Privacy-preserving TPA Auditing Scheme Based on Skip List for Cloud Storage

Haichun Zhao[1,2], Xuanxia Yao[1], and Xuefeng Zheng[1]

(Corresponding author: Xuanxia Yao)

School of Computer and Communication Engineering, University of Science and Technology Beijing[1], Beijing, China

School of Computer Science and Technology, Inner Mongolia University for Nationalities[2], Tongliao, China

(Email: yaoxuanxia@163.com)

(Received Jan. 3, 2018; Revised and Accepted June 15, 2018; First Online Jan. 25, 2019)

Abstract

Recently, researchers have proposed several privacy-preserving public auditing schemes to remotely check the integrity of outsourced data based on homomorphic authenticators, random block sampling and random masking techniques. However, almost all these schemes require users to maintain tables related to the block index. These tables are difficult to maintain, especially when the outsourced data is frequently updated. In this paper, we propose a privacy-preserving public auditing scheme with the support of dynamics using rank-based authenticated skip list for the integrity of the data in cloud storage, of which users do not need to maintain the relevant table. And we give a formal security proof for data integrity guarantee and analysis for privacy-preserving property of the audit protocol. The performance analysis demonstrates that our scheme is highly efficient.

Keywords: Audit Protocol; Cloud Storage; Privacy-preserving; Public Auditing; Rank-based Authenticated Skip List

1 Introduction

Cloud computing has many advantages; this has led to an increasing number of individuals and companies choosing to store their data and conduct their business using cloud-based services [22]. Unlike traditional systems, users lose their physical control over their data. Although the cloud infrastructure is significantly more reliable than personal computing devices, data security/privacy is still one of the core considerations for users when adopting cloud services because of the internal and external threats associated with cloud services [1, 35, 38]. Therefore, researchers have proposed various security models and schemes to overcome the issue of data integrity auditing [3, 12–15, 18, 20, 27, 29–31, 33, 34, 36, 37].

The public auditable schemes allow external parties, in addition to the user, to audit the integrity of outsourced data; however, this could potentially leak the user's data to auditors. Hence, researchers have proposed privacy-preserving public auditing schemes to avoid auditors learning user's data in the auditing phase. The construction of the signatures in some of these schemes involve the block index information i, such as $H(name \parallel i)$ or $H(B_i \parallel V_i \parallel R_i)$ [30, 36, 37]. Users need to maintain a table in the local storage for each file, such as map-version Table [5] or index-hash table [36, 37]. The table is also sent to the third-party auditor (TPA) before the data is audited. If the table is corrupted, effective audits or dynamic operations cannot be conducted on the outsourced data. In addition, if a large file is stored in cloud storage server (CSS) and undergoes frequent insert and delete operations, the block index will continue to increase and become very large. This is because the block index cannot be reused. Consequently, it becomes increasingly difficult for users to maintain the table. To address the problem, the index i is removed, and $H(m_i)$ is used in constructing the signature for block m_i to prevent replay attack on the same hash values. To support privacy-preserving TPA auditing, $(H(m_i))^{\alpha/\beta}$ is used in the signature construction and assigned to the data item value for the leaf node of the skip list [9].

In this paper, a secure public auditing algorithm is proposed with the support of dynamics using a rank-based authenticated skip list [9] for the outsourced data. The contributions of this paper can be summarized as follows:

1) A privacy-preserving public auditing scheme which fully supports dynamics by employing rank-based authenticated skip list is proposed. $(H(m_i))^{\alpha/\beta}$ is used as the data item of the bottom node of the skip list to realize privacy-preserving.

2) Based on the cryptography reduction theory [16, 21] and *CPoR's* model [27] a formal security proof

is given for the integrity guarantee of outsourced data and privacy-preserving property of the auditing phase for the scheme.

The remainder of the paper is organized as follows. Section 2 contains the related work. Section 3 introduces the system model and our design goals. In Section 4, we elaborate our proposed scheme. Section 5 analyzes the security and performance of our scheme. The conclusion is given in Section 6.

2 Related Work

Ateniese *et al.* proposed the provable data possession (*PDP*) model, which can be used for remotely checking data integrity [2, 3]. This model can generate probabilistic proofs of possession by randomly sampling data blocks from the server, in which the tags of the sampled blocks can be aggregated into a single value using homomorphic verifiable tags(*HVTs*). It is believed to be the first scheme to provide blockless verification and public verifiability at the same time. Erway *et al.* proposed dynamic PDP (*DPDP*), which applies the structure of *rank-based authenticated skip list* to ensures the integrity of the tags using the skip list structure and the integrity of the blocks by their tags. This scheme effectively supports provable secure updates to the remotely stored data [9]. Juels and Kaliski presented the proof of retrievability (*PoR*) model. This model ensures both the possession and the retrievability of outsourced data by using spot-checking and error-correcting codes. However, the number of audit challenges a user can perform is predetermined and public auditability is not supported in [15].

Shacham *et al.* designed a compact version of *PoR* (*CPoR*) [27] and proved the security of their scheme against arbitrary adversaries in the Juel-Kaliski model. The construction of the publicly verifiable *CPoR* scheme is based on Boneh-Lynn-Shacham (*BLS*) signatures [8]. Wang *et al.* proposed a public auditing scheme that supports dynamic data operations in [31]. The authentication information of the scheme is managed using the Merkle hash tree (MHT) [23], in which the leaf nodes are the values of $H(m_i)$(m_i is the i-th block of the file). To prevent *TPA* extracting data content from the collected information, they designed a privacy-preserving public auditing scheme using a random mask technique to blind the response information in the follow-up work [30]. But its description for the dynamics is ambiguous. Zhu *et al.* proposed another privacy-preserving public auditing scheme which supported dynamic data updates employing an index-hash table [36]. However, in these two privacy-preserving schemes, block index related information is involved in the signature construction. Users are required to maintain a relevant table. To guarantee the integrity of the multiple replicas in cloud, Curtmola *et al.* proposed the replication-based remote data auditing scheme, called Multiple-Replica PDP (*MR-PDP*), which extends the (single-copy) *PDP* scheme for overcoming the collusion attack in a multi-server environment. However, *MR-PDP* only supports private verification [7].

