A Secure Privacy-Preserving Cloud Auditing Scheme with Data Deduplication

Chen Li and Zhenhua Liu
(Corresponding author: Chen Li)

School of Mathematics and Statistics, Xidian University
No. 2 South Taibai Road, Xi'an, Shaanxi 710071, P.R. China
(Email: lichen0906@foxmail.com)
(Received July 18, 2017; revised and accepted Nov. 5, 2017)

Abstract

In this paper, we propose a privacy-preserving public auditing scheme supporting data deduplication. In the process of auditing, since the authentication tag of a message contains only one element, the storage and transmission cost of the tag can be significantly reduced. Meanwhile, by eliminating user's private key in the response, our scheme achieves unconditional anonymity against third party auditor. Moreover, during data deduplication, Bloom filter is utilized to efficiently check ownership of the data that a user claims to have. For public auditing, the proposed scheme is proven to be uncheatable and anonymous under the variant of the BDH hardness assumption in the random oracle model. And security analysis indicates that our scheme is unforgeable during deduplication. Compared to existing schemes with similar features, our scheme achieves higher security and better functionality through function evaluation and security analysis.

Keywords: Cloud Storage; Data Deduplication; Privacy Preservation; Public Auditing

1 Introduction

With the development of cloud computing, a rising number of enterprises and organizations choose to outsource their data to a third-party cloud service provider, who can provide resource-constrained users with convenient storage and computing services and thus reducing users' storage burden [15, 17]. Although cloud storage offers many advantages, it also brings some security challenges such as data integrity and storage efficiency.

Different from local data, cloud data is stored in an uncertain domain via Internet. Therefore, users surely can suspect the integrity of their data that stored in cloud due to the fact that their data is vulnerable to the attack from both outside and inside of the cloud [7,8]. Once their data is corrupted, cloud server might passively hide some data loss from users to maintain their reputation. Worse, due to the insufficiency of storage space or some other economic reasons, cloud server might even delete users' data and cheat users that their data still stores integrally. To cope with the conflict between resource-constrained users and large amounts of data, it is essential to consider how can users verify the integrity of data efficiently without retrieving them.

Since cloud service is increasingly used, data redundancy inevitably occurs in cloud storage. Research shows that 80% - 90% of cloud data is redundant [12, 22], and this rate is still increasing, which causes a big waste of cloud storage space. In order to save storage space in cloud, a technique called deduplication came into being, in which cloud server keeps only one single copy of data and sends a storage link to every user who possess the data. However, several security threats potentially exists during deduplication [9]. For instance, if a malicious user needs to gain access to a message that already exists in the cloud, he can pass the examination by only owing the hash value of the massage rather than the concrete message. It is obvious that cloud server cannot distinguish whether user indeed possess the data only through matching its hash value. Therefore, how to convince cloud server that user who upload a duplicate of the data indeed possess the data becomes another issue in cloud service.

In this paper, aiming at solving both data integrity and storage efficiency, we concentrate on how to design a secure and efficient public auditing scheme with data deduplication and users' anonymity. Inspired by a Proof of Ownership protocol that Blasco *et al.* [3] designed, we will propose a privacy-preserving auditing scheme with data deduplication which achieves a better trade-off between efficiency and security through improving Wu *et al.*'s auditing scheme [23].

The rest of this paper is organized as follows. A review about some related works is given in Section 2. Some preliminaries are presented in Section 3. The system model and security model for the proposed scheme are described in Section 4. The concrete construction of privacy-preserving auditing scheme with data deduplication is detailed in Section 5. We analyze the proposed

scheme in Section 6. The performance evaluation and efficiency improvement are discussed in Section 7. Finally, some concluding remarks are given in Section 8.

2 Related Works

This section mainly consists of the research advance of three related works: integrity auditing, data deduplication, and auditing schemes with data deduplication. Moreover, a comparison among several related works that achieve both integrity auditing and data deduplication is shown below.

2.1 Integrity Auditing

In order to efficiently verify the integrity of stored cloud data, Ateniese *et al.* [2] came up with the notion of provable data possession (PDP) in 2007. More precisely, under the situation that cloud server could hide data errors for his own benefit, PDP allows cloud server to proof that users' data are completely stored without retrieving the entire data. Considering the size of users' data and users' limited computation resource, outsourced data are not suitable for users themselves to audit in many cases. Therefore, it is a preferable way to introduce a third party auditor (TPA) to ensure data's integrity and availability. Liu *et al.* [16] summarized some existing research situations and development trends of public auditing.

Although PDP can assist user verifying data integrity, TPA may reveal users' identities for personal benefits during public auditing [27]. To preserve users' privacy, Wang *et al.* [19] came up with a public auditing scheme supporting data sharing and privacy-preserving. In Wang *et al.*'s scheme, challenge generated by TPA utilizes all users' public keys, and thus the privacy of user's identity can be realized. Several constructions [11, 20, 21] were subsequently presented. The main solution in these constructions is the anonymity of ring signature or group signature techniques. However, the tag size of ring signature or group signature is significantly large, which causes a higher transmission and verification cost than many traditional signature schemes. Therefore, by reducing the size of authentication tag to only one element, Wu *et al.* [23] proposed an efficient auditing scheme, which can achieve users' identity privacy by eliminating user's private key during a challenge-and-response protocol. Besides, the authentication tag in the scheme is irrelevant to the number of users within the group.

2.2 Data Deduplication

For increasing storage efficiency, cloud server needs to identify and remove redundant data by retaining only one copy of each block (block-level deduplication) or file (file-level deduplication). And data deduplication can take place before data are uploaded to cloud server (client-side deduplication) or after they are uploaded (server-side deduplication) [10]. However, server-side deduplication only reduces storage cost of cloud server instead of reducing bandwidth. Hence, client-side deduplication is more widely used, since user does not need to upload data if a duplicate already exists, and thus reducing bandwidth cost between user and cloud server remarkably.

During a client-side deduplication system, user sends the hash value of data to cloud server and cloud server checks whether the duplicate exists in cloud storage. Nonetheless, Halevi *et al.* [9] explained several security attacks that may occur in client-side deduplication systems. For instance, if a malicious user needs to gain access to a message that already exists in the cloud, he can pass the examination by only owing the hash value of the massage rather than the concrete message. As a solution to these attacks, Halevi *et al.* [9] first introduced the concept of Proof of Ownership (PoW), which has been extended into a number of related works [18, 24], but all require a higher computational complexity. Therefore, based on Bloom filters, Blasco *et al.* [3] introduced a novel efficient PoW protocol that provides a flexible and scalable solution to the weaknesses of client-side deduplication.

2.3 Auditing Schemes with Data Deduplication

From the above discussions, privacy-preserving public auditing and data deduplication are two main branches of the research for efficient cloud storage [13]. So it becomes a significant matter to support these two functions simultaneously. However, a mechanical combination of privacy-preserving public auditing and efficient deduplication mechanisms cannot efficiently solve both data deduplication and integrity auditing. The reason is that storage efficiency contradicts with the authentication tags (*i.e.*, signatures) during public auditing.

