent research mainly concentrates on verifiability of outsourced decryption for the authorized users. How to guarantee the correctness of outsourced decryption for unauthorized users still remains a challenging problem. Recently Li et al. [44] presented a full verifiability for outsourced decryption in ABE, which can simultaneously verify the correctness of transformed ciphertext for the unauthorized users and authorized users.

1.3 Organizations

The remainder of the paper is organized as follow. Section 2 introduces the preliminaries and Section 3 gives the detailed construction of our scheme. The security proof and performance analysis are demonstrated in Sections 4 and 5. We conclude the paper in Section 6.

2 Preliminaries

In this section, we introduce the preliminary knowledge used throughout this paper.

2.1 Bilinear Maps

$\mathbb{G}_1$ and $\mathbb{G}_2$ are two multiplicative cyclic groups with large prime order q. g is a generator of $\mathbb{G}_1$. $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map with the following properties:

1) Computability: for $\forall u, v \in \mathbb{G}_1$, $e(u, v)$ can be computed efficiently.
2) Bilinearity: for $\forall u, v \in \mathbb{G}_1, \forall a, b \in Z_q^*$, it has $e(u^a, v^b) = e(u, v)^{ab}$.
3) Non-degeneracy: $\exists u, v \in \mathbb{G}_1$ such that $e(u, v) \neq 1_{\mathbb{G}_2}$.

2.2 Complexity Assumption

Definition 1 (Computational Diffie-Hellman (CDH) problem). *Suppose $\mathbb{G}_1$ is a multiplicative cyclic groups. g is a generator of $\mathbb{G}_1$. Given the tuple (g, g^a, g^b) with the unknown elements $a, b \in Z_q^*$, the CDH problem is to compute g^{ab}.*

Definition 2 (CDH assumption). *For any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, the advantage for $\mathcal{A}$ to solve the CDH problem in $\mathbb{G}_1$ is negligible, which can be defined as: $Adv^{CDH}_{\mathbb{G}_1 \mathcal{A}} = \Pr[\mathcal{A}(g, g^a, g^b) = g^{ab} : a, b \overset{R}{\leftarrow} Z_q^*] \leq \varepsilon$.*

Definition 3 (Discrete Logarithm (DL) problem). *Assume that $\mathbb{G}_1$ is a multiplicative cyclic group. g is a generator of $\mathbb{G}_1$. Given the values of (g,g^a) with unknown element $a \in Z_q^*$, the DL problem is to output a.*

Definition 4 (DL assumption). *For any PPT algorithm $\mathcal{A}$, the advantage for $\mathcal{A}$ to solve the DL problem in $\mathbb{G}_1$ is negligible, which can be defined as: $Adv^{DL}_{\mathbb{G}_1 \mathcal{A}} = \Pr[\mathcal{A}(g, g^a) = a : a \overset{R}{\leftarrow} Z_q^*] \leq \varepsilon$.*

Notably, ε denotes a negligible value in the above definitions.

2.3 System Model

Referring to the papers [25], [26], [27], [28], the system model of our scheme is composed of three major entities: user group, cloud service provider (CSP) and public verifier. The user group includes numbers of users, who can upload, access and update the data shared within the group, and honestly execute the protocol. Without loss of generality, the original creator of the group plays the role of group manager, who sets up the system and generates partial keys for general group users. CSP owns powerful storage and computational abilities to supply cloud users with data storage service. In our scheme, the shared data is divided into many blocks and each block is attached with an authentication tag. Thus, the CSP stores all the blocks and the corresponding tags for cloud user. The data verifier is a person who checks the integrity of the data on CSP. Due to the feature of public verification, anyone could be the verifier in our scheme.

The Fig. 1 shows the relationships and the interactions among the three entities of the system. As most previous works [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], we assume the CSP is semi-trusted. That is, the CSP can honestly execute the protocol, but may cheat the verifier about the incorrectness of the data so as to keep its reputation or get extra benefits.

2.4 Outline of RDPC Scheme

Definition 5. *The RDPC scheme is composed of eight algorithms:*

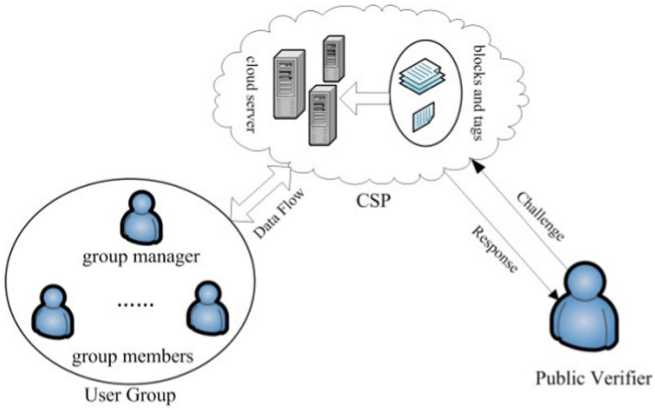


Fig. 1. System model of our scheme.

Setup. This algorithm inputs the security parameter k, outputs the master key msk and the system public parameters $params$. This algorithm is executed by the manager of the group.

PartialKeyGen: This algorithm is performed by the group manager to extract the partial key for group users. It takes the master key msk and the identity ID_i of the user u_i as the inputs, outputs the u_i's partial key D_i.

SecretValueGen. The group user executes this algorithm to generate the secret value. The algorithm randomly selects S_i as the secret value for the user u_i. Thus, the private key of group user contains two components: secret value and partial key.

PublicKeyGen. The algorithm is performed by group user to generate the public key. It inputs the u_i's secret value S_i and outputs the u_i's public key PK_i.

TagGen. The group user runs this algorithm to generate authentication tag for a block. This algorithm inputs the u_i's partial key D_i, the secret value S_i and the block m_j, and outputs the tag T_j of the m_j.

Challenge. The algorithm is performed by the verifier. It inputs the count of challenged blocks c and outputs the challenge information $chal$.

ProofGen. CSP runs this algorithm to obtain the possession proof. It takes the challenged file F, tag set T of all the blocks and the challenge information $chal$ as inputs, and outputs the integrity proof P.

Verify. The verifier runs this algorithm to verify the proof P. It inputs the proof P, the challenge information $chal$ and the public key set PK of all the group users. If P is correct, the algorithm outputs 1, otherwise it outputs 0.

2.5 Security Model

Since our new scheme introduces the idea of certificateless cryptography [33], we consider three adversaries namely $\mathcal{A}_I$, $\mathcal{A}_{II}$ and $\mathcal{A}_{III}$ in our security model. Both $\mathcal{A}_I$ and $\mathcal{A}_{II}$ try to forge the tag of the block. But the differences of them are that $\mathcal{A}_I$ can't access the master key of the system but can substitute the user's public key with any other value, and $\mathcal{A}_{II}$ is able to get the master-key but can't substitute the user's public key. The $\mathcal{A}_{III}$ aims to forge the data integrity proof to cheat the verifier. Referring to [20], [26], we define the security of our scheme by three games which involves a challenger $\mathcal{C}$ and the adversaries $\mathcal{A}_I$, $\mathcal{A}_{II}$ and $\mathcal{A}_{III}$ respectively. Three security games are described as follows.

Game I. This game is played by $\mathcal{C}$ and $\mathcal{A}_I$.

Setup. $\mathcal{C}$ performs the $Setup$ algorithm to obtain the master key and the public parameters. $\mathcal{C}$ keeps the master key privately and sends the public parameters to $\mathcal{A}_I$.

Queries. $\mathcal{A}_I$ can make different queries to $\mathcal{C}$ for polynomial times. $\mathcal{C}$ responses the queries to $\mathcal{A}_I$ as follows.