Barsoum *et al.* proposed two multi-copy *DPDP* public auditing schemes, supporting data dynamics based on the MHT and map-version table, respectively. Different copies are generated through encrypting the concatenation of the copy number and file blocks [5]. In the latter, the map-version table must be stored in the local storage of the user and is managed by the user during the various update operations performed on the file. In [34], Yang *et al.* propose a public auditing scheme for shared cloud data in which a group manager is introduced to help members generate authenticators to protect the identity privacy. This method uses two lists to record the members who performed the most-recent modification on each block to achieve the identity traceability. This scheme also achieves data privacy during authenticator generation by employing a blind signature technique. To overcome the issue of resource-constrained users dealing with heavy burdens, Shen *et al.* proposed a cloud storage auditing scheme for group users by introducing a third party medium (*TPM*) to perform time-consuming operations on behalf of users [29]. Utilizing proxy re-signatures and homomorphic linear authenticators, Li *et al.* propose a privacy-preserving cloud data audit scheme that can support key-updating and authenticator-evolving [18].

Researchers have proposed a number of cloud storage auditing schemes in the recent past. All these schemes primarily focus on several different aspects of cloud storage auditing. However, almost none of these schemes address the issue that users need to maintain a block index related table in the local storage for the privacy-preserving public auditing schemes. Users should be "stateless" and must not be required to store and update the table between different dynamic operations, since such table is difficult to maintain.

3 Problem Statements

3.1 System Model

The auditing system for cloud storage involves cloud users, *CSS* and *TPA* as shown in Figure 1. The cloud user is the data owner, who has large amount of data to be stored in the *CSS*. The users can access and dynamically update their data in the *CSS* by interacting with the *CSS*. The *CSS*, which is managed by the cloud service provider (*CSP*), has significant storage space and massive amount of computational resources. The users' data is stored in the cloud storage and is managed and maintained by the *CSP*. The *TPA* has expertise and capabilities that users do not have and can audit the users' outsourced data in the *CSS* on behalf of users at the users' request.

To ensure the integrity and correctness of the users' outsourced data, users need to make periodic checks. To save computation resources and network bandwidth, users can delegate the *TPA* to perform the periodic data integrity verification. However, users do not want informa-

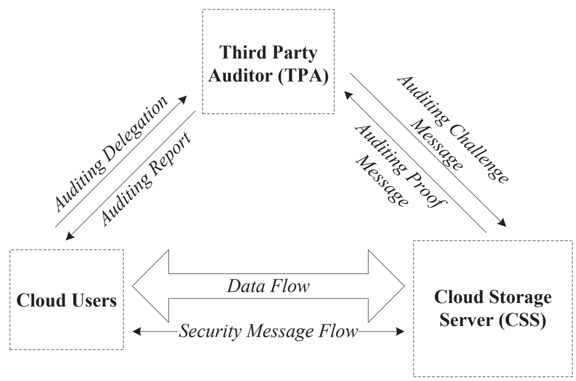

Figure 1: The cloud storage architecture includes the *CSS*, the cloud users and the *TPA*

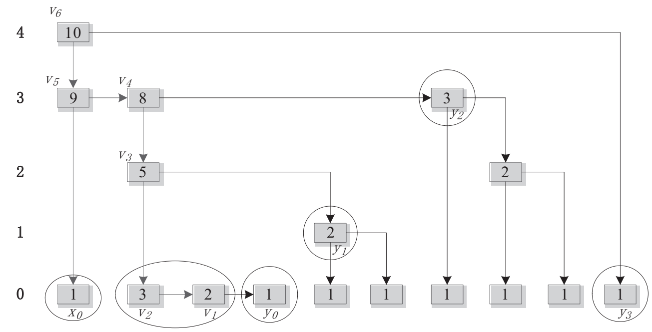

Figure 2: Example of a rank-based skip list

tion from their data to be learned by the *TPA* during the auditing process.

In this model, it is assumed that the cloud server does not have the incentive to reveal their hosted data to any external entity. It is also assumed that the *TPA* has no incentive to collude with either the *CSP* or the user during the auditing process. However, it is interested in the users' data.

3.2 Design Goals

In the aforementioned model, a scheme is proposed in which the design goals can be summarized as follows [19]:

1) Public auditability: To allow any authorized *TPA* to verify the integrity of the cloud data without retrieving a copy of the whole data;

2) Storage correctness: To ensure that there no *CSP* exists that can pass the audit of the *TPA* without storing cloud users' data intact;

3) Privacy preserving: To ensure that it is infeasible for the *TPA* to recover the user's data from the information collected during the auditing phase;

4) High performance: The *TPA* can perform data auditing with minimum communication and computation overhead;

5) Dynamic data: To allow the data owners to modify, insert and delete data blocks in the cloud storage when they want to update their data at any time;

6) Batch auditing: The *TPA* can audit the data of different users at the same time.

4 The Proposed Construction

4.1 Preliminaries

Relevant functions. A pseudo-random function (*PRF*) f and a pseudo-random permutation (*PRP*) π are used with the following parameters [3]:

$$f_k : \{0,1\}^{log_2 n} \times K \to \{0,1\}^l;$$
$$\pi_k : \{0,1\}^{log_2 n} \times K \to \{0,1\}^{log_2 n}.$$

Bilinear maps. Suppose a group G is a Gap Diffie-Hellman (*GDH*) group with prime order p. G_T is another multiplicative cyclic group with prime order p. Then, the bilinear map is a map $e : G \times G \to G_T$ with the following properties [8]:

1) Bilinearity $- \forall u, v \in G, a, b \in Z_p, e(u^a, v^b) = e(u, v)^{ab}$;

2) Non-degeneracy – $e(g, g) \neq 1$, where g is a generator of G;

3) Computability – e should be efficiently computable.

The following scheme description uses the symmetric bilinear map for the purpose of simplicity. The asymmetric bilinear map is in the form of $e : G_1 \times G_2 \to G_T$.

Rank-Based Skip List [9,11,24]. The main information related to i-th node v on level 0 (bottom-level) includes: the level of i-th node $l(v)$, the rank of i-th node $r(v)$, the data item of i-th node $T(m_i)$ and the label of i-th node $f(v)$; that on non-bottom level includes: the level of the node $l(v)$, the rank of the node $r(v)$, the label of the node $f(v)$; In addition to these, each node contains some information related to the structure of the skip list, such as, right and down pointers.

The rank value of a node indicates the number of the reachable bottom nodes (or leaf nodes) departing from the node. The rank of a Null node equals 0. The location of each bottom node can be calculated from the rank values of the relevant nodes.

The label value of a node on bottom-level is

$$f(v) = h_2(l(v) \parallel r(v) \parallel T(m_i) \parallel f(right(v)))$$

and that on non-bottom level is

$$f(v) = h_2(l(v) \parallel r(v) \parallel f(down(v)) \parallel f(right(v)))$$

where the symbol "$||$" denotes concatenation, $f(down(v))$ and $f(right(v))$ are the label of the down and right node of v, respectively. The label value of a Null node is 0. The function $h_2(\cdot)$ is a collision-resistant cryptographic hash function. Users hold the label $f(s)$ of the top leftmost node (or start node) of the skip list. The $f(s)$ is called the basis (or *root*). It is equivalent to the user's verification metadata.