To the best of our knowledge, only the following papers achieve both public auditing and data deduplication. There follows some analysis. Yuan and Yu [26] proposed a constant cost storage public auditing scheme supporting data deduplication, but they did not consider the privacy-preserving property. Then Alkhojandi and Miri [1] showed a privacy-preserving public auditing mechanism supporting a variant of client-side deduplication performed by a mediator, but the mediator may reveal users' data during deduplication. Besides, Alkhojandi and Miri's scheme failed to reduce user's bandwidth overhead. Subsequently, Li *et al.* [14] presented a scheme called SecCloud which aims to solve both data integrity and secure deduplication by using a MapReduce cloud to replace TPA. Nevertheless, uploading data to the MapReduce cloud violates the privacy of user's identity. Furthermore, Li *et al.*'s scheme [14] cannot reduce the bandwidth for users. To solve the bandwidth problem, Kardas and Kiraz [13] came up with a secure deduplication scheme that supports client-side deduplication along with privacy-preserving public auditing. However, when uploading a message, user needs to compute at least an asymmetric encryption to complete the PoW protocol,

Table 1: Comparison of auditing mechanisms with deduplication

Schemes	Anonymous	No extra entities	Public auditing	User-side bandwidth reduction (client-side deduplication)
Yuan *et al.* [26]	No	Yes	Yes	Yes
Naelah *et al.* [1]	Yes	No	Yes	No
Li *et al.* [14]	No	Yes	Yes	No
Kardas *et al.* [13]	Yes	No	Yes	Yes
Ours	Yes	Yes	Yes	Yes

which consequentially causes efficiency problem. Besides, the key server may recover part of message's encrypt key through the value received by user. Table 1 compares the function that these schemes [1, 13, 14, 26] can realize.

3 Preliminaries

We now explain some preliminary notions that will form the foundations of our scheme.

3.1 Bilinear Pairings

Let $\mathbb{G}_1, \mathbb{G}_2$ be the cyclic groups of prime order p, g be a generator of $\mathbb{G}_1$, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map [5] with the following properties:

1) **Bilinearity**: $e(g^a, g^b) = e(g, g)^{ab}, a, b \in \mathbb{Z}_p$;
2) **Non-degeneracy**: There exist $u, v \in \mathbb{G}_1$ such that $e(u, v) \neq 1$;
3) **Computability**: For all $u, v \in \mathbb{G}_1$, $e(u, v)$ can be efficiently computed.

3.2 Complexity Assumptions

Definition 1. *Given (g, g^a, g^b, g^c), for random $a, b, c \in \mathbb{Z}_p^*$, the Bilinear Diffie-Hellman(BDH) problem [5] is to compute $e(g, g)^{abc}$.*

A challenger $\mathcal{C}$ has advantage ε in solving the BDH problem if

$$\Pr\left[e(g, g)^{abc} \leftarrow \mathcal{A}(g, g^a, g^b, g^c)\right] \geq \varepsilon.$$

The (ε, t)-BDH assumption holds if no t-time algorithm has the advantage at least ε in solving the BDH problem.

Definition 2. *Given (g, g^a, g^b, g^{ac}), for random $a, b, c \in \mathbb{Z}_p^*$, the variant of the BDH (vBDH) problem [23] is to compute $e(g, g)^{bc}$.*

A challenger $\mathcal{C}$ has advantage ε in solving the vBDH problem if

$$\Pr\left[e(g, g)^{bc} \leftarrow \mathcal{A}(g, g^a, g^b, g^{ac})\right] \geq \varepsilon.$$

The (ε, t)-vBDH assumption holds if for any t-time algorithm, the advantage ε in solving the vBDH problem is negligible.

3.3 Bloom Filter

As a probabilistic data structure, Bloom filter [4] can approximately represent the elements of a set and verify the membership of elements. Since both the storage space and the insert or query time are constant, Bloom filter has the advantage of memory and time efficiency [3]. On the other hand, Bloom filter sacrifices a certain amount of accuracy, since an element that is not in the set may be recognized as being part of the set, which is called as false positives. But false negatives cannot occur in Bloom filter.

Bloom filter consists of k random hash functions $h_1(\cdot), h_2(\cdot), \cdots, h_k(\cdot)$ and an m bit array. When initializing the Bloom filter, all the positions of bit array are set to 0. To insert an element x into Bloom filter, we compute k addresses $a_1 = h_1(x) \mod m, a_2 = h_2(x) \mod m, \cdots, a_k = h_k(x) \mod m$ and set the position of corresponding bit array to 1. To determine whether the element is in the set, we need to compute k hash values $h'_1, h'_2, \cdots, h'_k$ and check if all the corresponding values are 1. With certain false positive rate, the element is in the set if and only if all bits are 1 in Bloom filter. In other words, the element is not in the set if not all the bits are 1.

4 Problem Statement

4.1 System Model

In this system, there are three main entities named cloud server, user and TPA, as shown in Figure 1.

- Cloud server (CS) provides users with cloud storage and computing service. Therefore, user can rent or buy storage space from CS to store their individual data and perform some specific computation with CS's help. The data format stored in CS is a tuple $(index, message, tag)$.
- User computes an index according to a message, and uses her or his secret key to compute the message's tag. Then, the user uploads the tuple $(index, message, tag)$ to CS. During data deduplication, the user does not upload the message when the duplicate exists.

- TPA can verify users' messages by challenging CS with a message index set and the corresponding challenge value. Afterwards, TPA checks the response from CS and sends the auditing result back to users.

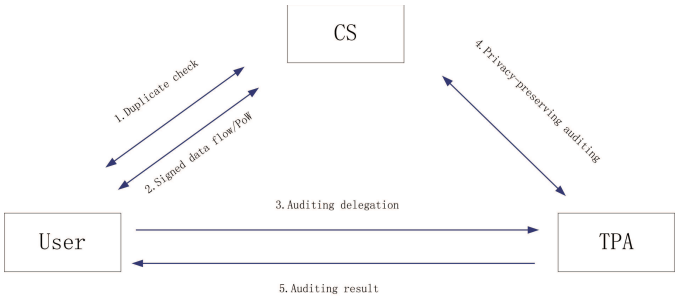

Figure 1: Architecture of a general scheme

Figure 1 shows an outline of the procedure for a general scheme that supports deduplication and public auditing.

1) Before uploading a message, user first checks if there is a duplicate one in CS.
2) When uploading a message, user either sends the message with a corresponding signature or passes the PoW challenge to avoid uploading the message.
3) If user needs to check the integrity of data, she or he sends an auditing delegation to TPA.
4) TPA and CS run the privacy-preserving auditing interactively.
5) After verifying the auditing result, TPA sends the result back to user.

4.2 General Scheme

A scheme supporting public auditing and deduplication [1, 13, 14, 26] is generally composed of these six algorithms, namely **Initialize**, **KeyGen**, **FileUpload**, **Challenge**, **Respond** and **Verify**. The detailed algorithms come as follows.

- **Initialize** (1^k): Take the security parameter 1^k as input, and output the public parameter *params*.
- **KeyGen**: User u_i's secret and public key pair (sk_i, pk_i) are generated by running the key generation algorithm.
- **FileUpload** (sk_i, id_j, m_j): If user u_i needs to upload a message m_j which is identified by the index id_j, then he computes $H(m_j)$ and sends $H(m_j)$ to CS for checking whether the message m_j has been stored in CS at first.