1) Hash Query. $\mathcal{A}_I$ adaptively makes hash function queries to $\mathcal{C}$. $\mathcal{C}$ responses the hash values to $\mathcal{A}_I$.
2) Partial Key Query. $\mathcal{A}_I$ adaptively chooses different ID and submits it to $\mathcal{C}$ for querying the partial key of the ID. $\mathcal{C}$ executes the *PublicKeyGen* algorithm to obtain the partial key for the ID and sends it to $\mathcal{A}_I$.
3) Secret Value Query. $\mathcal{A}_I$ adaptively chooses different ID and submits it to $\mathcal{C}$ for querying the secret key of the ID. $\mathcal{C}$ runs the *SecretValueGen* to generate the secret value for this ID and sends it to $\mathcal{A}_I$.
4) Public Key Query. $\mathcal{A}_I$ adaptively selects different ID and submits it to $\mathcal{C}$ for querying the public key of the ID. $\mathcal{C}$ performs the algorithm of *PublicKeyGen* to compute the public key for the ID and sends it to $\mathcal{A}_I$.
5) Public Key Replacement. $\mathcal{A}_I$ can repeatedly choose a value to substitute the public key of any ID.
6) Tag Query. $\mathcal{A}_I$ adaptively chooses the tuple (ID, m) and submits it to $\mathcal{C}$ for querying the tag of the block m generated by the ID. By the algorithm *TagGen*, $\mathcal{C}$ generates the tag for m and sends the tag to $\mathcal{A}_I$.

Forge. Finally, $\mathcal{A}_I$ outputs a forged tag T' for the m' with the identity ID' and the public key $PK_{ID'}$.

If the conditions listed below are satisfied, $\mathcal{A}_I$ wins the game:

1) The forged tag T' is valid for the block m' with the identity ID' and the public key $PK_{ID'}$.
2) $\mathcal{A}_I$ doesn't query the private key of the user with ID'.
3) $\mathcal{A}_I$ does not query the partial key of the ID' and replace the public key of the ID' simultaneously.
4) $\mathcal{A}_I$ does not make the tag query for the m' with the identity ID' and the public key $PK_{ID'}$.

Game II. This game is played by $\mathcal{C}$ and $\mathcal{A}_{II}$.

Setup. $\mathcal{C}$ performs the $Setup$ algorithm to obtain the master key and the public parameters. $\mathcal{C}$ sends the public parameters and the master key to $\mathcal{A}_{II}$.

Queries. $\mathcal{A}_{II}$ can make different queries to $\mathcal{C}$ for polynomial times. $\mathcal{C}$ responses the queries to $\mathcal{A}_{II}$ as follows.

1) Hash Query. $\mathcal{A}_{II}$ adaptively asks hash function queries to $\mathcal{C}$. $\mathcal{C}$ responses the hash values to $\mathcal{A}_{II}$.
2) Secret Value Query. $\mathcal{A}_{II}$ adaptively chooses different ID and submits it to $\mathcal{C}$ for querying the secret value of the ID. $\mathcal{C}$ executes the *SecretValueGen* to obtain the secret value for the ID and sends it to $\mathcal{A}_{II}$.
3) Public Key Query. $\mathcal{A}_{II}$ adaptively chooses different ID and submits it to $\mathcal{C}$ for querying the public key of the ID. $\mathcal{C}$ executes the algorithm of *PublicKeyGen* to compute the public key for the ID and returns it to $\mathcal{A}_{II}$.
4) Tag Query. $\mathcal{A}_{II}$ adaptively chooses the tuple (ID, m) and submits it to $\mathcal{C}$ for querying the tag for the block m generated by the ID. By the algorithm *TagGen*, $\mathcal{C}$ generates the tag for m and returns it to $\mathcal{A}_{II}$.

Forge. Finally, $\mathcal{A}_{II}$ outputs a forged tag T' for the m' with the identity ID'.

If the conditions listed below are satisfied, $\mathcal{A}_{II}$ wins the game:

1) The forged tag T' is valid for the block m' with the identity ID'.
2) 2) $\mathcal{A}_{II}$ doesn't query the secret value for the ID'.
3) 3) $\mathcal{A}_{II}$ doesn't query the tag for the m' with the identity ID'.

Definition 6. *If for any PPT adversary $\mathcal{A}$ ($\mathcal{A}_I$ or $\mathcal{A}_{II}$), the probability that $\mathcal{A}$ wins the Game I and Game II is negligible, the single tag of block is existentially unforgeable.*

Game III. The challenger $\mathcal{C}$ interacts with the adversary $\mathcal{A}_{III}$ in this game. Here, $\mathcal{A}_{III}$ is regarded as the untrusted CSP. If the data is corrupted, $\mathcal{A}_{III}$ tries to cheat the verifier that the data is kept intact. From the Definition 6, we know that any adversary can't forge the tag of single block without the right private key. Thus, in this game we just focus on the issue that whether $\mathcal{A}_{III}$ can forge the integrity proof on incorrect data to pass the verification. Inspired by [26], the procedure of game III is defined below.

Setup. $\mathcal{C}$ generates the public parameters, the master key and private keys for all users. $\mathcal{C}$ keeps all the private keys and the master key privately, but sends the public parameters to $\mathcal{A}_{III}$.

Tag-Query. $\mathcal{A}_{III}$ adaptively chooses the tuple (ID, m) and sends it to $\mathcal{C}$ for querying the tag of m generated by the ID. $\mathcal{C}$ generates the tag of m and returns it to $\mathcal{A}_{III}$ by the algorithm $TagGen$.

Challenge.$\mathcal{C}$ generates a random challenge $chal$, sends $chal$ to $\mathcal{A}_{III}$ and requests $\mathcal{A}_{III}$ to give the corresponding data possession proof P for $chal$.

Forge. For the challenge $chal$, $\mathcal{A}_{III}$ generates a proof P and gives it to $\mathcal{C}$. If P can pass the integrity verification and the block information in P is wrong, $\mathcal{A}_{III}$ wins the game.

Definition 7. *If any PPT adversary $\mathcal{A}_{III}$ can win the Game III about a set of blocks with only negligible probability, the probability to forge the integrity proof without correct data is negligible.*

3 Our New Scheme

3.1 Construction of our Scheme

We suppose there are z users in one group. Let ID_i represent the unique identity of the user u_i for $1 \le i \le z$. Without loss of the generality, we set u_1 to be the group manager, who will set up the system and generate the partial secret keys for other users. We suppose the file F is split into n blocks, represented as $F = (m_1, m_2, \ldots, m_n)$, and each block is an element in Z_q. The detailed construction for the scheme is demonstrated as follows:

Setup. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be multiplicative cyclic groups with primer order q. $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear map and g is a generator of $\mathbb{G}_1$. u_1 selects two secure hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$ and $H_2 : \{0,1\}^* \to \mathbb{G}_1^*$, a pseudo-random permutation (PRP) π: $Z_q^* \times \{1, \ldots, n\} \to \{1, \ldots, n\}$ and a pseudo-random function (PRF) ϕ: $Z_q^* \times Z_q^* \to Z_q^*$. u_1 randomly chooses $s \in Z_q^*$ as the master secret key and calculates $P_0 = g^s$. u_1 keeps the master secret key privately and publishes the public system parameters $params = (q,\ g,\ \mathbb{G}_1,\ \mathbb{G}_2,\ e,\ P_0,\ H_1,\ H_2,\ \phi,\ \pi)$.

PartialKeyGen. When receiving the identity ID_i of the group user u_i, the group manager u_1 computes the $D_i = H_1(ID_i)^s$ and returns the D_i to the u_i as the partial private key. It is noted that the ID_i is unique identity in the group.

SecretValueGen. u_i randomly chooses $x_i \in Z_q^*$ and sets $S_i = x_i$ as the secret value and keeps it private.

PublicKeyGen. u_i uses the secret value S_i to compute the public key $PK_i = g^{S_i} = g^{x_i}$.

TagGen. Each user in the group can access the blocks and generate tags for them with his private key. Suppose the user u_i $(1 \le i \le z)$ wants to generate the tag for the block $m_j (1 \le j \le n)$, the equation for computing tag is $T_j = {D_i}^{m_j} \cdot H_2(\omega_j)^{S_i}$, where $\omega_j = F_{id}||n||j$, and F_{id} denotes the unique file identity. The group manager maintains a public log file which stores the information of the ω_j $(1 \le j \le n)$ and the identity of tag generators. After adding or modifying data blocks, the user should add or update the corresponding ω_j $(1 \le j \le n)$ and the identity of tag generators in the log file. The user uploads the blocks and the tags to CSP. CSP can check the validation of the tag by

$$e(T_j, g) = e(H_2(\omega_j), PK_i) \cdot e(H_1(ID_i)^{m_j}, P_0). \tag{1}$$

Challenge. The verifier randomly picks the count of challenged blocks $c \in [1, n]$ and the two random values (k_1, k_2), where $k_1, k_2 \in Z_q^*$ are seeds for PRP and PRF respectively. The verifier sends $chal = (c, k_1, k_2)$ to the CSP as the challenge information.