To obtain the proof information of some block i, the skip list needs traversing from the start node v_k to the node v_1 associated with block i through the rank of the nodes. The reverse path $v_1, \cdots, v_k$ is called *verification path* of the block i, as shown in Figure 2. The information of the nodes $x_0, y_0 - y_3$ and $v_1 - v_2$ is used as auxiliary authentication information (AAI) for calculating each rank and label value from v_1 to v_k on the *verification path*.

The proof of a block is composed of a sequence of tuples made of the relevant information of each node on the *verification path*. That is, the proof for block i with data $T(m_i)$ is a sequence $\Pi_i = (A(v_1), \cdots, A(v_k))$ where $A(v_j) = (l(v_j), q(v_j), d(v_j), g(v_j)), 1 \leq j \leq k$, from which we can get the AAI. The $l(v_j)$ is the level of the node and Boolean $d(v_j)$ indicates whether v_{j-1} is to the right or below v_j. The value of $g(v_j)$ is used to calculate the label of the corresponding node along the *verification path*. For the non-bottom level nodes, if $d(v_j) = rgt$, then $g(v_j) = f(dwn(v_j))$, else if $d(v_j) = dwn$, then $g(v_j) = f(rgt(v_j))$. For bottom-level nodes $v_j (j > 1)$ on the *verification path*, the value of $g(v_j)$ is the data item of the node. The value of $g(v_1)$ is the label of the right node of v_1. For nodes at the bottom-level, $q(v_1)$ is the sum of the rank of the right node of v_1 and 1, this 1 means that the node v_1 itself is also a reachable node on the bottom-level. The value of $q(v_j)$ of each node on the left side of the node v_1 at bottom-level is 1. For non-bottom level nodes, if the node v_{j-1} is the right (or down) node of v_j, then $q(v_j) = r(dwn(v_j))$ (or $q(v_j) = r(rgt(v_j))$).

4.2 The Privacy-preserving Scheme

The notions proposed in [3, 15, 27, 28, 30, 31, 36] were followed in this study. The proposed scheme is based on *CPoR's* model [27] and the relevant method in [25].

The scheme consists of two algorithms $KeyGen(1^k), St(sk, F)$ and an interactive audit protocol $Audit(CSP, TPA)$.

Let $S = (p, G, G_T, e)$ be a bilinear map group system with randomly selected generators $g, h \in_R G$, where G, G_T are two groups of large prime order p. $H(\cdot)$ is a secure map-to-point hash function: $\{0,1\}^* \rightarrow G$, which maps strings uniformly to G. Another hash function $h_1(\cdot) : G_T \rightarrow Z_p$ maps the group element of G_T uniformly to Z_p.

$KeyGen(1^k)$: This randomized algorithm generates the public and secret parameters. The cloud user chooses a random signing key pair (*spk,ssk*) and two random $\alpha, \beta \in_R Z_p$. The secret parameter is *sk*= (α, β, ssk) and the public parameter is *pk*=(g,h,X,Y) , where $X = h^\alpha$, $Y = h^\beta \in G$.

$St(sk, F)$: The data file F is split into $n \times s$ sectors $F = \{m_{ij}\}^{n\times s}, m_i = \{m_{ij}\}_{1\leq j\leq s}, m_{ij} \in Z_p$. The cloud user chooses s random $\tau_1, \cdots, \tau_s \in Z_p$ as the secret of the file and computes $u_j = g^{\tau_j} \in G, 1 \leq j \leq s$ and authenticator

$$\sigma_i \leftarrow (H(m_i))^\alpha \cdot (\prod_{j=1}^{s} u_j^{m_{ij}})^\beta = ((H(m_i))^{\alpha/\beta} \cdot g^{\sum_{j=1}^{s}\tau_j \cdot m_{ij}})^\beta \quad (1)$$

for each block i. The cloud user constructs a *rank-based authenticated skip list* of which the data item of the i-th bottom node is $(H(m_i))^{\alpha/\beta}, 1 \leq i \leq n$. Let $\Phi = (\sigma_1, \cdots, \sigma_n)$ and t_0 *be* "*fn* $\| n \| u_1 \| \cdots \| u_s \| M_c$", *fn* is chosen by the user uniformly at random from Z_p as the identifier of file F, M_c is the root of the skip list. The cloud user computes $t = t_0 \| SSig_{ssk}(t_0)$ as the file tag for F under the private key *ssk*. The user then sends $\{F, \Phi, t\}$ and the skip list to the cloud server and deletes $\{F, \Phi\}$ and the skip list from his local storage. Then the user holds t as the *metadata*.

$Audit(CSP, TPA)$: This is a 3–move protocol between *TPA* and *CSP* as the following:

- *Commitment*(*CSP* $\rightarrow$ *TPA*): The *CSP* chooses s random $\lambda_j \in_R Z_p, (1 \leq j \leq s)$, then computes $T_j = u_j^{\lambda_j}, (1 \leq j \leq s)$ and sends its commitment, $\{T_j\}_{j\in[1,s]}$, to *TPA*.
- *Challenge*(*TPA* $\rightarrow$ *CSP*): The authorized *TPA* first retrieves the file tag t. The *TPA* checks the validity of t via *spk*, and quits by outputting *reject* if the verification fails. Otherwise, the *TPA* recovers the values in t_0. Then *TPA* generates a set of challenge information $Chal = \{c, k_1, k_2\}$ [3], in which c is the number of the data blocks to be audited and k_1, k_2 are randomly chosen keys for the pseudo-random permutation π_k and pseudo-random function f_k, respectively. The π_k and f_k are used to generate c indices $s_j (1 \leq j \leq c, 1 \leq s_j \leq n)$ and c relevant coefficients $v_i (i \in \{s_j | 1 \leq j \leq c\}, v_i \in_R Z_p)$ of the challenged data blocks. Let I denotes the set of c random indices s_j. Let Q be the set $\{(i, v_i)\}_{i\in I}$ of the index and coefficient pairs. Then *TPA* sends *Chal* to the prover *CSP*.
- *Response*(*TPA* $\leftarrow$ *CSP*): Upon receiving the challenge *Chal*, *CSP* chooses a random $r \in_R Z_p$ and calculates

$$\psi = e(g^r, h), \gamma = h_1(\psi), s_j = \pi_{k_1}(j),$$

and

$$v_i = f_{k_2}(j),$$

where

$$1 \leq j \leq c, i \in \{s_j | 1 \leq s_j \leq n\}, v_i \in Z_p.$$

Then the *CSP* computes

$$\begin{cases} \sigma \leftarrow \prod_{(i,v_i)\in Q} \sigma_i^{\gamma \cdot v_i} \cdot g^r \\ \mu_j \leftarrow \lambda_j^{-1} \cdot (\gamma \cdot \sum_{(i,v_i)\in Q} v_i \cdot m_{ij} + 1) \end{cases} \quad (2)$$

and sends the response $\theta = (\sigma, \{\mu_j\}_{j\in[1,s]}, \psi)$, the set $\{\Pi_i\}_{i\in I}$ of the proof for every block i and $\{(H(m_i))^{\alpha/\beta)}\}_{i\in I}$ to the *TPA*.