  Case 1: If there is no duplicate of m_j, user u_i computes m_j's authentication tag $\sigma_{i,j}$ using her or his secret key sk_i, generates Bloom filter BF_j by splitting m_j into n blocks $\{m_{j,1}, \cdots, m_{j,n}\}$, and then uploads the tuple $(id_j, m_j, \sigma_{i,j})$ along with the Bloom filter BF_j.

  Case 2: If a duplicate of m_j exists, for $1 \leq t \leq n$, CS randomly chooses t blocks and sends a corresponding identity set $K = \{k_1, \cdots, k_t\}$ to user u_i.

  According to the set K, user u_i computes a set of tokens $\{T_{j,k_q} | q = 1, \cdots, t\}$ and sends the set back to CS for checking whether the tokens are in the Bloom filter BF_j.

- **Challenge** $(pk_1, \cdots, pk_d, s, \mathcal{I})$: Taking public keys of d users, a secret value s and a random index subset $\mathcal{I}$ of the entire storage space as input, TPA computes and sends a challenge *chal* to CS.
- **Respond** $(chal, M, \Sigma)$: When CS receives the challenge, he computes the response (μ, σ_{res}) with a set of messages $M = \{m_j | id_j \in \mathcal{I}\}$ and a set of corresponding tags $\Sigma = \{\sigma_{i,j} | id_j \in \mathcal{I}\}$. Subsequently, the response (μ, σ_{res}) is sent to TPA.
- **Verify** $(\mu, \sigma_{res}, chal, s)$: Using challenge *chal* and secret value s as inputs, TPA checks whether the response (μ, σ_{res}) is correct and outputs "true" if (μ, σ_{res}) pass the validation or "false" otherwise.

4.3 Security or Threat Model

This subsection consists of three security aspects named uncheatable, information-theoretical anonymous and unforgeable of tokens.

Since CS is semi-honest, he may try to deceive users that their data are still securely stored when he is unable to recover the data due to some technical problems or storage devices damage. Moreover, with the risk that TPA can reveal users' identities during auditing process, the anonymity of user should be considered in a general scheme. Besides, a user might attempt to pass the PoW challenge so as to possess a message that the user does not.

Therefore, a general scheme should be uncheatable against adaptive chosen-message attack according to Wu *et al.*'s model [23], achieve information-theoretical anonymity which refers to Zhang and Zhao's model [28], and attain unforgeability of tokens based on Blasco *et al.*'s model [3]. In the rest of this subsection, we formalize the models mentioned above in the form of security model or threat model.

4.3.1 Uncheatability

During this part, CS is regarded as a semi-honest one who could attempt to cheat user. Therefore, a general scheme is uncheatable against adaptive chosen-message attack. In the following description, we consider a security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Setup: $\mathcal{C}$ inputs the security parameter 1^k and runs the **Initialize** and **KeyGen** algorithms. Then $\mathcal{C}$ gives the public parameter *params* and the public keys $(pk_1, \cdots, pk_d)$ of all users to $\mathcal{A}$.

Sign Query: $\mathcal{A}$ could query the sign oracle adaptively for tag of a pair (id_j, m_j) under the public key pk_i that $\mathcal{A}$ chooses. $\mathcal{C}$ returns the corresponding tag $\sigma_{i,j}$ through running the **FileUpload** algorithm.

Challenge: $\mathcal{A}$ chooses set $\mathcal{I}^*$ from all the message indexes, and ensures that at least one index in $\mathcal{I}^*$ has not been queried in the sign oracle before. $\mathcal{C}$ generates a challenge *chal* of $\mathcal{I}^*$ from the **Challenge** algorithm and returns *chal* to $\mathcal{A}$.

Respond: Finally, $\mathcal{A}$ outputs the response (μ, σ_{res}).

We define the advantage of adversary $\mathcal{A}$ in cheating challenger $\mathcal{C}$ as

$$Adv(A) = \Pr\left[\begin{array}{c|c} \textbf{Verify} = \text{“}true\text{”} & \begin{array}{c}(\textbf{params}, pk_1, \cdots pk_d) \\ \leftarrow \textbf{Setup}(1^k) \\ (pk_i, id_j, m_j) \leftarrow \mathcal{A} \\ \sigma_{i,j} \leftarrow \textbf{FileUp}(sk_i, id_j, m_j) \\ \mathcal{I}^* \leftarrow \mathcal{A} \\ chal \leftarrow \textbf{Chal}(pk_1, \cdots pk_d, \mathcal{I}^*) \\ (\mu, \sigma_{res}) \leftarrow \mathcal{A}(chal)\end{array}\end{array}\right]$$

Definition 3. *A general scheme is uncheatable against adaptive chosen-message attack if for any polynomial-time adversary $\mathcal{A}$, the advantage $Adv(\mathcal{A})$ is negligible.*

4.3.2 Information-Theoretical Anonymity

Suppose the challenge that TPA chooses only contains one message's index during **Challenge** and **Respond** algorithms, and TPA attempts to correctly determine the identity of user from the **Challenge** and **Respond** algorithms. Thus, TPA acts as a malicious one in this part.

A general scheme can achieve information-theoretical anonymity described by a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Setup: $\mathcal{C}$ inputs the security parameter 1^k and runs the **Initialize** and **KeyGen** algorithms. $\mathcal{C}$ sends the public parameter *params* and all d users' secret and public key pairs $\{(sk_1, pk_1), \cdots, (sk_d, pk_d)\}$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ chooses a pair (id_j, m_j) and computes the challenge *chal* of this pair by running **Challenge** algorithm.

Respond: $\mathcal{C}$ picks $i \in \{1, \cdots, d\}$ at random and computes the tag $\sigma_{i,j}$ using i-th user's secret key sk_i through the **FileUpload** algorithm. Then, $\mathcal{C}$ generates the response (μ, σ_{res}) from the **Respond** algorithm and returns (μ, σ_{res}) to $\mathcal{A}$.

Guess: $\mathcal{A}$ checks whether the response is correct through the **Verify** algorithm and outputs $i' \in \{1, \cdots, d\}$ if the response passes the verification.

We define the advantage of the adversary $\mathcal{A}$ in distinguishing user's integrity of a pair $(message, tag)$ as

$$Adv(\mathcal{A}) = \left|\Pr(\cdot) - \frac{1}{d}\right|,$$

where

$$\Pr(\cdot) = \Pr\left[\begin{array}{c|c} i' = i & \begin{array}{c}(params, (sk_1, pk_1), \cdots, (sk_d, pk_d)) \\ \leftarrow \textbf{Setup}\,(1^k) \\ (id_j, m_j) \leftarrow \mathcal{A} \\ chal \leftarrow \textbf{Chal}\,(pk_1, \cdots pk_d, id_j) \\ i \leftarrow_R \{1, \cdots d\} \\ \sigma_{i,j} \leftarrow \textbf{FileUp}\,(sk_i, id_j, m_j) \\ (\mu, \sigma_{res}) \leftarrow \textbf{Res}\,(chal, m_j, \sigma_{i,j}) \\ i' \leftarrow \mathcal{A}\,(\mu, \sigma_{res})\end{array}\end{array}\right]$$

Definition 4. *A general scheme achieves information-theoretical anonymity if for any polynomial-time adversary $\mathcal{A}$, the advantage $Adv(\mathcal{A})$ is negligible.*

4.3.3 Unforgeability of Tokens

The existing deduplication schemes [3, 24] did not construct a formal security model, but only give some threat models. Therefore, a threat model conducted by a malicious user is given below.