ProofGen. When the CSP receives the challenge information $chal = (c, k_1, k_2)$, the CSP computes the set $C = \{(v_i, a_i)\}$, in which $v_i = \pi(k_1, i)$, $a_i = \phi(k_2, i)$ for $1 \le i \le c$. According to tag generator of each challenged block, the CSP splits the set C into d subsets $C = \{C_1, \ldots, C_d\}$ $(d \le z)$ in which the subset C_j is the collection of the tags generated by the user $u_{l_j} (1 \le j \le d, 1 \le l_j \le z)$. Let c_j denote the count of the element in C_j. We can get $c = \sum_{j=1}^{d} c_j$, $C = C_1 \cup C_2 \cdots \cup C_d$ and $C_j \cap C_{j'} = \emptyset$ for $j \ne j'$. For each subset C_j, the CSP computes $(\overline{T_j}, \overline{F_j})$ by $\overline{T_j} = \prod_{v_i \in C_j} {T_{v_i}}^{a_i}$ and $\overline{F_j} = \sum_{v_i \in C_j} a_i m_{v_i}$. The CSP returns $P = (\overline{T}, \overline{F})$ as the final proof, where $\overline{T} = \{\overline{T_1}, \ldots, \overline{T_d}\}$, $\overline{F} = \{\overline{F_1}, \ldots, \overline{F_d}\}$.

Verify. When receiving the proof P, the verifier calculates $v_i = \pi(k_1, i)$,$a_i = \phi(k_2, i)$ and divides the challenge set according to the tag generators as the CSP does. By searching in the log file, the verifier gets all ω_{v_i} of the challenged blocks. Finally the verifier checks whether the Equation (2) holds:

$$\begin{aligned} e\left(\prod_{j=1}^{d} \overline{T_j}, g\right) = \prod_{j=1}^{d} \left(e\left(\prod_{v_i \in C_j} H_2(\omega_{v_i})^{a_i}, PK_{l_j} \right) \right) \\ \cdot e\left(\prod_{j=1}^{d} H_1(ID_{l_j})^{\overline{F_j}}, P_0 \right) \end{aligned} \tag{2}$$

If the Equation (2) holds, verifier outputs 1, otherwise outputs 0. If the CSP and the verifier honestly run the protocol, we can check the correctness of the protocol by the following equality:

$$e\left(\prod_{j=1}^{d}\overline{T_j},g\right)=\prod_{j=1}^{d}e(\overline{T_j},g)=\prod_{j=1}^{d}e\left(\prod_{v_i\in C_j}{T_{v_i}}^{a_i},g\right)$$
$$=\prod_{j=1}^{d}e\left(\prod_{v_i\in C_j}\left(H_1(ID_{l_j})^{sa_im_{v_i}}\cdot\left(H_2(\omega_{v_i})\right)^{x_{l_j}a_i}\right),g\right)$$
$$=\prod_{j=1}^{d}\left(e\left(\prod_{v_i\in C_j}H_1(ID_{l_j})^{a_im_{v_i}},g^s\right)\cdot e\left(\prod_{v_i\in C_j}H_2(\omega_{v_i})^{a_i},g^{x_{l_j}}\right)\right)$$
$$=\prod_{j=1}^{d}\left(e\left(H_1(ID_{l_j})^{\overline{F_j}},P_0\right)\cdot e\left(\prod_{v_i\in C_i}H_2(\omega_{v_i})^{a_i},PK_{l_j}\right)\right)$$
$$=\prod_{j=1}^{d}e\left(\prod_{v_i\in C_j}H_2(\omega_{v_i})^{a_i},PK_{l_j}\right)\cdot e\left(\prod_{j=1}^{d}H_1(ID_{l_j})^{\overline{F_j}},P_0\right)$$

3.2 Supporting User Revocation

If any user u_i $(2\le i\le z)$ leaves the group, the secret keys and public key of u_i should be claimed to be invalid. Thus, the secret key of u_i should be removed from the tags generated by the u_i. Otherwise, the validation of these tags can't be verified and the integrity of the file can't be checked either. The traditional method for re-generating tags of the revoked users is to download the blocks from CSP, re-generate the tags and then upload the new tags again. It inevitably increases the communication and the computation cost for the users. Therefore, to relieve these potential overhead, our scheme designs to outsource the tag update work to the CSP and reduces the communication cost. We use the algorithm $ReTagGen$ to update the tags generated by revoked users. The construction of $ReTagGen$ is shown as follows, in which the u_i is the revoked user, u_j $(1\le j\le z, j\ne i)$ is another valid user in the group. Notably, u_1 is the group creator and manager, we consider u_1 should not be revoked.

$ReTagGen$: This algorithm contains three interactions among u_i, u_j and CSP. We suppose there is no collusion among u_i, u_j and CSP, and the secure channel is used during the interactions. Besides, it is required that u_i, u_j and CSP are online simultaneously among the revocation procedure.

1) CSP randomly chooses a value $\rho\in Z_q^*$ and sends ρ to u_j by secure channel.
2) u_j computes and sends $(W_1=(D_j)^{\frac{1}{S_j}}, W_2=\rho\cdot S_j)$ to u_i.
3) u_i calculates and sends $(R_1=\frac{W_1{}^{S_i}}{D_i}, R_2=\frac{W_2}{S_i})$ to CSP.
4) When receiving the (R_1,R_2), CSP calculates $R_3=\frac{R_2}{\rho}=\frac{S_j}{S_i}$. Using the Equation (1) to retrieve and check all the tags-block pairs $[T_{i'},m_{i'}]$ $(1\le i'\le n)$ generated by u_i. Then the CSP transforms the tags $T_{i'}$ for the block $m_{i'}$ as

$$T_{i'}{}'=(R_1{}^{m_{i'}}\cdot T_{i'})^{R_3}=\left(\left(\frac{D_j^{\frac{S_i}{S_j}}}{D_i}\right)^{m_{i'}}\cdot {D_i}^{m_{i'}}\cdot H_2(\omega_{i'})^{S_i}\right)^{\frac{S_j}{S_i}}$$
$$=\left(D_j^{\frac{S_im_{i'}}{S_j}}\cdot H_2(\omega_{i'})^{S_i}\right)^{\frac{S_j}{S_i}}=D_j^{m_{i'}}\cdot H_2(\omega_{i'})^{S_j}$$

where $T_{i'}{}'$ is the valid tag of $m_{i'}$ for the generator u_j.

4 Security Proof

Our scheme is proved secure via the following three theorems.

4.1 Security Proof

Theorem 1. *If a PPT adversary $\mathcal{A}_I$ wins the Game I defined in Section 2.5 at the advantage ε within time t, after asking H_1-Query, PartialKey-Query, SecretValue-Query, PublicKey-Query, PublicKey-Replace, H_2-Query and Tag-Query for at most q_{H_1}, q_{k1}, q_{k2}, q_{k3}, q_{kr}, q_{H_2}, q_T times respectively, then a simulator $\mathcal{B}$ can break the CDH problem with the probability $\varepsilon'\ge\varepsilon/((q_{k1}+q_T)\cdot 2e)$ within the time $t'\le t+O(q_{H_1}+q_{k1}+q_{k2}+q_{k3}+q_{kr}+q_{H_2}+q_T)$.*

Proof. Given a CDH instance $(g,\mathbb{G}_1,g^a,g^b)$. If the adversary $\mathcal{A}_I$ wins the Game I at non-negligible probability, the simulator $\mathcal{B}$ can calculate the value of g^{ab} at non-negligible probability by the capability of $\mathcal{A}_I$. $\mathcal{B}$ simulates each interaction step with $\mathcal{A}_I$ as below.