Verification: *TPA* calculates the root R_t from $\{(H(m_i))^{\alpha/\beta}, \Pi_i\}_{i\in I}$ and checks $R_t \stackrel{?}{=} M_c$. If it is not true, *TPA* outputs *reject*, otherwise *TPA* can check

$$e(\sigma,\ h) \stackrel{?}{=} \psi \cdot e(\prod_{(i,v_i)\in Q}((H(m_i))^{\alpha/\beta})^{v_i \cdot \gamma} \cdot \prod_{j=1}^{s} T_j^{\mu_j} \cdot u_j^{-1},\ Y) \quad (3)$$

If it holds, *TPA* outputs *accept*, otherwise *reject*.

4.3 Support for Dynamic Data Operation

Merkle hash tree can perfectly work for the static case and also do well when the elements are inserted in a random order for the dynamic case. When it undergoes a sequence of inserting operations in a certain order, the structure of the binary tree may degenerate and the performance may become poor. In this case, the binary tree will need rebalancing continuously with the operations [4,17]. While the skip lists are balanced probabilistically, in dealing with a variety of dynamic operations, the performance of the skip list is relatively stable [26]. So, we choose the *rank-based authenticated skip list* [9] as the authenticated search data structure of the dynamic case [10]. Through this structure, various dynamic operations can be efficiently performed, the order of data blocks in the file can be guaranteed not to be changed, the integrity of $(H(m_i))^{\alpha/\beta}$ can be ensured and then the integrity of the signatures and the data blocks can be ensured.

Now we describe the dynamic data operations. Our scheme can fully handle block-level dynamic operations including modification $('M')$, insertion $('I')$ and deletion $('D')$ for the outsourced data. Each operation affects only nodes along a verification path in the skip list. We assume that the file F, the signatures of data blocks Φ and the corresponding skip list with the elements $(H(m_i))^{\rho}(1 \le i \le n, \rho = \alpha/\beta)$ have been stored in the cloud server. The user keeps the root as *verification metadata*, which is the label of the start node of the skip list.

Data modification: We assume that the user wants to modify the i-th data block m_i to $m_i^{'}$. Firstly, the user sends a query "*Prepareupdate* $= (i)$" to the server to get the message which includes $H(m_i)$ and the proof Π_i of block i. After receiving these information, the user computes $(H(m_i))^{\rho}$ and generates root S. Then the user checks $M_c \stackrel{?}{=} S$. If it is not true, output *reject*, otherwise the user computes the new block signature $\sigma_i \leftarrow ((H(m_i^{'}))^{\alpha/\beta} \cdot \prod_{j=1}^{s} u_j^{m_{ij}^{'}})^{\beta}$. Then, he constructs an update request message "*Update* $= ('M', i, m_i^{'}, \sigma_i^{'}, H(m_i^{'})^{\rho})$" and sends it to the server. Upon receiving the request, the server runs *PerformUpdate*$(F, \Phi, Update)$. Through the procedure the server completes the following tasks:

1) Replaces m_i and σ_i with $m_i^{'}$ *and* $\sigma_i^{'}$, respectively;
2) Replaces $(H(m_i))^{\rho}$ with $(H(m_i^{'}))^{\rho}$ of the leaf node i, then updates the labels of the affected nodes and generates the new root $M_c^{'}$.

Finally the server returns $M_c^{'}$ to the user. Then the user generates the new root $S^{'}$ using $\Pi_i, (H(m_i^{'}))^{\rho}$ and compares it with $M_c^{'}$ to check whether the server has performed the modification operation as required or not. If it is not true, output *reject*, otherwise output *accept*. Then, the user replaces M_c with $M_c^{'}$ as the new root metadata and deletes *Update* and $m_i^{'}$ from its local storage.

Data insertion: Data insertion means inserting a new block after some specified position in the file F. Suppose the user wants to insert a block $m_{i+1}^{'}$ after the i–th block m_i. Firstly, the user sends a query "*Prepareupdate* $= (i)$" to the server, then the server returns $H(m_i)$ and the proof Π_i of block i. Next, the user computes $(H(m_i))^{\rho}$ and generates root S using $\{\Pi_i, H(m_i)^{\rho}\}$. Then the user checks $M_c \stackrel{?}{=} S$. If it is not true, output *reject*, otherwise the user computes the new block signature $\sigma_{i+1}^{'} = ((H(m_{i+1}^{'}))^{\alpha/\beta} \prod_{j=1}^{s} u_j^{m_{i+1,j}^{'}})^{\beta}$ and determines the height of the tower of the skip list associated with the new block. Finally he constructs an update request message "*Update* $= ('I', l, i, m_{i+1}^{'}, \sigma^{'}, H(m_{i+1}^{'})^{\rho})$" and sends it to the server, where 'l' denotes the height of the tower related to the new node. Upon receiving the request, the server runs *PerformUpdate*$(F, \Phi, Update)$. The server completes the following tasks:

1) The server stores data block $m_{i+1}^{'}$ and its signature $\sigma_{i+1}^{'}$;
2) The server adds a leaf node after the position i of which the data item is $(H(m_{i+1}^{'}))^{\rho}$ according to the height l, then updates the labels, levels and ranks of the affected nodes and generates the new root $M_c^{'}$ based on the updated skip list.

The server sends to the user $M_c^{'}$ in response. Then the user generates the new root $S^{'}$ using $\{\Pi_i, (H(m_{i+1}^{'}))^{\rho}\}$ and compares it with $M_c^{'}$ to check whether the server has performed the insertion operation as required or not. If it is not true, output *reject*, otherwise output *accept*. Then, the user replaces M_c with $M_c^{'}$ as the new root *metadata* and deletes *Update* and m'_{i+1} from its local storage.