A malicious user's propose is to pass the PoW challenge for a message m_j he does not own. Suppose that the malicious user possesses several message blocks and the hash value of m_j. Therefore, the malicious user can attempt to forge tokens for passing the PoW challenge. Moreover, a general scheme cannot prevent a malicious user who almost obtains the whole message from passing the PoW challenge. In other words, the malicious user does not need to forge tokens if he already possess the message anyway.

5 Our Construction

In this section, we will construct a concrete scheme which mainly contains two parts. The first one is deduplication, which is conducted by user and CS. Before uploading a message, user divides the message into n blocks, and generates a Bloom filter utilizing a pseudorandom function. The other one is public auditing. Unlike existing auditing works [19, 20, 25] by adopting ring signature or group signature techniques, our scheme uses a constant-size tag generation algorithm, which is a variant of Boneh *et al.*'s signature scheme [6]. Therefore, the transmission and communication cost are less than these existing works [19, 20, 25]. The detailed algorithms of our scheme are shown below.

Initialize(1^k)**:** Take the security parameter 1^k as input, and output the public parameter

$$params = \{\omega, g \in \mathbb{G}_1, H, H_1\},$$

where $H : \{0,1\}^* \to \mathbb{G}_1$, $H_1 : \{0,1\}^m \to \{0,1\}^l$ are two collision resistant hash functions, and m, l are the length of message and token respectively. Let $Prf : \{0,1\}^l \times \{0,1\}^* \to \{0,1\}^\kappa$ be a pseudorandom function [3], where κ is a positive integer.

KeyGen: User u_i selects $x_i \in \mathbb{Z}_p^*$ at random and computes g^{x_i}. Then user u_i's secret and public key pair are $(sk_i, pk_i) = (x_i, g^{x_i})$.

FileUpload(sk_i, id_j, m_j)**:** Suppose that the j-th message is $m_j \in \mathbb{Z}_p$ with an index id_j. Before uploading m_j, user u_i computes and uploads $h_{i,j} = H(m_j)$ to CS. Upon receiving the upload request, CS first checks whether $h_{i,j}$ already exists.

Case 1: If there is no duplicate in CS, *i.e.* m_j does not exist in CS, CS returns "No Duplicate" to user u_i. Then user u_i computes the authentication tag of m_j as

$$\sigma_{i,j} = (H(id_j) \cdot \omega^{m_j})^{1/x_i}.$$

Furthermore, user u_i splits message m_j into n message blocks $\{m_{j,1}, \cdots, m_{j,n}\}$ with equal length. For each message block $m_{j,l} (l = 1, \cdots, n)$, user u_i computes its corresponding token $T_{j,l} = H_1(m_{j,l})$ and a pseudorandom value

$$P_{j,l} = Prf(T_{j,l}, l).$$

Then user u_i inserts every $P_{j,l}$ into Bloom filter BF_j and uploads the tuple

$$(index, message, tag) = (id_j, m_j, \sigma_{i,j})$$

along with the Bloom filter BF_j. After that, CS computes $H(m_j)$ and verifies whether

$$h_{i,j} = H(m_j),$$
$$e(\sigma_{i,j}, g) = e(H(id_j) \cdot \omega^{m_j}, pk_i)$$

hold. If these two euqations hold, CS stores the tuple $(id_j, m_j, \sigma_{i,j})$ along with the Bloom filter BF_j and returns a storage link of message m_j to user u_i. Otherwise, CS returns an error message to user u_i.

Case 2: If the duplicate of m_j exists, user u_i performs a PoW protocol by interacting with CS. Specifically speaking, CS chooses t message blocks at random and sends the identifier set of blocks $K = \{k_1, \cdots, k_t\}$ to user u_i, where $1 \le t \le n$. Upon receiving the set K, user u_i computes each token

$$T_{j,k_q} = H_1(m_{j,k_q}), \text{ for } q = 1, \cdots, t.$$

Then user u_i sends $\{T_{j,k_q} | q = 1, \cdots, t\}$ back to CS. For all t chosen message blocks, CS computes $P_{j,k_q} = Prf(T_{j,k_q}, k_q)$ with the returned $\{T_{j,k_q} | q = 1, \cdots, t\}$. Next, CS checks whether all P_{j,k_q} belong to the Bloom filter BF_j. If yes, a storage link of message m_j is sent to user u_i. Otherwise, returns an error message to user u_i.

Challenge($pk_1, \cdots, pk_d, s, \mathcal{I}$)**:** To check the integrity of users' data, TPA selects a random index subset $\mathcal{I}$ from the whole storage space $\mathcal{S}$, chooses $s \in \mathbb{Z}_p^*$, $h \in \mathbb{G}_1$, and $s_j \in \mathbb{Z}_p^*$ for every $id_j \in \mathcal{I}$ at random. Then TPA computes the challenge

$$chal = (Q, pk_{chal}),$$

where

$$Q = \{(id_j, s_j) | id_j \in \mathcal{I}\}$$

and

$$pk_{chal} = (pk_1^s, \cdots, pk_d^s, h, h^s).$$

Respond($chal, M, \Sigma$)**:** Upon receiving the challenge $chal$ from TPA, CS checks whether

$$e(pk_i^s, h) = e(pk_i, h^s), \text{ for } i = 1, \cdots, d,$$

and computes the response (μ, σ_{res}) from a set of messages $M = \{m_j | id_j \in \mathcal{I}\}$ and a set of corresponding tags $\Sigma = \{\sigma_{i,j} | id_j \in \mathcal{I}\}$, where

$$\mu = \sum_{(id_j, s_j) \in Q} s_j \cdot m_j,$$
$$\sigma_{res} = \prod_{(id_j, s_j) \in Q} e(\sigma_{i,j}^{s_j}, pk_i^s).$$

Verify($\mu, \sigma_{res}, chal, s$)**:** Finally, TPA checks whether

$$\sigma_{res} = e\left(\prod_{(id_j, s_j) \in Q} H(id_j)^{s_j} \cdot \omega^{\mu}, g^s\right),$$

and outputs the verification result.

6 Security Analysis

6.1 Consistency

In this part, we analyze the correctness mainly about the **Challenge** and **Respond** algorithms. During message uploading and integrity verification, assume that the tuple $(id_j, m_j, \sigma_{i,j})$ is stored in the whole storage space $\mathcal{S}$, and the tag $\sigma_{i,j}$ of each message m_j is signed by user u_i. Thus, for $id_j \in \mathcal{S}$, CS stores $(id_j, m_j, \sigma_{i,j})$, where

$$\sigma_{i,j} = (H(id_j) \cdot \omega^{m_j})^{1/x_i}.$$

To check the integrity of users' messages, TPA selects a random index subset $\mathcal{I} \subset \mathcal{S}$, chooses $s \in \mathbb{Z}_p^*$, $h \in \mathbb{G}_1$, and $s_j \in \mathbb{Z}_p^*$ for every $id_j \in \mathcal{I}$ at random.