Setup. $\mathcal{B}$ produces the public parameters and sets $P_0=g^a$ in which the master key is the unknown value a implicitly.

H_1-Query. $\mathcal{A}_I$ adaptively executes the H_1-Query for any identity ID^*. $\mathcal{B}$ maintains a list $L_1=\{(ID,h_1,Q,T)\}$ for the H_1-Query. If the ID^* exists in L_1, $\mathcal{B}$ retrieves the tuple $(ID^*,{h_1}^*,Q^*,T^*)$ and returns Q^* to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ randomly picks a value ${h_1}^*\in Z_q^*$ and flips a coin $T\in\{0,1\}$. Suppose the probability of $T^*=0$ is γ, so the probability of $T^*=1$ is $1-\gamma$. If $T=0$, $\mathcal{B}$ computes $Q^*=g^{{h_1}^*}$. If $T=1$, $\mathcal{B}$ sets $Q^*=(g^b)^{{h_1}^*}$. $\mathcal{B}$ responses Q^* to $\mathcal{A}_I$ and inserts $(ID^*,{h_1}^*,Q^*,T^*)$ to L_1.

PartialKey-Query. $\mathcal{A}_I$ adaptively executes the PartialKey-Query for any identity ID^*. $\mathcal{B}$ first checks whether $(ID^*,{h_1}^*,Q^*,T^*)$ exists in L_1. If not, $\mathcal{B}$ makes the H_1-Query for ID^* itself. Notably, if $T^*=1$, $\mathcal{B}$ aborts. $\mathcal{B}$ also maintains a list $L_2=\{(ID,D_{ID},PK_{ID},S_{ID})\}$ for the PartialKey-Query.

1) If the ID^* exists in L_2, $\mathcal{B}$ checks whether the corresponding value of $D_{ID^*}=\perp$. If $D_{ID^*}=\perp$, $\mathcal{B}$ searches the tuple $(ID^*,{h_1}^*,Q^*,T^*)$ from L_1, computes the $D_{ID^*}=(Q^*)^a=g^{a{h_1}^*}$ for $T^*=0$ and updates D_{ID^*} in the tuple. If $T^*=1$, $\mathcal{B}$ aborts. For $D_{ID^*}\ne\perp$, $\mathcal{B}$ gets the D_{ID^*} directly. Then $\mathcal{B}$ returns D_{ID^*} to $\mathcal{A}_I$.
2) If the ID^* doesn't exist in L_2, $\mathcal{B}$ retrieves the tuple $(ID^*,{h_1}^*,Q^*,T^*)$ from L_1 and computes $D_{ID^*}=(Q^*)^a=g^{a{h_1}^*}$ for $T^*=0$. If $T^*=1$, $\mathcal{B}$ aborts. Finally $\mathcal{B}$ returns D_{ID^*} to $\mathcal{A}_I$, and inserts the new tuple $(ID^*,D_{ID^*},\perp,\perp)$ into L_2.

SecretValue-Query. $\mathcal{A}_I$ adaptively executes the SecretValue-Query for any identity ID^*. $\mathcal{B}$ first checks whether $(ID^*,{h_1}^*,Q^*,T^*)$ has existed in L_1. If not, $\mathcal{B}$ makes the H_1-Query for ID^* itself. $\mathcal{B}$ checks whether ID^* exists in L_2.

1) If ID^* exists in L_2, $\mathcal{B}$ checks whether the corresponding value $S_{ID^*}=\perp$. If $S_{ID^*}=\perp$, $\mathcal{B}$ randomly selects a value $x\in Z_q^*$ and sets $S_{ID^*}=x$, $PK_{ID^*}=g^x$. Then $\mathcal{B}$ updates this tuple with S_{ID^*} and PK_{ID^*}. If $S_{ID^*}\ne\perp$, $\mathcal{B}$ retrieves it from the tuple and returns it to $\mathcal{A}_I$.

2) If ID^* doesn't exist in L_2, $\mathcal{B}$ randomly chooses a value $x \in Z_q^*$ and sets $S_{ID^*} = x$, $PK_{ID^*} = g^x$. Finally, $\mathcal{B}$ adds a new tuple $(ID^*, \perp, g^x, x)$ into L_2 and returns S_{ID^*} to $\mathcal{A}_I$.

PublicKey-Query. $\mathcal{A}_I$ adaptively executes the Public-Key-Query for any identity ID^*.

1) If the tuple $(ID^*, D_{ID^*}, PK_{ID^*}, S_{ID^*})$ exists in L_2, $\mathcal{B}$ checks whether $PK_{ID^*} = \perp$ or not. If $PK_{ID^*} = \perp$, $\mathcal{B}$ randomly picks a value $x \in Z_q^*$ and sets $S_{ID^*} = x$, $PK_{ID^*} = g^x$. $\mathcal{B}$ returns PK_{ID^*} and updates the S_{ID^*}, PK_{ID^*} of the tuple. If $PK_{ID^*} \neq \perp$, $\mathcal{B}$ returns it to $\mathcal{A}_I$ directly.
2) If the L_2 does not include the tuple $(ID^*, D_{ID^*}, PK_{ID^*}, S_{ID^*})$, $\mathcal{B}$ randomly chooses $x \in Z_q^*$ and sets $S_{ID^*} = x$, $PK_{ID^*} = g^x$. $\mathcal{B}$ adds a new tuple $(ID^*, \perp, PK_{ID^*}, S_{ID^*})$ into L_2 and sends PK_{ID^*} to $\mathcal{A}_I$.

PublicKey-Replace. $\mathcal{A}_I$ adaptively executes the Public-Key-Replace with the (ID^*, PK'_{ID^*}).

1) If the tuple $(ID^*, D_{ID^*}, PK_{ID^*}, S_{ID^*})$ exists in L_2, $\mathcal{B}$ updates the tuple to $(ID^*, D_{ID^*}, PK'_{ID^*}, \perp)$.
2) If L_2 does not contain the tuple $(ID^*, D_{ID^*}, PK_{ID^*}, S_{ID^*})$, $\mathcal{B}$ adds a new tuple $(ID^*, \perp, PK'_{ID^*}, \perp)$ to L_2.

H_2-Qurey. $\mathcal{A}_I$ adaptively executes the H_2-Query for $\omega^* \in Z_q^*$. $\mathcal{B}$ also maintains a list $L_3 = \{(\omega, h_2)\}$ for the H_2-Query. If the L_3 contains the ω^*, $\mathcal{B}$ retrieves the h_2^* and returns $g^{h_2^*}$ to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ selects a random value $h_2^* \in Z_q^*$ and sends $g^{h_2^*}$ to $\mathcal{A}_I$. Then $\mathcal{B}$ inserts the (ω^*,h_2^*) into L_3.

Tag-Query. $\mathcal{A}_I$ adaptively executes the Tag-Query with (ω^*, m^*, ID^*). $\mathcal{B}$ first checks whether $T^* = 0$ in the list L_1 for ID^*. If $T^* = 1$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ gets the $H_2(\omega^*)$ from L_3, the D_{ID^*} and S_{ID^*} from L_2. Then $\mathcal{B}$ computes the tag for (ω^*, m^*, ID^*) by the algorithm $TagGen$ and returns it to $\mathcal{A}_I$.

Forge. Finally, $\mathcal{A}_I$ outputs a tuple $O = (T', \omega', m', ID', PK_{ID'})$ in which T' is the forged tag of the block m' on the identity ID' with the public key $PK_{ID'}$.

Analysis. If $\mathcal{A}_I$ wins the game I, $\mathcal{B}$ can get $e(T', g) = e(H_1(ID')^{m'}, P_0) \cdot e(H_2(\omega'), PK_{ID'})$ according to the verification Equation (1). $\mathcal{B}$ retrieves the tuple (ID', h'_1, Q', T') from L_1. If $T^* = 0$, $\mathcal{B}$ aborts and outputs 'fail'. Otherwise, $\mathcal{B}$ retrieves the $H_1(ID') = g^{bh'_1}$ from L_1 and $H_2(\omega') = g^{h'_2}$ from L_3. According to the verification equation mentioned above, $\mathcal{B}$ can obtain $e(T', g) = e(g^{bh'_1 m'}, g^a) \cdot e(g^{h'_2}, PK_{ID'})$. Thus, we can derive $g^{ab} = (\frac{T'}{(PK_{ID'})^{h'_2}})^{1/h'_1 m'}$.