Data deletion: Data deletion refers to deleting a specified data block from the file. The corresponding element in the skip list will be deleted at the same time. Data deletion is the opposite operation of data insertion. However, the parameters specified by the user don't include the tower height. The details of the operation procedure are similar to that of data modification and insertion, so we omit them here.

4.4 Support for Batch Auditing

When the *TPA* simultaneously copes with different auditing delegations from different *D* users on different *D* files respectively, we can extend our scheme to implement batch auditing tasks. If the i in the Q is within the range of the number of blocks of the file, the auditing for the file can be added into the batch auditing. The batch auditing scheme can reduce the number of relatively expensive pairing operations from *2D* to *D+1*.

The k^{th} user randomly chooses parameters $u_{k,j} \in_R G, 1 \leq k \leq D, 1 \leq j \leq s$. His/her secret key and corresponding public key are denoted as $sk_k = (\alpha_k, \beta_k, ssk_k)$ and $pk_k = (X_k, Y_k, spk_k)$. The user's outsourced file is $F_k = \{m_{k,i,j}\}, (1 \leq k \leq D, 1 \leq i \leq n, 1 \leq j \leq s)$, the file name is fn_k and the tag of the file is $t_k = t_{k,0} \parallel SSig_{ssk_k}(t_{k,0})$. The signature of the block i is $\sigma_{k,i} = ((H(m_{k,i}))^{\alpha_k/\beta_k} \cdot \prod_{j=1}^{s} u_{k,j}^{m_{k,i,j}})^{\beta_k}$. The root of the corresponding skip list is $M_{k,c}$.

The *CSP* chooses $\lambda_{k,j} \in_R Z_p$, then computes $T_{k,j} = u_{k,j}^{\lambda_{k,j}}$ as the commitments for each user. The *TPA* chooses the challenge $Chal = \{c, k_1, k_2\}$ and sends *Chal* to *CSP*. After receiving *Chal*, the *CSP* gets $Q = \{(i, v_i)\}_{i \in I}$, chooses randomly $r_k \in_R Z_p$ and calculates $\psi_k = e(g^{r_k},\ h), \gamma_k = h_1(\psi_k)$ and

$$\begin{cases} \sigma_k \leftarrow \prod_{(i,v_i)\in Q} \sigma_{k,i}^{\gamma_k \cdot v_i} \cdot g^{r_k} \\ \mu_{k,j} \leftarrow \lambda_{k,j}^{-1} \cdot (\gamma_k \cdot \sum_{(i,v_i)\in Q} v_i \cdot m_{k,i,j} + 1) \end{cases} \quad (4)$$

The *CSP* sends $\theta_k = (\{\sigma_k\}_{1\leq k\leq D}, \{\mu_{k,j}\}_{1\leq k\leq D, 1\leq j\leq s}$ $, \{\psi_k\}_{1\leq k\leq D})$, the set $\{\Pi_{k,i}\}_{1\leq k\leq D, i\in I}$ of the proof for block $m_{k,i}$ and $\{(H(m_{k,i}))^{\alpha_k/\beta_k}\}_{1\leq k\leq D, i\in I}$ to the *TPA*.

After receiving the response from the *CSP*, the *TPA* calculates the root $R_{k,t}$ from $\{(H(m_{k,i}))^{\alpha_k/\beta_k}, \Pi_{k,i}\}_{1\leq k\leq D, i\in I}$ and checks $R_{k,t} \stackrel{?}{=} M_{k,c}$ for every file. If it is not true, *TPA* outputs *reject*, otherwise *TPA* can check

$$e(\prod_{k=1}^{D} \sigma_k,\ h) \stackrel{?}{=} \prod_{k=1}^{D} (\psi_k \cdot e(\prod_{(i,v_i)\in Q}((H(m_{k,i}))^{\alpha_k/\beta_k})^{v_i \cdot \gamma_k} \cdot \prod_{j=1}^{s} T_{k,j}^{\mu_{k,j}} \cdot u_{k,j}^{-1},\ Y_k)) \quad (5)$$

If it holds, *TPA* outputs *accept*, otherwise *reject*.

5 Evaluation

5.1 Security Evaluation

Completeness property: For each random challenge Q and its corresponding correct responses, the completeness of the protocol can be elaborated as follows:

$$e(\sigma, h) \stackrel{?}{=} \psi \cdot e(\prod_{(i,v_i)\in Q}((H(m_i))^{\alpha/\beta})^{v_i \cdot \gamma} \cdot \prod_{j=1}^{s} T_j^{\mu_j} \cdot u_j^{-1},\ Y)$$

The right side

$$= e(\prod_{(i,v_i)\in Q}(H(m_i))^{(\alpha/\beta)\cdot v_i \cdot \gamma} \cdot \prod_{j=1}^{s} u_j^{\gamma \cdot \sum_{(i,v_i)\in Q} v_i \cdot m_{ij} + 1} \cdot u_j^{-1},\ Y) \cdot \psi$$
$$= e(\prod_{(i,v_i)\in Q}(H(m_i))^{(\alpha/\beta)\cdot v_i \cdot \gamma} \cdot \prod_{j=1}^{s} u_j^{\gamma \cdot \sum_{(i,v_i)\in Q} v_i \cdot m_{ij}},\ Y) \cdot \psi$$
$$= e(\prod_{(i,v_i)\in Q}(H(m_i))^{(\alpha/\beta)\cdot v_i \cdot \gamma} \cdot \prod_{(i,v_i)\in Q}(\prod_{j=1}^{s} u_j^{m_{ij}})^{v_i \cdot \gamma},\ Y) \cdot \psi$$
$$= e(\prod_{(i,v_i)\in Q}((H(m_i))^{\alpha/\beta} \cdot \prod_{j=1}^{s} u_j^{m_{ij}})^{v_i \cdot \gamma \cdot \beta} \cdot g^r,\ h)$$

= The left side of the equation

So the equation means that the protocol is valid for the correct responses.

Soundness property: The soundness property means that a false response will not be accepted as the correct. In this context, it means that the *CSS* cannot generate a valid response to the *TPA's* challenge if the outsourced data is not stored well.

Theorem 1. *If the CSS passes the verification of the Audit protocol, it must indeed store the specified data intact.*

Following from the proof of *CPoR* [[27], Theorem 4.2], we give a proof of Theorem 1 in the random oracle model.