Then TPA computes the challenge

$$chal = (Q, pk_{chal}),$$

where

$$Q = \{(id_j, s_j) | id_j \in \mathcal{I}\}$$

and

$$pk_{chal} = (pk_1^s, \cdots, pk_d^s, h, h^s).$$

With the challenge *chal* received from TPA, CS computes the response

$$\mu = \sum_{(id_j, s_j) \in Q} s_j \cdot m_j,$$

and

$$\begin{aligned}\sigma_{res} &= \prod_{(id_j, s_j) \in Q} e(\sigma_{i,j}^{s_j}, pk_i^s) \\ &= \prod_{(id_j, s_j) \in Q} e(H(id_j)^{s_j}, g^s) \cdot \prod_{(id_j, s_j) \in Q} e(\omega^{s_j \cdot m_j}, g^s).\end{aligned}$$

Finally, TPA checks the correctness of the response (μ, σ_{res}) by computing

$$\sigma_{res} = e\left(\prod_{(id_j, s_j) \in Q} H(id_j)^{s_j} \cdot \omega^{\mu}, g^s\right).$$

The above analysis indicates that

$$\begin{aligned}\sigma_{res} &= \prod_{(id_j, s_j) \in Q} e(H(id_j)^{s_j}, g^s) \cdot \prod_{(id_j, s_j) \in Q} e(\omega^{s_j \cdot m_j}, g^s) \\ &= e\left(\prod_{(id_j, s_j) \in Q} H(id_j)^{s_j} \cdot \omega^{\mu}, g^s\right).\end{aligned}$$

6.2 Uncheatability

In this section, we prove that our scheme is uncheatable in the random oracle model through the following theorem. The method of the proof is similar to the one in Wu *et al.*'s scheme [23].

Theorem 1. *If there exists an adversary $\mathcal{A}$ that has advantage ε in outputting a valid response of the challenge, then there is a simulation algorithm $\mathcal{C}$ that runs in polynomial time and has advantage at least ε/v in solving the vBDH problem through interacting with $\mathcal{A}$.*

Proof. Suppose that the simulator $\mathcal{C}$ receives an instance of vBDH problem as

$$(p, \mathbb{G}_1, \mathbb{G}_2, e, g, g^a, g^b, g^{ac}),$$

and the propose of $\mathcal{C}$ is to compute the solution $e(g, g)^{bc}$. By interacting with adversary $\mathcal{A}$ who runs in time t, queries hash oracle at most v times and could adaptively query the sign oracle, $\mathcal{C}$ computes the solution as the challenger in the following game.

Setup: $\mathcal{C}$ randomly chooses $r_0, r_1, \cdots, r_d \in \mathbb{Z}_p^*$, computes

$$(g^a)^{r_0}, (g^a)^{r_1}, \cdots, (g^a)^{r_d},$$

sets $\omega = (g^a)^{r_0}$ and selects hash function $H : \{0,1\}^* \to \mathbb{G}_1$, which can be regarded as the random oracle later. Then $\mathcal{C}$ returns the public parameter

$$params = \{g, \omega \in \mathbb{G}_1, H\}$$

and public keys of all d users

$$(pk_1, \cdots, pk_d) = ((g^a)^{r_1}, \cdots, (g^a)^{r_d})$$

to $\mathcal{A}$.

Hash Query: $\mathcal{A}$ can adaptively query the hash oracle for the hash values of messages' indexes. $\mathcal{C}$ maintains a list which is initially empty and randomly chooses $j^* \in \{1, \cdots, v\}$ and $t^* \in \mathbb{Z}_p^*$. If $\mathcal{A}$ queries the hash oracle for the hash value of index id_{j^*}, then $\mathcal{C}$ sets $h_{j^*} = (g^b)^{t^*}$, adds (id_{j^*}, t^*) to the list and returns h_{j^*} back to $\mathcal{A}$. Otherwise, $\mathcal{C}$ selects $t_j \in \mathbb{Z}_p^*$ at random, adds (id_j, t_j) to the list and returns $h_j = (g^a)^{t_j}$ back to $\mathcal{A}$.

Sign Query: $\mathcal{A}$ can adaptively query the sign oracle for the message m_j signed by i-th user's public key pk_i. Assume that $\mathcal{A}$ has queried the hash value of index id_j before, then $\mathcal{C}$ checks the list and finds the corresponding value (id_j, t_j). If $id_j = id_{j^*}$, $\mathcal{C}$ aborts. Otherwise, returns

$$\sigma_{i,j} = g^{t_j/r_i} \cdot g^{r_0 \cdot m_j / r_i}$$

as the tag of the pair (id_j, m_j) under public key pk_i. It is obvious that if

$$\begin{aligned}\sigma_{i,j} &= g^{t_j/r_i} \cdot g^{r_0 \cdot m_j / r_i} \\ &= (g^{a \cdot t_j})^{1/a \cdot r_i} \cdot (g^{a \cdot r_0 \cdot m_j})^{1/a \cdot r_i} \\ &= (H(id_j) \cdot \omega^{m_j})^{1/sk_i},\end{aligned}$$

the tag is valid.

Challenge: $\mathcal{A}$ chooses an index set $\mathcal{I}^*$ of messages. Moreover, $\mathcal{A}$ should ensure that at least one index in set $\mathcal{I}^*$ has not been queried in the sign oracle before. Without loss of generality, we suppose that there is only one index $id_{j'}$ that has not been queried before. If $id_{j'} \neq id_{j^*}$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ selects s_j, for $id_j \in \mathcal{I}^*$ and $y \in \mathbb{Z}_p^*$ at random, and computes

$$pk_{chal} = ((g^{ac})^{r_1}, \cdots, (g^{ac})^{r_d}, (g^a)^y, (g^{ac})^y).$$

Then the challenge $chal = (Q, pk_{chal})$, where

$$Q = \{(id_j, s_j) | id_j \in \mathcal{I}^*\}$$

is sent to $\mathcal{A}$. It is evident that *chal* is a valid challenge since

$(g^{ac})^{r_i} = (g^{a \cdot r_i})^c = pk_i^c$, and $(g^{ac})^y = (g^{ay})^c = h^c$

for $c \in \mathbb{Z}_p^*$ and $h = g^{ay}$ at random.

Respond: Finally, $\mathcal{A}$ outputs the response (μ, σ_{res}).

If $\mathcal{A}$ wins the game, *i.e.* the response can successfully pass the verification, which means that (μ, σ_{res}) satisfies

$$\sigma_{res} = e\left(\prod_{(id_j, s_j) \in Q} H(id_j)^{s_j} \cdot \omega^{\mu}, g^c\right)$$

Therefore, $\mathcal{C}$ can output

$$\left(\frac{\sigma_{res}}{e\left(\prod\limits_{\substack{(id_j, s_j) \in Q \\ id_j \neq id_{j'}}} g^{t_j}, g^{ac}\right)^{s_j} \cdot e\left(g^{r_0 \cdot \mu}, g^{ac}\right)}\right)^{1/(s_{j'} \cdot t^*)}$$
$$= e(H(id_{j'})^{s_{j'}}, g^c)^{1/(s_{i'} \cdot t^*)}$$
$$= e(g, g)^{bc}$$

as the solution of the vBDH problem.

It is obvious that if the simulator $\mathcal{C}$ does not abort and adversary $\mathcal{A}$ could output a valid response of the challenge, $\mathcal{C}$ can successfully output the correct solution of the vBDH problem.