Now, we evaluate the probability of $\mathcal{B}$ outputting the right result. Obviously, if $\mathcal{B}$ does not abort in the process above, $\mathcal{B}$ performs a perfect interaction simulation with $\mathcal{A}_I$. Further, we can see that the H_1-Query, SecretValue-Query, PublicKey-Query, PublicKey- Replace and H_2-Query can be executed completely without additional conditions. The abortion of $\mathcal{B}$ only happens in the procedures of PartialKey-Query and Tag-Query. Therefore, the probability that $\mathcal{B}$ perfectly simulates the interactions with $\mathcal{A}_I$ without abortion is higher than $(1-\gamma)^{q_{k1}+q_T}$. Thus, $\mathcal{B}$ outputs the right value g^{ab} with the probability $\varepsilon' \geq \varepsilon \cdot \gamma \cdot (1-\gamma)^{q_{k1}+q_T} \geq \varepsilon/((q_{k1}+q_T) \cdot 2e)$. The corresponding time cost is $t' \leq t + O(q_{H_1} + q_{k1} + q_{k2} + q_{k3} + q_{kr} + q_{H_2} + q_T)$. □

Theorem 2. *If a PPT adversary $\mathcal{A}_{II}$ wins the Game II defined in Section 2.5 with the advantage ε within time t, after making H_1-Query, SecretValue-Query, PublicKey-Query, H_2-Query and Tag-Query for at most q_{H_1}, q_{k1}, q_{k2}, q_{H_2}, q_T times respectively, then a simulator $\mathcal{B}$ can break the CDH problem with the probability $\varepsilon' \geq \varepsilon/((q_{k1}+q_T) \cdot 2e)$ within the time $t' \leq t + O(q_{H_1} + q_{k1} + q_{k2} + q_{H_2} + q_T)$.*

Proof. Given a CDH instance $(g, \mathbb{G}_1, g^a, g^b)$. If the adversary $\mathcal{A}_{II}$ wins the Game II at non-negligible probability, the simulator $\mathcal{B}$ can calculate the value g^{ab} at non-negligible probability by the capability of $\mathcal{A}_{II}$. $\mathcal{B}$ simulates each interaction step with $\mathcal{A}_{II}$ as below.

Setup. $\mathcal{B}$ randomly picks a value $s \in Z_q^*$ as the master key and generates the public parameters. $\mathcal{B}$ returns the master key and the public parameters to $\mathcal{A}_{II}$.

H_1-Query. $\mathcal{A}_{II}$ adaptively executes the H_1-Query for any identity ID^*. $\mathcal{B}$ maintains a list $L_1 = \{(ID, h_1)\}$ for the H_1-Query. If the ID^* exists in L_1, $\mathcal{B}$ retrieves the tuple $(ID^*, {h_1}^*)$ and returns $g^{h_1^*}$ to $\mathcal{A}_{II}$. Otherwise, $\mathcal{B}$ selects a random value ${h_1}^* \in Z_q^*$ and calculates $Q^* = g^{{h_1}^*}$. $\mathcal{B}$ responses Q^* to $\mathcal{A}_{II}$ and inserts $(ID^*, {h_1}^*)$ to L_1.

SecretValue-Query. Because $\mathcal{A}_{II}$ knows the master key, it is no need for $\mathcal{A}_{II}$ to query the partial key. $\mathcal{A}_{II}$ adaptively executes the SecretValue-Query for any identity ID^*. $\mathcal{B}$ maintains a list $L_2 = \{(ID, PK_{ID}, S_{ID}, T)\}$ for the Secret-Value-Query. $\mathcal{B}$ first checks whether ID^* exists in L_2.

1) If the ID^* doesn't exist in L_2, $\mathcal{B}$ randomly chooses a value $x^* \in Z_q^*$ and flips a coin $T^* \in \{0, 1\}$. Suppose the probability of $T^* = 0$ is γ, so the probability of $T^* = 1$ is $1 - \gamma$. If $T^* = 1$, $\mathcal{B}$ calculates $PK_{ID^*} = (g^a)^{x^*}$ and adds the tuple $(ID^*, PK_{ID^*}, x^*, T^*)$ to L_2. Then $\mathcal{B}$ outputs 'fail' and aborts. If $T^* = 0$, $\mathcal{B}$ calculates $PK_{ID^*} = g^{x^*}$ and adds the tuple $(ID^*, PK_{ID^*}, x^*, T^*)$ to L_2. Then $\mathcal{B}$ returns x^* to $\mathcal{A}_{II}$.
2) If the ID^* exists in L_2, $\mathcal{B}$ checks the corresponding value T^*. If $T^* = 1$, $\mathcal{B}$ outputs 'fail' and abort. Otherwise, $\mathcal{B}$ retrieves the S_{ID^*} and returns it to $\mathcal{A}_{II}$. (It is sure that S_{ID^*} exists in L_2 when $T^* = 0$).

PublicKey-Query. $\mathcal{A}_{II}$ adaptively executes the Public-Key- Query for any identity ID^*.

1) If the tuple $(ID^*, PK_{ID^*}, S_{ID^*}, T^*)$ exists in L_2, $\mathcal{B}$ returns PK_{ID^*} to $\mathcal{A}_{II}$ directly.
2) If the L_2 does not contain the tuple $(ID^*, PK_{ID^*}, S_{ID^*}, T^*)$, $\mathcal{B}$ randomly selects a value $x^* \in Z_q^*$ and flips a coin $T^* \in \{0, 1\}$. Suppose the probability of $T^* = 0$ is γ, so the probability of $T^* = 1$ is $1 - \gamma$. If $T^* = 1$, $\mathcal{B}$ calculates $PK_{ID^*} = (g^a)^{x^*}$. If $T^* = 0$, $\mathcal{B}$ sets $S_{ID^*} = x^*$ and calculates $PK_{ID^*} = g^{x^*}$. $\mathcal{B}$ adds a new tuple $(ID^*, PK_{ID^*}, x^*, T^*)$ into L_2 and returns PK_{ID^*} to $\mathcal{A}_{II}$.

H_2-Qurey. $\mathcal{A}_{II}$ adaptively executes the H_2-Query for $\omega^* \in Z_q^*$. $\mathcal{B}$ also maintains a list $L_3 = \{(\omega, h_2)\}$ for the H_2-Query. If the L_3 contains the ω^*, $\mathcal{B}$ retrieves the h_2^* and returns $(g^b)^{h_2^*}$ to $\mathcal{A}_{II}$. Otherwise, $\mathcal{B}$ selects a random value $h_2^* \in Z_q^*$ and returns $(g^b)^{h_2^*}$ to $\mathcal{A}_{II}$. Then $\mathcal{B}$ inserts the (ω^*,h_2^*) into L_3.

Tag-Query. $\mathcal{A}_{II}$ adaptively executes the Tag-Query with the (ω^*, m^*, ID^*). $\mathcal{B}$ first checks whether $T^* = 0$ in the list

L_2 for ID^*. If $T^* = 1$, $\mathcal{B}$ aborts and outputs 'fail'. Otherwise, $\mathcal{B}$ calculates the D_{ID^*} and gets the $H_2(\omega^*)$ from L_3, the S_{ID^*} from L_2. Then $\mathcal{B}$ computes the tag for (ω^*, m^*, ID^*) by the algorithm $TagGen$ and returns it to $\mathcal{A}_{II}$.

Forge. Finally, $\mathcal{A}_{II}$ outputs a tuple $O = (T', \omega', m', ID')$ in which T' is the forged tag on the block m' for the identity ID'.