Proof. To prevent the *TPA* from extracting the value of σ_i from $\prod_{(i,v_i)\in Q} \sigma_i^{v_i \cdot \gamma}$, we blind it with g^r at each instance. To prove that the cloud server cannot falsify $\sigma, \{\mu_j\}_{1\leq j\leq s}$, we assume that the response information contains g^r instead of ψ and also contains $\lambda_j, (1 \leq j \leq s)$, corresponding to the commitment. □

There are a challenger and an adversary , and the latter is a malicious *CSP*. The challenger constructs a simulator *S* that will simulate the entire environment of the scheme for the adversary A. For any file F on which it previously made *St* query, the adversary A

can perform the *Audit* protocol with the challenger. In these executions of the protocol, the simulator S plays the part of the verifier and the adversary A plays the part of the prover: $S(pk, sk, t) \rightleftharpoons A$.

For some file F, if the adversary A can successfully forge the aggregate signature $\sigma^{'}$ with a non-negligible probability resulting in $\sigma^{'} \neq \prod_{(i,v_i)\in Q} \sigma_i^{v_i\cdot\gamma}\cdot g^r$ and successfully pass the verification, the simulator can make use of the adversary to solve the Computational Diffie-Hellman problem.

The simulator is given as input values $h, X = h^{\alpha}, Y = h^{\beta}$, and its goal is to output $h^{\alpha\cdot\beta}$.

Let $H : \{0,1\}^* \rightarrow G$ be a hash function which will be modeled as a random oracle. The simulator programs the random oracle H. When answering queries from the adversary, it chooses a random $\varphi \xleftarrow{R} Z_p$ and respond with $h^{\varphi} \in G$. When answering the queries of the form $H(m_i)$, the simulator programs it in a special way described below.

For each j, $1 \leq j \leq s$, the simulator chooses random values $\eta_j, \theta_j \xleftarrow{R} Z_p$ and sets $u_j \leftarrow X^{\eta_j}\cdot h^{\theta_j}$.

For each i, $1 \leq i \leq n$, the simulator chooses a random value $r_i \xleftarrow{R} Z_p$, and programs the random oracle at i as

$$H(m_i) = h^{r_i}/Y^{\sum_{j=1}^{s}\eta_j\cdot m_{ij}}.$$

Now the simulator computes:

$$\begin{aligned}
\sigma_i &= (H(m_i))^{\alpha}\cdot(\prod\nolimits_{j=1}^{s} u_j^{m_{ij}})^{\beta} \\
&= (h^{r_i}/(Y^{\sum_{j=1}^{s}\eta_j\cdot m_{ij}}))^{\alpha}\cdot(\prod\nolimits_{j=1}^{s}(X^{\eta_j}\cdot h^{\theta_j})^{m_{ij}})^{\beta} \\
&= (h^{r_i}/(Y^{\sum_{j=1}^{s}\eta_j\cdot m_{ij}}))^{\alpha} \\
&\quad \cdot(X^{\sum_{j=1}^{s}\eta_j\cdot m_{ij}}\cdot h^{\sum_{j=1}^{s}\theta_j\cdot m_{ij}})^{\beta} \\
&= h^{\alpha\cdot r_i}\cdot h^{\beta\cdot\sum_{j=1}^{s}\theta_j\cdot m_{ij}} \\
&= X^{r_i}\cdot Y^{\sum_{j=1}^{s}\theta_j\cdot m_{ij}} \qquad (6)
\end{aligned}$$

The challenger keeps a list of its responses to *St* queries made by the adversary. Now the challenger observes each instance of the *Audit* protocol with the adversary A. If in any of these instances the adversary is successful (*i.e.*, the verification equation holds), but the adversary's aggregate signature $\sigma^{'} \neq \prod_{(i,v_i)\in Q}\sigma_i^{v_i\cdot\gamma}\cdot g^r$, the challenger declares failure and aborts.

Suppose $Q = \{(i, v_i)\}_{i\in I}$ is the query that causes the challenger to abort, and the adversary's response to that query is $\mu_1^{'}, \cdots, \mu_s^{'}$ together with $\sigma^{'}$. Let the expected response be $\mu_1, \cdots, \mu_s$ and σ. By the correctness of the scheme, the expected response satisfies the verification equation, *i.e.*, that

$$\begin{aligned}
&e(\sigma,\ h)/\psi \\
=&e(\prod\nolimits_{(i,v_i)\in Q}((H(m_i))^{\alpha/\beta})^{v_i\cdot\gamma}\cdot\prod\nolimits_{j=1}^{s}T_j^{\mu_j}\cdot u_j^{-1},\ Y)
\end{aligned} \qquad (7)$$

Because the challenger aborts, we know that $\sigma \neq \sigma^{'}$ and that $\sigma^{'}$ passes the verication equation, *i.e.* that

$$\begin{aligned}
&e(\sigma^{'},\ h)/\psi \\
=&e(\prod\nolimits_{(i,v_i)\in Q}((H(m_i))^{\alpha/\beta})^{v_i\cdot\gamma}\cdot\prod\nolimits_{j=1}^{s}T_j^{\mu_j^{'}}\cdot u_j^{-1},\ Y)
\end{aligned} \qquad (8)$$

Observe that if $\mu_j^{'} = \mu_j$ for each j, we can get $\sigma^{'} = \sigma$, which contradicts our assumption above. Therefore, if we define $\Delta\mu_j \overset{\text{def}}{=} \mu_j^{'} - \mu_j$ for $1 \leq j \leq s$, it must be the case that at least one of $\{\Delta\mu_j\}$ is nonzero. Let $\sigma^* = \prod_{(i,v_i)\in Q}\sigma_i^{v_i}$ and $\mu_j^* = \sum_{(i,v_i)\in Q} v_i\cdot m_{ij}$. So, dividing the Equation (8) by the Equation (7) , we obtain

$$\begin{aligned}
&e((\sigma^{*'})^{\gamma}/(\sigma^*)^{\gamma},\ h) \\
=&e(\prod\nolimits_{j=1}^{s} u_j^{\gamma\cdot\Delta\mu_j^*},\ Y) \\
=&e(\prod\nolimits_{j=1}^{s}(X^{\eta_j}\cdot h^{\theta_j})^{\gamma\cdot\Delta\mu_j^*},\ Y) \\
=&e(\prod\nolimits_{j=1}^{s} X^{\gamma\cdot\eta_j\cdot\Delta\mu_j^*},\ Y)\cdot e(\prod\nolimits_{j=1}^{s} h^{\gamma\cdot\theta_j\cdot\Delta\mu_j^*},\ Y) \\
=&e(X^{(\sum_{j=1}^{s}\eta_j\cdot\Delta\mu_j^*)\cdot\gamma},\ Y)\cdot e(h^{(\sum_{j=1}^{s}\theta_j\cdot\Delta\mu_j^*)\cdot\gamma},\ Y)
\end{aligned} \qquad (9)$$

$$\begin{aligned}
&e((\sigma^{*'})^{\gamma}\cdot((\sigma^*)^{\gamma})^{-1}\cdot Y^{-\gamma\cdot(\sum_{j=1}^{s}\theta_j\cdot\Delta\mu_j^*)},\ h) \\
=&e((X^{\gamma\cdot(\sum_{j=1}^{s}\eta_j\cdot\Delta\mu_j^*)})^{\beta},\ h)
\end{aligned} \qquad (10)$$