Suppose that $\mathcal{A}$ is able to make **Hash Query** at most v times, and the index set $\mathcal{I}^*$ of messages contains at least one index that $\mathcal{A}$ has not queried before **Challenge** algorithm.

$\mathcal{C}$ selects $j^* \in \{1, \cdots v\}$ at random and sets $h_{j^*} = (g^b)^{t^*}$, which makes $\mathcal{A}$ unable to answer the **Sign Query** for index id_{j^*}. During **Challenge** algorithm, if $\mathcal{C}$ does not abort, then it can be shown that the index which satisfies $id_j = id_{j^*}$ has not been queried before. Since $\mathcal{C}$ could answer all the queries sent by $\mathcal{A}$ except for id_{j^*}, $\mathcal{C}$ does not abort in **Sign Query** during the case that $\mathcal{C}$ did not abort in the **Challenge** algorithm. All in all, the probability that $\mathcal{C}$ does not abort is at least

$$\Pr(\neg abort_{\mathcal{C}}) \geq 1/v.$$

Therefore, if the advantage for $\mathcal{A}$ to output a valid response is ε, $\mathcal{C}$ has at least

$$Adv_{vBDH}(\mathcal{C}) \geq \Pr(\neg abort_{\mathcal{C}}) \cdot Adv(\mathcal{A}) \geq \varepsilon / v$$

advantage solving the vBDH problem. □

6.3 Information-Theoretical Anonymity

Then, we prove that our scheme achieves information-theoretical anonymity [28].

Theorem 2. *Our scheme achieves information-theoretical anonymity,* i.e. *the advantage of any adversary $\mathcal{A}$ in distinguishing the user's identity of a pair $(message, tag)$ is negligible.*

Proof. Suppose that the adversary $\mathcal{A}$ needs to reveal the identity of user who signed the message m_j. Thus, $\mathcal{A}$ interacts with the simulator $\mathcal{C}$ to guess user's identity through the following game.

Setup: $\mathcal{C}$ runs the **Initialize** and **KeyGen** algorithms for all d users' secret and public key pairs. And then $\mathcal{C}$ sends the public parameter **params** to $\mathcal{A}$ along with all the secret and public key pairs.

Challenge: $\mathcal{A}$ chooses a pair (id_j, m_j) and computes

$$pk_{chal} = ((pk_1)^s, \cdots, (pk_d)^s, h, h^s)$$

for $s \in \mathbb{Z}_p^*$ and $h \in \mathbb{G}_1$ at random. Then, $\mathcal{A}$ sends the challenge $chal = (id_j, s_j, pk_{chal})$ to $\mathcal{C}$.

Respond: $\mathcal{C}$ checks whether

$$e(pk_i^s, h) = e(pk_i, h^s)$$

holds for $i = 1, \cdots, d$. If these equations hold, $\mathcal{C}$ randomly picks an $i \in \{1, \cdots, d\}$ and computes the tag

$$\sigma_{i,j} = (H(id_j) \cdot \omega^{m_j})^{1/sk_i}$$

using i-th user's secret key sk_i. Then, $\mathcal{C}$ generates the response (μ, σ_{res}) where

$$\mu = \sum_{(id_j, s_j) \in Q} s_j \cdot m_j,$$
$$\sigma_{res} = \prod_{(id_j, s_j) \in Q} e(\sigma_{i,j}^{s_j}, pk_i^s),$$

and returns the response (μ, σ_{res}) to $\mathcal{A}$.

Guess: $\mathcal{A}$ checks whether the response is valid, and outputs an $i' \in \{1, \cdots, d\}$.

Adversary $\mathcal{A}$ outputs the guess as $i' \in \{1, \cdots, d\}$ representing user's identity. If $\mathcal{A}$ wins the game, *i.e.* the identity is true. Assume $i \neq i'$, then we can easily generate two identical responses which stem from two tags produced by u_i and $u_{i'}$ for the same message. That is to say, the advantage for $\mathcal{A}$ to distinguish user's identity from the response is negligible. As a result, $\mathcal{A}$ can gain no more information from the response than randomly guessing the signer of the tag even if $\mathcal{A}$ holds all users' secret keys. □

6.4 Unforgeability of Tokens

When a user needs to upload a message to CS, the hash value of the message should be sent to CS to check whether a duplicate has been stored in CS or not. If there is no duplicate in CS, user needs to upload the tuple $(index, message, tag)$ and a Bloom filter. However, if there is a duplicate already stored in CS, an additional PoW protocol is needed in order to avoid the case that a malicious user only knows the message's hash value instead of the real message. Briefly speaking, the proposed scheme uses Bloom filters to match the tokens generated by user to conduct the PoW challenge.

Since the parameters in PoW protocol are independent from the main scheme and none of the available PoW schemes utilizing Bloom filter gives out a concrete security proof, security analysis is given in this part. According to a classical PoW scheme proposed by Blasco *et al.* [3], the security of our scheme focuses on the unforgeability of message's tokens.

The PoW challenge requires user to produce tokens for t message blocks at randomly chosen positions. Once received by CS, the tokens are processed with pseudorandom function and checked for membership in the corresponding Bloom filter.

Suppose a malicious user wants to gain access to message m_j. If the malicious user possesses several message blocks and attempts to pass the PoW challenge, he needs to generate all tokens of the t message blocks challenged by CS. Considered by Blasco *et al.* [3], if the malicious user possesses only a few message blocks, the parameters of Bloom filter are set to a proper range, and t is large enough, then the probability that the malicious user successfully forges message blocks' tokens and thus passes the PoW challenge is negligible. We should be aware that a user can pass the PoW protocol when he possesses almost all m_j's message blocks. However, this user can be regarded as a legitimate user for m_j since he almost possess the message.

7 Efficiency Evaluation

In this section, we evaluate the performance of our mechanism and provide a comparison among several schemes [1, 13, 14, 26].

7.1 Communication Overhead

According to the description in Section 4, our mechanism does not introduce communication overhead to users during **Initialize**, **KeyGen** and **Verify** algorithms. The size of the Bloom filter BF_i is $t \cdot |q|$ bits. The size of an auditing message (Q, pk_{chal}) is $k \cdot (|I| + |q|) + p \cdot |q|$ bits. The size of an auditing proof (μ, σ_{res}) is $2t \cdot |q|$ bits. Therefore, if the message is a new one for CS, the total communication overhead of an auditing task is $(k+p+2t+1)|q|+k|I|$ bits, otherwise the total communication overhead is $(t+k+p+2t+1)|q|+k|I|+|m|+t$ bits. It is obvious that if there is a duplicate in CS, the communication overhead is smaller than the case that there is no duplicate. Table 2 provides a comparison between some existing schemes [1, 13, 14, 26] about the communication cost.

7.2 Computation Overhead

As shown in the **FileUpload** algorithm of the proposed scheme, user generates a Bloom filter by splitting message into n blocks, and then computes the authentication tag on the file-level. Under condition that CS receives a new message, the computation cost of uploading a message is $2exp + (n+4)hash + 2mul + 2pair + n \cdot prf$. Otherwise, the corresponding computation cost is $exp + (t+1)hash + t \cdot prf$. Moreover, the computation cost of auditing phase is $(n + 3k + 1)exp + k \cdot hash + 3k \cdot mul + (2n + k)pair$. Therefore, the total computation cost of our mechanism is $(n+3k+3)exp + (n+k+4)hash + (3k+2)mul + (2n + k + 2)pair + n \cdot prf$ if the message is a new one for CS. Otherwise, the total computation cost is $3exp + (n + t + 5)hash + 2mul + 2pair + (n+t)prf$. Evaluating the existing schemes [1, 13, 14, 26], we provide a detail comparison in Table 3 along with some notations in Table 4.