Analysis. If $\mathcal{A}_{II}$ wins the game II, $\mathcal{B}$ can get $e(T', g) = e(H_1(ID')^{m'}, P_0) \cdot e(H_2(\omega'), PK_{ID'})$ according to the verification Equation (1). Then $\mathcal{B}$ retrieves the $(ID', PK_{ID'}, x', T')$ from L_2. If $T^* = 0$, $\mathcal{B}$ aborts and outputs 'fail'. Otherwise, $\mathcal{B}$ gets the $H_1(ID') = g^{h'_1}$ from L_1, $PK_{ID'} = g^{ax'}$ from L_2 and $H_2(\omega') = g^{bh'_2}$ from L_3. According to the above equation, $\mathcal{B}$ obtains $e(T', g) = e(g^{h'_1 m'}, g^s) \cdot e(g^{bh'_2}, g^{ax'})$. Thus, we can derive $g^{ab} = (T')^{1/(x'h'_2h'_1sm')}$. Now, we evaluate the probability for $\mathcal{B}$ outputting the right result. Obviously, if $\mathcal{B}$ does not abort in the process above, $\mathcal{B}$ performs a perfect interaction simulation with $\mathcal{A}_{II}$. Further, we can see that the H_1-Query, PublicKey-Query and H_2-Query are executed completely without additional conditions. The abortion of $\mathcal{B}$ only happens in the procedures of SecretValue-Query and Tag-Query. Therefore, the probability that $\mathcal{B}$ perfectly simulates the interactions with $\mathcal{A}_{II}$ without abortion is higher than $(1-\gamma)^{q_{k1}+q_T}$. Thus, $\mathcal{B}$ outputs the right value g^{ab} with the probability $\varepsilon' \geq \varepsilon \cdot \gamma \cdot (1-\gamma)^{q_{k1}+q_T} \geq \varepsilon/((q_{k1}+q_T)\cdot 2e)$. The corresponding time cost is $t' \leq t + O(q_{H_1} + q_{k1} + q_{k2} + q_{H_2} + q_T)$. □

Theorem 3. *If the DL assumption holds, the adversary $\mathcal{A}_{III}$ wins the Game III only at negligible probability.*

Proof. Let the challenge information be $chal = (c, k_1, k_2)$. If $\mathcal{A}_{III}$ outputs the integrity proof $P' = (\overline{T'}, \overline{F'})$ and wins the Game III at non-negligible probability, we can get the verification equation:

$$e\left(\prod_{i=1}^{d'} \overline{T_i'}, g\right) = \prod_{i=1}^{d'} \left(e\left(\prod_{l\in C_i} H_2(\omega_l)^{a_l}, PK_i\right)\right) \cdot e\left(\prod_{i=1}^{d'} H_1(ID_i)^{\overline{F_i'}}, P_0\right),$$

where d' denotes the count of the group subset in the challenge. Assume the real proof for $chal = (c, k_1, k_2)$ is $P = (\overline{T}, \overline{F})$, we also gets verification equation: $e(\prod_{i=1}^{d'} \overline{T_i}, g) = \prod_{i=1}^{d'} (e(\prod_{l\in C_i} H_2(\omega_l)^{a_l}, PK_i)) \cdot e(\prod_{i=1}^{d'} H_1(ID_i)^{\overline{F_i}}, P_0)$. Because $\mathcal{A}_{III}$ wins the game III, there exists $\overline{T} = \overline{T'}$ but $\overline{F} \neq \overline{F'}$. According to the two equations above, it has $\prod_{i=1}^{d'} H_1(ID_i)^{\overline{F_i'}} = \prod_{i=1}^{d'} H_1(ID_i)^{\overline{F_i}}$. Define $\Delta\overline{F_i} = \overline{F_i'} - \overline{F_i}$ for each $1 \leq i \leq d'$, we know that $\prod_{i=1}^{d'} (H_1(ID_i))^{\overline{\Delta F_i}} = 1$. Based on this conclusion, the DL problem can be solved as follows. Give two element $h, u \in \mathbb{G}_1$ that $u = h^x$, we will compute the value $x \in Z_q^*$. Let $H_1(ID_i) = \chi_i = h^{\alpha_i} u^{\beta_i}$, in which α_i and β_i are randomly selected from Z_q^*. We can get the following equality $\prod_{i=1}^{d'} \chi_i^{\overline{\Delta F_i}} = \prod_{i=1}^{d'} (h^{\alpha_i} u^{\beta_i})^{\overline{\Delta F_i}} = h^{\sum_{i=1}^{z} \alpha_i \overline{\Delta F_i}} u^{\sum_{i=1}^{z} \beta_i \overline{\Delta F_i}} = 1$, $u = h^{-\frac{\sum_{i=1}^{z} \alpha_i \overline{\Delta F_i}}{\sum_{i=1}^{z} \beta_i \overline{\Delta F_i}}}$. Then we can derive $x = -\frac{\sum_{i=1}^{z} \alpha_i \overline{\Delta F_i}}{\sum_{i=1}^{z} \beta_i \overline{\Delta F_i}}$. Since $\overline{F} \neq \overline{F'}$, there at least one $\overline{\Delta F_i}$ $(1 \leq i \leq d')$ is not zero. β_i $(1 \leq i \leq d')$ is a random value in Z_q^*, so the probability of $\sum_{i=1}^{z} \beta_i \overline{\Delta F_i} = 0$ is only $1/q$. Therefore, we can output the right value of x with non-negligible probability $1 - 1/q$. □

4.2 Error Detection Probability

Suppose p of n blocks on CSP have been tampered. CSP randomly selects c_1 of p $(c_1 \leq p)$ blocks to generate the proof. In order to detect the data corrupt, $c_1 \geq 1$ must be satisfied. Therefore, the probability of error detection is equal to the probability of $c_1 \geq 1$. Let P_a denote error detection probability, it has $P_a = P\{c_1 \geq 1\} = 1 - P\{c_1 < 1\} = 1 - P\{c_1 = 0\} = 1 - \frac{n-p}{n} \cdot \frac{n-p-1}{n-1} \cdot \cdots \cdot \frac{n-p-c+1}{n-c+1}$. It indicates that: $1 - (1-\frac{p}{n})^c \leq P_a \leq 1 - (1 - \frac{p}{n-c+1})^c$. From the above equation, we can see that greater number of challenged blocks will cause higher error detection probability. By the analysis of [5], for 1 percent tampered blocks in the file, 300 challenged blocks will make $P_a \geq 95\%$ and 460 blocks will make $P_a \geq 99\%$. Thus, our scheme can achieve a high error detection probability.

5 Performance Analysis

In this section, we give the performance analysis and experiment results of our scheme.

5.1 Performance Evaluation

We summarize the computation and the communication cost of our scheme as follow.

Computation Cost. For simplicity, let T_p, $T_{\exp}$, $T_{\mathbb{G}_1-mul}$, $T_{\mathbb{G}_2-mul}$ represent the cost of one pairing operation, one exponentiation operation on group $\mathbb{G}_1$, one multiplication operation on group $\mathbb{G}_1$ and one multiplication operation on group $\mathbb{G}_2$ respectively. The operations like hashing, pseudo-random number generation, pseudo-random permutation, addition and multiplication on Z_q and so on are omitted in our evaluation, because the computation cost of them is negligible. Suppose there are n blocks in total, c challenged blocks and d user subsets for the challenge. To generate the block tags, the algorithm $TagGen$ needs to run n times, whose computation cost is $2nT_{\exp} + nT_{\mathbb{G}_1-mul}$. To challenge the CSP, the verifier needs to run the algorithm $Challenge$, which only causes negligible cost. To generate the integrity proof of challenged blocks, CSP needs to run the algorithm $ProofGen$, which needs $c \cdot T_{\exp} + (c-d) \cdot T_{\mathbb{G}_1-mul}$ computation costs. To check the file integrity, the computation cost of the algorithm $Verify$ is $(d+2)T_p + (c+d)T_{exp} + (c+2d)T_{\mathbb{G}_1-mul} + dT_{\mathbb{G}_2-mul}$. To revoke a group user, the revoked user needs one T_{exp}, the new generator of updated tags needs one T_{exp} and the CSP needs $N(T_{\mathbb{G}_1-mul} + 2T_{exp})$, where N denotes the total count of tags that need to be updated. It is obvious that our scheme moves most part of computation cost to CSP for revoking user.