So, if $\sum_{j=1}^{s}\eta_j\cdot\Delta\mu_j^* \neq 0$, we see that we have found the solution to the computational Diffie-Hellman problem:

$$h^{\alpha\cdot\beta} = ((\sigma^{*'})^{\gamma}\cdot(\sigma^{*\gamma})^{-1}\cdot Y^{-\gamma\cdot(\sum_{j=1}^{s}\theta_j\cdot\Delta\mu_j^*)})^{1/(\gamma\cdot\sum_{j=1}^{s}\eta_j\cdot\Delta\mu_j^*)}$$

Except the case that $\sum_{j=1}^{s}\eta_j\cdot\Delta\mu_j^*$ is equal to zero. However, we have already realized that not all of $\{\Delta\mu_j^*\}$ can be zero, and the values of $\{\eta_j\}$ are information that is theoretically hidden from the adversary, so $\sum_{j=1}^{s}\eta_j\cdot\Delta\mu_j = 0$ is only with the probability $1/p$, which is negligible.

As demonstrated before, we know $\sigma^{'} = \sigma$. Equating the verifications gives us

$$e(\sigma,\ h) = e(\sigma^{'},\ h),$$

from which using μ and $\mu^{'}$ we get that

$$\begin{aligned}
e(\prod\nolimits_{j=1}^{s} u_j^{\gamma\cdot\mu_j{}^*}),\ Y) &= e(\prod\nolimits_{j=1}^{s} u_j^{\gamma\cdot\mu_j^{*'}}, Y) \\
\prod_{j=1}^{s} u_j^{\gamma\cdot\Delta\mu_j{}^*} &= 1 \\
\prod\nolimits_{j=1}^{s}(X^{\eta_j}\cdot h^{\theta_j})^{\gamma\cdot\Delta\mu_j{}^*} &= 1 \\
X^{\sum_{j=1}^{s}\gamma\cdot\eta_j\cdot\Delta\mu_j{}^*}\cdot h^{\sum_{j=1}^{s}\gamma\cdot\theta_j\cdot\Delta\mu_j{}^*} &= 1 \\
X^{\sum_{j=1}^{s}\gamma\cdot\eta_j\cdot\Delta\mu_j{}^*} &= h^{-\sum_{j=1}^{s}\gamma\cdot\theta_j\cdot\Delta\mu_j{}^*}
\end{aligned}$$

So we find the solution to the discrete logarithm problem,

$$\alpha = -(\sum_{j=1}^{s} \gamma \cdot \theta_j \cdot \Delta\mu_j^*)/(\sum_{j=1}^{s} \gamma \cdot \eta_j \cdot \Delta\mu_j^*),$$

except the case that $\sum_{j=1}^{s} \eta_j \cdot \Delta\mu_j^*$ is equal to zero. While not all of $\{\Delta\mu_j^*\}$ can be zero, and the values of $\{\eta_j\}$ are information that is theoretically hidden from the adversary, so $\sum_{j=1}^{s} \eta_j \cdot \Delta\mu_j^* = 0$ is only with probability $1/p$, which is negligible. This completes the proof of the Theorem 1.

Privacy-Preserving Property: The privacy-preserving property means that TPA cannot extract users' data from the information gleaned during the auditing phase.

Theorem 2. *The TPA cannot extract users' data from the CSP's response* θ and $\{(H(m_i))^{\alpha/\beta}\}_{i\in I}$.

Proof. The m_i, α and β are all hidden from the *TPA*, so $H(m_i)$ cannot be determined from $(H(m_i))^{\alpha/\beta}$. Although $e(H(m_i), X)$ is equal to $e((H(m_i))^{\alpha/\beta}, Y)$, $H(m_i)$ cannot be calculated from it either. Because the isomorphism $f_Q : G \rightarrow G_T$ by $f_Q(P) = e(P, Q)$ is believed to be one-way function [6], when given $f_Q(P)$, it is infeasible to find its inverse. In addition, X can be removed from the pk in a concrete implementation. Therefore, it is hard to recover m_i from $(H(m_i))^{\alpha/\beta}$. Similarly, it is hard to extract σ_i from σ.

Every λ_j is randomly chosen by *CSP*, the λ_j^{-1} is the inverse element of it. Both of them are hidden from *TPA*. The $\sum_{i,v_i\in Q} v_i m_{ij}$ is blinded with λ_j^{-1}, so μ_j is uniformly distributed in Z_p for every response. Although *TPA* can obtain enough linear combinations of the data block m_i and its coefficient v_i, he must firstly obtain λ_j^{-1} if he wants to get μ_j^*. The λ_j can be calculated from $T_j = u_j^{\lambda_j}, (1 \leq j \leq s)$. But this means to solve the discrete logarithm problem (*DLP*). Due to the hardness assumption of *DLP*, *TPA* cannot get λ_j. So it is hard to obtain users' data from $\mu_j, (1 \leq j \leq s)$. This completes the proof of the Theorem 2. □

5.2 Performance Analysis

In order to elaborate the computation overhead of each entity, we specify some notations for the basic computational operations in the Table 1 [32].

We compared the two typical privacy-preserving data auditing schemes with that of ours in Table 2 for the computational cost of the user, *CSP*, and *TPA*, respectively. Here, n, s, and c are the number of data blocks, number of sectors and number of sampled data blocks, respectively. For the storage and communication overhead of our scheme, we present the following complexity analysis:

Table 1: Notations of relevant operations

Notation	Meaning
$Mult_G^x$	x multiplications in group G
$Mult_{Z_p}^x$	x multiplications in group Z_p
$Hash_{Z_p}^x$	x hash values into group Z_p
$Hash_G^x$	x hash values into group G
$Hash_{D_g}^x$	x times hash function $h_2(\cdot)$, generating message digest
$Add_{Z_p}^x$	x additions in group Z_p
Exp_G^x	x exponentiations g^t, for $g \in G, t \in Z_p$
$Exp_{G_T}^x$	x exponentiations g^t, for $g \in G_T, t \in Z_p$
$Pair_{G_T}^x$	x pairings $e(u, v)$, where $u, v \in G, e(u, v) \in G_T$
PRP_S^x	x pseudo–random permutations in $S = \{0, 1\}^{\log_2 n}$
$PRF_{Z_p}^x$	x pseudo–random functions in Z_p

1) The user storage complexity is $O(1)$ and the server storage complexity is $O(n)$.