As shown in Table 3, since our scheme can verify the integrity of several messages instead of one message during one challenge-and-response protocol, the efficiency of the proposed scheme is a little lower than Kardas and Kiraz's scheme [13] in **Challenge** and **Respond** algorithms.

However, the proposed scheme has advantage on efficiency during **KeyGen** and **FileUpload** due to the use of asymmetric encryption and signature in Kardas and Kiraz's scheme [13].

Furthermore, Alkhojandi and Miri's scheme [1] and our proposed scheme have almost the same efficiency, while the former has two security problem as discussed in Section 1. So, all these above indicate that the proposed scheme achieves a better trade-off for efficient and secure than the existing schemes.

By utilizing the Pairing Based Cryptography (PBC) Library, an efficiency experiment result is given under the Linux environment. The following experiments run on a personal computer with its configuration parameters as Intel Core i5 2.5 GHz Processor and 4 GB RAM. We assume that the size of element in $\mathbb{G}_1$ and $\mathbb{Z}_p$ is 160 bits, the size of one message block is 2 KB, the size of an element in set $\mathcal{I}$ is 20 bits. The experiment result given below comes from the average of 50 experiments.

Figures 2 and 3 show the communication time changes when k ranges from 100 to 300. Meanwhile, Figure 4 shows the computation time with t ranges from 50 to 250. Figures 2 and 3 indicate that with the increasing number of auditing message blocks, communication in Yuan and Yu's scheme [26], Li *et al.*'s scheme [14] and our scheme has an upward trend, which indicates these three schemes apply to smaller amount of data. However, our scheme has a higher efficiency than Yuan and Yu's scheme [26] and Li *et al.*'s scheme [14]. Though Alkhojandi and Miri's

Table 2: Communication overhead comparison

Scheme	There is a duplicate in CS	The message is a new one for CS
Yuan *et al.* [26]	$(k+t)\|I\|+(k+4)\|q\|+(\frac{t}{n}+1)\|m\|$	$(k+t)\|I\|+(k+4)\|q\|+(\frac{t}{n}+1)\|m\|$
Naelah *et al.* [1]	$k\|I\|+(n+t+6)\|q\|+(\frac{t}{n}+1)\|m\|+k+t$	$k\|I\|+(n+6)\|q\|+2\cdot\|m\|+k$
Li *et al.* [14]	$k\|I\|+(n+2k+t+2)\|q\|+\|m\|+k+t$	$k\|I\|+(3n+2k+3)\|q\|+\|m\|+k$
Kardas *et al.* [13]	$6\|q\|+7\|m\|+t$	$k\|I\|+(n+4)\|q\|+(n+2)\|m\|+n+k+t$
Ours	$k\|I\|+(k+p+3t+1)\|q\|+t$	$k\|I\|+(k+p+2t+1)\|q\|+\|m\|$

Table 3: Computation overhead comparison

Schemes	KeyGen	FileUpload	Challenge	Respond	Verify
Yuan *et al.* [26]	$(n+2)exp$	$(t+1)hash+$ $(2n+t+3)exp+$ $(2t+1)mul+4pair$	-	$k\cdot(n+1)mul+$ $(k+1)exp$	$khash+(k+1)$ $exp+4pair$
Naelah *et al.* [1]	$exp+pair$	$2n\cdot hash+2n\cdot$ $exp+t\cdot mul$	-	$hash+3k\cdot exp+$ $(3k+1)mul+$ $k\cdot pair$	$kt\cdot hash+$ $(kt+2t+1)exp+$ $(t+1)pair$
Li *et al.* [14]	exp	$(n+1)hash+$ $(nt+2)exp+$ $n\cdot(t+3)mul$	-	$(k+1)mul+kexp$	$(k+1)hash+$ $2pair+exp$
Kardas et al. [13]	$2hash+prf$	$(n+1)AsymEnc+$ $n\cdot hash+n\cdot$ $exp+n\cdot mul$	-	$k\cdot exp+2k\cdot$ $mul+pair+hash$	$2pair+(k+4)exp$
Ours	exp	$t\cdot hash+2mul+$ $2pair+exp$	$(n+1)exp$	$(2n+k)pair+$ $2k\cdot mul+k\cdot exp$	$k\cdot hash+2k\cdot exp$ $+k\cdot mul+pair$

Table 4: Notations

Notation	Significance
exp	One exponentiation operation
$hash$	One hashing operation
mul	One multiplication operation
$pair$	One pairing operation on $e:\mathbb{G}_1\times\mathbb{G}_1\to\mathbb{G}_2$
prf	One pseudorandom function operation
$AsymEnc$	An asymmetric encryption or signature
$\|I\|$	The size of an element of set $\mathcal{I}$
$\|q\|$	The size of an element of $\mathbb{Z}_p$ or $\mathbb{G}_1$
$\|m\|$	The size of the message m
p	The number of users within the group
k	The number of selected blocks during challenge
n	The number of blocks in one message m_i
t	The number of blocks CS choose to challenge users during PoW