Communication Cost. In challenge phase, the verifier submits the challenge information $chal = (c, k_1, k_2)$ to CSP. CSP responses the proof $P = (\overline{T}, \overline{F})$ to the verifier. Thus, the communication cost is $(logc + (2+d)|q| + d|\mathbb{G}_1|)$ bits. The communication cost for user revocation is bounded by $3|q| + 2|\mathbb{G}_1|$.

Moreover, we compare the HA-CLS scheme in paper [20] with our scheme from different aspects and the results are shown in the Table 1.

TABLE 1
Comparison with HA-CLS

Schemes	Tag generation	Tag verification	Working scenario
HA-CLS	$2T_{\exp} + 2\,T_{\mathbb{G}_1-mul}$	$3T_p + T_{\exp} + T_{\mathbb{G}_1-mul} + T_{\mathbb{G}_2-mul}$	personal data
Our scheme	$2T_{\exp} + T_{\mathbb{G}_1-mul}$	$3T_p + T_{\exp} + T_{\mathbb{G}_2-mul}$	group data

5.2 Experimental Results

To evaluate the efficiency of our scheme, we implement it using Pairing Based Cryptography (PBC) Library [45] and GNU Multiple arithmetic Precision (GMP) Library [46]. All experiments are implemented by the VMware Workstation 10, in which uses the Ubuntukylin-15.10-desktop-i386 operating system. The configuration of the VMware Workstation is 1 CPU, 4G Ram and 20G disk. We utilize the Lenovo Laptop L440 as the host computer with the setting of Win7 operation system, Core i7-4712MQ@2.3GHz CPU, 8G Ram. The elliptic curve of Type A supplied by PBC is used to construct groups in our experiments. The group order is set to 160-bit which has the equivalent security level as 1024-bit RSA. All the experiments are conducted 50 trials so as to pursue more precise results.

We make the first experiment to evaluate the tag generation cost of our scheme. In this experiment, we set the group user to be 50 and the number of blocks ranges from 100,000 to 1,000,000. Without loss of generality, we suppose the blocks are average assigned to group users. Fig. 2 shows the result of this experiment. As observed, the cost of tag generation increases linearly to the count of blocks. However, to generate tags for 1,000,000 blocks only needs about 112 seconds which can be accepted by most groups. Furthermore, tag generation for a file is usually conducted only once which brings little influence on the performance of integrity checking.

The efficiency of integrity verification relies on the number of challenged blocks and the size of the subset group associated with challenged blocks. However, according to [4], when 1 percent blocks in the file are corrupted, 95 percent error detection probability will be achieved by 300 challenged blocks and 460 challenged blocks will achieve 99 percent error detection probability. Thus, we pay special attention to the two situations and make experiments to evaluate their verification cost. We use 1000 blocks in this experiment and changes the group size from 50 to 100. The experiment results are illustrated in Fig. 3,

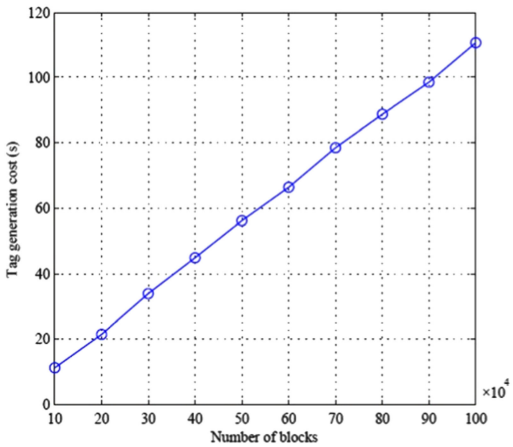


Fig. 2. Computation cost of TagGen.

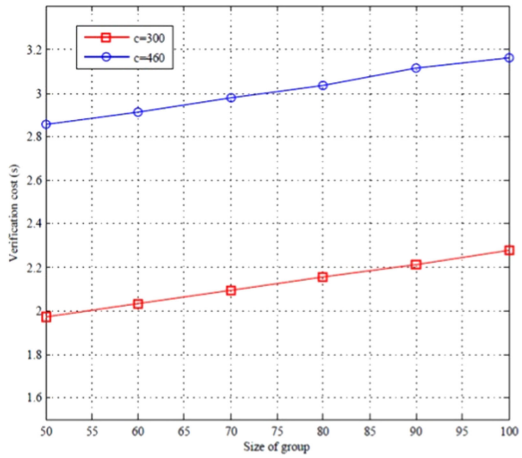


Fig. 3. Computation cost of verification.

from which we find that the verification cost increases proportionally to the size of group for both 300 and 460 challenged blocks. The cost for $c = 460$ is larger than $c = 300$ under the same group size. Specifically, for a group with 100 users, checking the integrity of file consumes 3.18 s when $c = 460$, and consumes 2.28 s when $c = 300$, which is very feasible for applications.

The traditional method for revoking member from group is that a normal user first downloads all the blocks signed by the revoked user, verifies the correctness of them, recomputes the tags of them and then stores the new tags to the cloud server. In our scheme, the cloud server can update the tags by taking the place of normal user. The normal user only needs to compute two parameters. The Fig. 4 depicts the comparison results between our scheme and the traditional method during the user revocation. As we discussed in the above section, our scheme moves most computation operation to CSP, which greatly reduces the burden of group user and efficiently updates the tags generated by the revoked user. From Fig. 4, we see that the cost of revocation is linear with the count of updated tags. It spends about 552 ms to update 100 tags by using traditional method. However, our method only needs 280 ms. Furthermore, the computation cost of traditional method has a bigger increasing speed compared with our scheme. We predict that the gap between our scheme and the traditional method

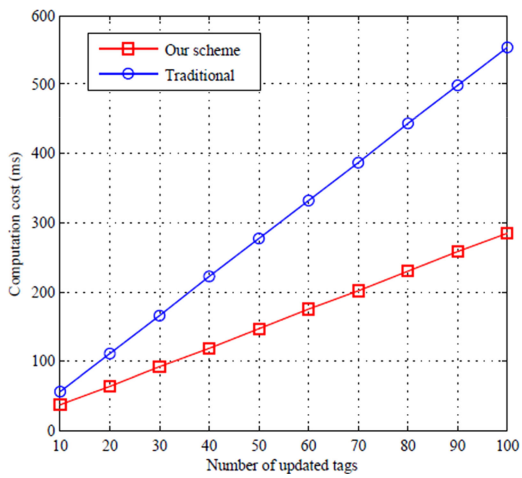


Fig. 4. Computation cost of user revocation.

becomes larger with the count of updated tags growing. Note that the computation cost of our scheme in Fig. 4 is the total cost for the complete procedure of user revocation, which includes the cost of the normal user, the revoked user and the cloud server. If only considering the computation cost of normal user, it is only about 2.8 ms and keeps constant. Thus, our scheme is very efficient.

6 Conclusion

In this paper, we present a novel RDPC scheme for data outsourced on cloud server. Our scheme devotes to solve the integrity checking for the group data which is shared among many clients of a team. We utilize the idea of certificateless signature to generate all the block tags. Because each user of a group has both partial key and secret value, the problem of key escrow is eliminated in our scheme and the certificate management in PKI does not exist. Besides, our scheme supports public verification, efficient user revocation and multi-user data modification. We give the detailed description of the system model and security model of our scheme. At last, based on the CDH and DL assumption, we prove the security of our scheme. The experiment results show that our scheme has good efficiency.

Acknowledgments

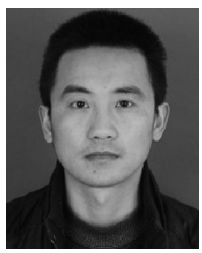
This work was supported in part by the National Natural Science Foundation of China (U1736112, 61772009, 61672207), Jiangsu Provincial Natural Science Foundation of China (BK20161511), the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, the Fundamental Research Funds for the Central Universities (2016B10114), Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, NJUPT.