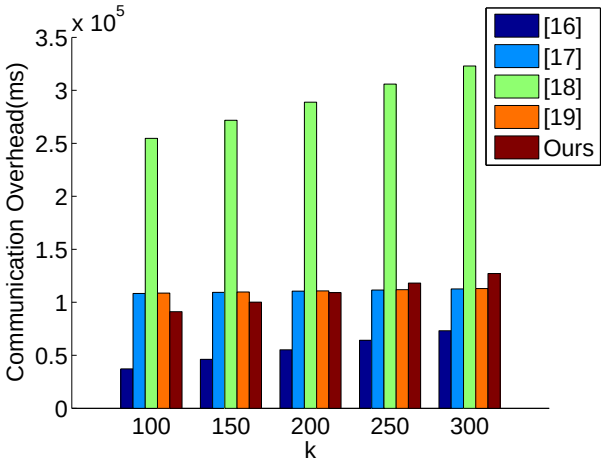

Figure 2: Communication overhead when there is no duplicate

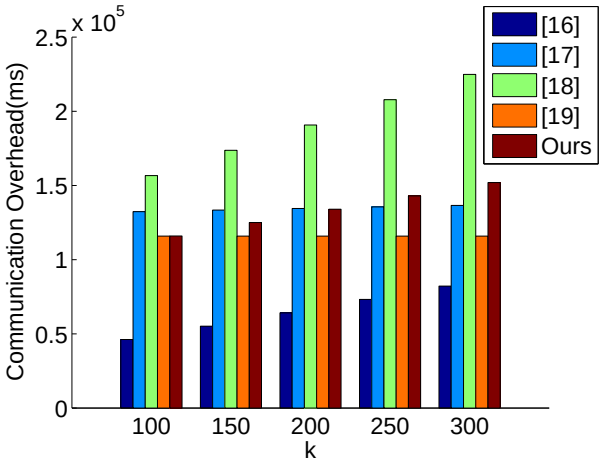

Figure 3: Communication overhead when there is a duplicate

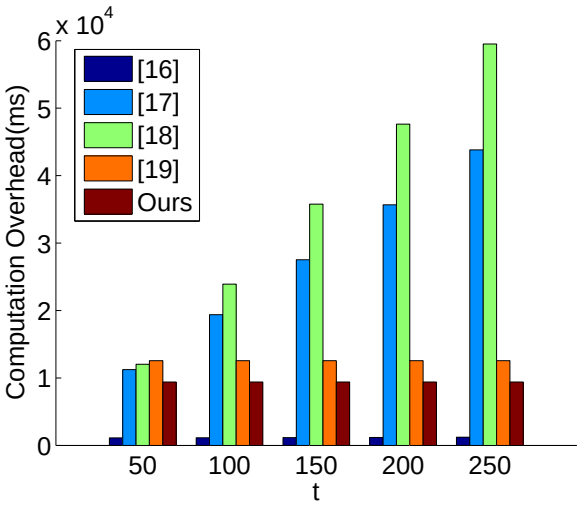

Figure 4: Computation overhead

scheme [1] and Kardas and Kiraz's scheme [13] both have the advantage of communication time compared with our scheme, Figure 4 indicates that the computation time of their schemes are higher than ours. Thus, the experimental results are in great agreement with the above theoretical analysis.

8 Conclusions

In this paper, we have proposed a privacy-preserving public auditing system with data deduplication, which provides data integrity and storage efficiency in cloud computing. Traditional privacy-preserving auditing schemes apply ring signature or group signature to achieve anonymity. And this kind of technology inevitably causes the tag size significantly large. Thus, we use another way to guarantee users' identity privacy against the TPA in order to reduce the tag size. In the proposed scheme, the tag generation algorithm reduces the tag size to only one element. On the other hand, our scheme uses Bloom filter to perform PoW protocol during deduplication, which can be more efficient than some state of the art solutions. Efficiency analysis indicates that the proposed scheme achieves a better trade-off between efficiency and function compared with existing schemes with similar features.

Acknowledgments

We are grateful to the anonymous reviewers for their invaluable suggestions. This work is supported by the National Key R&D Program of China under Grant No. 2017YFB0802000, the National Natural Science Foundation of China under Grants No.61472470 and 61572390, and the Scientific Research Plan Project of Education Department of Shaanxi Province under Grant No.17JK0362.

References

[1] N. Alkhojandi, and A. Miri, "Privacy-preserving public auditing in cloud computing with data deduplication," *International Symposium on Foundations and Practice of Security (FPS'14)*, pp. 35–48, Nov. 2014.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, pp. 598–609, Oct. 2007.

[3] J. Blasco, R. D. Pietro, A. Orfila, and A. Sorniotti, "A tunable proof of ownership scheme for deduplication using Bloom filters," in *IEEE Conference on Communications and Network Security (CNS'14)*, pp. 481–489, Oct. 2014.

[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[5] D. Boneh, and M. Franklin, "Identity-based encryption from the Weil pairing," *Annual International Cryptology Conference (CRYPTO'01)*, pp. 213–229, Aug. 2001.

[6] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'01)*, pp. 514–532, Dec. 2001.

[7] Z. Cao, L. Liu, O. Markowitch, "Analysis of one scheme for enabling cloud storage auditing with verifiable outsourcing of key updates," *International Journal of Network Security*, vol. 19, no. 6, pp. 950–954, 2017.

[8] P. S. Chung, C. W. Liu, and M. S. Hwang, "A study of attribute-based proxy re-encryption scheme in cloud environments", *International Journal of Network Security*, vol. 16, no. 1, pp. 1-13, 2014.

[9] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*, pp. 491–500, Oct. 2011.

[10] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," *IEEE Symposium on Security and Privacy (S&P'10)*, vol. 8, no. 6, pp. 40–47, May 2010.

[11] W. F. Hsien, C. C. Yang, and M. S. Hwang, "A survey of public auditing for secure data storage in cloud computing," *International Journal of Network Security*, vol. 18, no. 1, pp. 133–142, 2016.

[12] M. S. Hwang, T. H. Sun, C. C. Lee, "Achieving dynamic data guarantee and data confidentiality of public auditing in cloud storage service," *Journal of Circuits, Systems, and Computers*, vol. 26, no. 5, 2017.

[13] S. Kardas, and M. S. Kiraz, "Solving the secure storage dilemma: An efficient scheme for secure deduplication with privacy-preserving public auditing," *Cryptology ePrint Archive*, Report 2016/696, 2016.

[14] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2386–2396, 2016.

[15] L. Liu, Z. Cao, C. Mao, "A note on one outsourcing scheme for big data access control in cloud," *International Journal of Electronics and Information Engineering*, vol. 9, no. 1, pp. 29–35, 2018.

[16] C. W. Liu, W. F. Hsien, C. C. Yang, and M. S. Hwang, "A survey of public auditing for shared data storage with user revocation in cloud computing," *International Journal of Network Security*, vol. 18, no. 4, pp. 650–666, 2016.

[17] Y. Ming, Y. Wang, "On the security of three public auditing schemes in cloud computing," *International Journal of Network Security*, vol. 17, no. 6, pp. 795–802, 2015.

[18] W. K. Ng, Y. Wen, and H. Zhu, "Private data deduplication protocols in cloud storage," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC'12)*, pp. 441–446, Mar. 2012.

[19] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 43–56, 2014.

[20] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *International Conference on Applied Cryptography and Network Security (ACNS'12)*, pp. 507–525, June 2012.

[21] B. Wang, H. Li, and M. Li, "Privacy-preserving public auditing for shared cloud data supporting group dynamics," in *IEEE International Conference on Communications (ICC'13)*, pp. 1946–1950, June 2013.

[22] Z. Wang, Y. Lu, and G. Sun, "A policy-based deduplication mechanism for securing cloud storage," *International Journal of Electronics and Information Engineering*, vol. 2, no. 2, pp. 70–79, 2015.

[23] G. Wu, Y. Mu, W. Susilo, and F. Guo, "Privacy-preserving cloud auditing with multiple uploaders," in *International Conference on Information Security Practice and Experience*, pp. 224–237, 2016.

[24] J. Xu, E. C. Chang, and J. Zhou, "Weak leakage-resilient client-side deduplication of encrypted data in cloud storage," in *Proceedings of 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, pp. 195–206, 2013.

[25] 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 *International Conference on Network and System Security (NSS'14)*, pp. 28–40, Oct. 2014.

[26] J. Yuan, and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in *IEEE Conference on Communications and Network Security (CNS'13)*, pp. 145–153, May 2013.

[27] J. Zhang, P. Li, and M. Xu, "On the security of an mutual verifiable provable data auditing in public cloud storage," *International Journal of Network Security*, vol. 19, no. 4, pp. 605–612, 2017.

[28] J. Zhang, and X. Zhao, "Efficient chameleon hashing-based privacy-preserving auditing in cloud storage," *Cluster Computing*, vol. 19, no. 1, pp. 47–56, 2016.

Biography

Chen Li is a master degree student in the School of Mathematics and Statistics at Xidian University. Her interest focuses on cryptography and network security.

Zhenhua Liu is a professor in the School of Mathematics and Statistics at Xidian University, Xi'an, China. His research interests include public key cryptography, cryptographic theory and security protocols in cloud computing.