References

[1] Dropbox for Business. [Online]. Available: https://www.dropbox.com/business, Accessed on: Sep. 16, 2016.
[2] TortoiseSVN. [Online]. Available: https://tortoisesvn.net/, Accessed on: Sep. 16, 2016.
[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comp. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.
[4] Y. Deswarte, J. J. Quisquater, and A. Saïdane, "Remote integrity checking," in *Proc. 6th Working Conf. Integr. Internal Control Inf. Syst.*, 2003, pp. 1–11.
[5] G. Ateniese, et al., "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 598–609.
[6] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int'l Conf. Security Privacy Commun. Netw.*, 2008, pp. 1–10.
[7] F. Sebé, J. Domingo-Ferrer, A. Martinez-balleste, Y. Deswarte, and J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1034–1038, Aug. 2008.
[8] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Comput. Commun. Security*, 2009, pp. 213–222.
[9] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May, 2011.
[10] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.
[11] L. Chen, S. Zhou, X. Huang and L. Xu, "Data dynamics for remote data possession checking in cloud storage," *Comput. Elect. Eng.*, vol. 39, no. 7, pp. 2413–2424, 2013.
[12] Y. Yu, Y. Zhang, J. Ni, M. H. Au, L. Chen, and H. Liu, "Remote data possession checking with enhanced security for cloud," *Future Generation Comput. Syst.*, no. 52, pp. 77–85, 2015.
[13] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013.
[14] H. Yan, J. Li, J. Han, and Y. Zhang, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Trans. Inf. Foren. Sec.*, vol. 12, no. 1, pp. 78–88, Jan. 2017.
[15] Y. Feng, Y. Mu, G. Yang, and J.K Liu, "A new public remote integrity checking scheme with user privacy," in *Proc 20th Australasian Conf. Inf. Security Privacy*, 2015, pp. 377–394.
[16] Y. Zhu, H. Hu, G. J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2231–2244, Dec. 2012.
[17] H. Wang, "Identity-based distributed provable data possession in Multicloud storage," *IEEE Trans. Service Comput.*, vol. 8, no. 2, pp. 328–340, Mar./Apr. 2015.
[18] H. Wang, D. He, J. Yu, and Z. Wang, "Incentive and unconditionally anonymous identity-based public provable data possession," *IEEE Trans. Services Comput.*, vol. PP, no. 99, pp. 1–1. doi: 10.1109/TSC.2016.2633260
[19] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Foren. Sec.*, vol. 11, no .6, pp. 1165–1176, Jun. 2016.
[20] B. Wang, B. Li, H. Li, and F. Li, "Certificateless public auditing for data integrity in the cloud," in *Proc. IEEE Conf. Commun. Network Security*, 2013, pp. 136–144.
[21] H. Wang and J. Li, "Private certificate-based remote data integrity checking in public clouds," in *Proc. 21th Int. Comput. Combinatorics*, 2015, pp. 575–586.
[22] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *Proc. 10th Int. Conf. Applied Cryptography Netw. Security*, 2012, pp. 507–525.
[23] B. Wang, H. Li, and M. Li, "Privacy-preserving public auditing for shared cloud data supporting group dynamics," in *Proc. IEEE Int. Conf. Commun.*, 2013, pp. 1946–1950.
[24] X. Liu, Y. Zhang, B. Wang, and J. Yan, "Mona: Secure multi-owner data sharing for dynamic groups in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1182–1191, Jun. 2013.
[25] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, Jan.-Mar. 2014.
[26] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Service Comput.*, vol. 8, no. 1, pp. 92–106, Jan./Feb. 2015.
[27] Y. Yu, Y. Mu, J. Ni, J. Deng, and K. Huang, "Identity privacy-preserving public auditing with dynamic group for secure mobile cloud storage," in *Proc. 8th Int. Conf. Netw. Syst. Security*, 2014, pp. 28–40.
[28] J. Yuan and S. Yu, "Public integrity auditing for dynamic data sharing with multiuser modification," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1717–1726, Aug. 2015.
[29] A. Juels and B. S. Kaliski Jr., "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 584–597.
[30] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. 14th Int. Conf. Theory Appl. Cryptology Inf. Security*, 2008, pp. 90–107.
[31] K. D. Bowers, A. Juels, and A. Oprea, "Hail: A high-availability and integrity layer for cloud storage," in *Proc. 16th ACM Conf. Comput. Commun. Security*, 2009, pp. 187–198.
[32] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *Proc. 6th Theory Cryptography Conf.*, 2009, pp. 109–127.
[33] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography, " in *Proc. 9th Int. Conf. Theory Appl. Cryptology and Inf. Security*, 2003, pp. 452–473.
[34] D. Boneh, H. Shacham, and B. Lynn, "Short signatures from the weil pairing," *J. Cryptology*, vol. 17, no. 4, pp. 297–319, Sep. 2004.
[35] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans. Service Comput.*, vol. 10, no. 5, pp. 785–796, Sep./Oct. 2017.

[36] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage," *IEEE Syst. J.*, 2017, doi: 10.1109/JSYST.2017.2667679.
[37] H. Wang, D. He, and J. Han, "VOD-ADAC: Anonymous distributed fine-grained access control protocol with verifiable outsourced decryption in public cloud," *IEEE Trans. Services Comput.*, vol. PP, no. 99, pp. 1–1. doi: 10.1109/TSC.2017.2687459
[38] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Service Comput.*, vol. 10, no. 5, pp. 715–725, Sep.-Oct. 2017.
[39] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud Storage," *Int. J. Commun. Syst.*, vol. 30, no. 1, 2017, Art. no. e2942.
[40] J. Li, H. Wang, Y. Zhang, and J. Shen, "Ciphertext-policy attribute-based encryption with hidden access policy and testing," *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 7, pp. 3339–3352, 2016.
[41] H. Qian, J. Li, Y. Zhang, and J. Han, "Privacy preserving personal health record using multi-authority attribute-based encryption with revocation," *Int. J. Inf. Security*, vol. 14, no. 6, pp. 487–497, 2015.
[42] S. Lin, R. Zhang, H. Ma, and M. Wang, "Revisiting attribute-based encryption with efficient verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 10, pp. 2119–2130, Oct. 2015.
[43] J. Li, F. Sha, Y. Zhang, X. Huang, and J. Shen, "Verifiable outsourced decryption of attribute-based encryption with constant ciphertext length," *Security Commun. Netw.*, vol. 2017, 2017, Art. no. 3596205, doi: 10.1155/ 2017/3596205.
[44] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Services Comput.*, 2017, doi: 10.1109/TSC.2017.2710190.
[45] The Pairing-based Cryptography Library (PBC). [Online]. Available: https://crpto.stanford.edu/pbc/download.html, Accessed on: Sep. 16, 2016.
[46] The GNU Multiple Precision Arithmetic Library (GMP). [Online]. Available: http://gmplib.org/, Accessed on: Sep. 16, 2016.

Jiguo Li received the BS degree in mathematics from Heilongjiang University, Harbin, China, in 1996, the MS degree in mathematics and the PhD degree in computer science from Harbin Institute of Technology, Harbin, China, in 2000 and 2003, respectively. During 2006.9-2007.3, he was a visiting scholar in Centre for Computer and Information Security Research, School of Computer Science & Software Engineering, University of Wollongong, Australia. During 2013.2-2014.1, he was a visiting scholar in the Institute for Cyber Security, University of Texas at San Antonio. He is currently a professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China and College of Computer and Information, Hohai University, Nanjing, China. His research interests include cryptography and information security, cloud computing, wireless security, and trusted computing etc. He has published more than 150 research papers in refereed international conferences and journals. His work has been cited more than 2000 times at Google Scholar. He has served as program committee member in more than 20 international conferences and served as the reviewers in more than 90 international journals and conferences.



Hao Yan received the BS and MS degrees in computer science & technology from the Nanjing University of Science and Technology, China, in 2003 and 2006, respectively. He is currently working toward the PhD degree with Hohai University, China. His research interests include cloud computing security, and applied cryptography.

Yichen Zhang received the PhD degree from the College of Computer and Information, Hohai University, Nanjing, China in 2015. She is currently an associate professor in the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China and College of Computer and Information, Hohai University, Nanjing, China. Her research interests include cryptography, network security. She has published more than 30 research papers in refereed international conferences and journals.