2) The communication complexity of the challenge phase is $O(1)$ and that of the response phase is $O(\log n)$.

We compared the complexities of the storage and communication of the audit protocol of our scheme with that of two other privacy-preserving schemes in Table 3. The communication complexity in the phase of auditing is $O(\log n)$ in our scheme; however, we could save the maintenance of a table with $O(n)$ complexity of storage space on the user side.

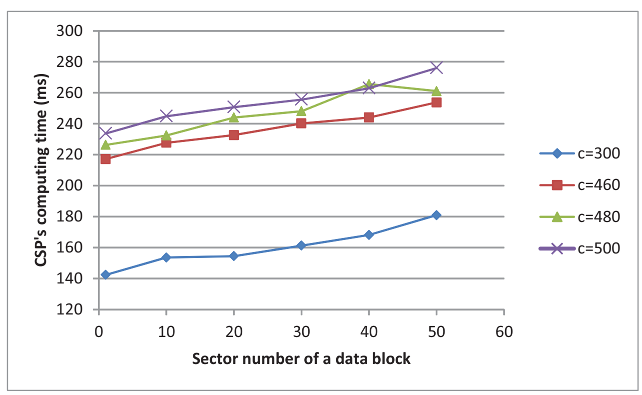

Figure 3: Comparison of computing time for CSP under different s and c

Based on the Pairing-Based Cryptography (PBC) library version 0.5.14, we implement our experiment using C language on an Ubuntu Linux system with an Intel Core i7-4790 CPU running at 3.60GHz with 8GB of RAM and a 7,200 RPM Seagate 1 TB drive. The elliptic curve we choose in the experiment is an MNT curve, with base field size of 159 bits and the embedding degree 6. The length

Table 2: Comparison of computation overhead of different privacy-preserving schemes

	The Computation overhead		
Scheme	**User**	**Server**	**Verifier**
[30]	$Exp_G^{n\cdot(s+2)} + Mult_G^{(n\cdot s)} + Hash_G^n$	$Pair_{G_T}^s + Exp_{G_T}^s + Exp_G^c + Mult_G^{c-1} + Mult_{Z_p}^{(c+1)\cdot s} + Add_{Z_p}^{c\cdot s} + Hash_{Z_p}^1$	$Pair_{G_T}^2 + Exp_G^{s+c+2} + Mult_G^{c+s-1} + Mult_{G_T}^s + Hash_G^c + Hash_{Z_p}^1$
[37]	$Exp_G^{2n+2+s} + Mult_G^n + Mult_{Z_p}^{n\cdot(s+1)} + Add_{Z_p}^{n\cdot(s-1)} + Hash_G^n$	$Pair_{G_T}^1 + Exp_G^{c+s+2} + Mult_G^{c+s-2} + Mult_{Z_p}^{(c+1)\cdot s} + Add_{Z_p}^{c\cdot s}$	$Pair_{G_T}^3 + Exp_G^{c+s} + Mult_G^{c+s-2} + Mult_{G_T}^2 + Hash_G^c$
Our scheme	$Exp_G^{3\cdot n+2} + Mult_G^n + Mult_{Z_p}^{n\cdot s} + Add_{Z_p}^{n\cdot(s-1)} + Hash_G^n$	$Pair_{G_T}^1 + Exp_G^{c+s+2} + Mult_G^c + Mult_{Z_p}^{c+2} + Add_{Z_p}^c + Hash_{Z_p}^1 + PRP_S^c + PRF_{Z_p}^c$	$Pair_{G_T}^2 + Exp_G^{c+s+2} + Mult_G^{c+s} + Hash_{Dg}^{c\cdot(\log n-1)} + PRP_S^c + PRF_{Z_p}^c$

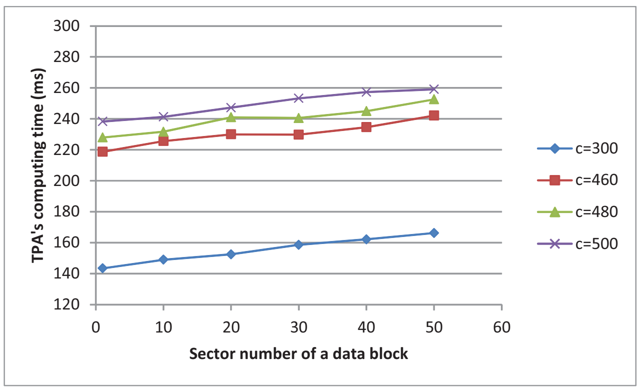

Figure 4: Comparison of computing time for TPA under different s and c

of p is 160 bits. Our test data is a randomly generated 100-MB file. All experimental results represent the mean of 30 trials.

Table 4 presents the experiment result of performance comparison between our scheme and that of [30] under different s and c. It shows that our scheme outperforms the other scheme except for the computing time of *CSP* in the case of $s = 1$. However, as the value of s increases, the *CSP* computing time of [30] will increase significantly because it needs to calculate s pairings during the auditing process. The communication overhead that the *CSP* sends to the *TPA* also increases significantly because the length of an element in group G_T is 120 bytes. Figure 3 and Figure 4 show computing time for *CSP* and *TPA* of our scheme under different s and c. With increase in s and c, the image curves change relatively smoothly and the distance between the curves is relatively uniform. This shows that our scheme is stable and there are no special expensive calculations related to s and c.

Table 3: Storage complexity and communication complexity of different privacy-preserving schemes

Scheme	**User storage complexity**	**Communication complexity of Audit protocol**
[30]	$O(n)$	$O(1)$
[37]	$O(n)$	$O(1)$
Our scheme	$O(1)$	$O(\log n)$

6 Conclusions

In this paper, we proposed a privacy-preserving public auditing scheme with supporting dynamics. The scheme uses the rank-based authenticated skip list as the authenticated search data structure. The formal proof demonstrates that our scheme is secure and has privacy-preserving property in the auditing phase, and performance analysis shows that our scheme is highly efficient.

Acknowledgments

This work is supported by the grants from National Natural Science Foundation of China (No. 61872038). The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] Cloud Security Alliance, *Top Threats to Cloud Computing*, 2010. (`https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf`)

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote data checking using provable data possession," *Acm Transactions on Information & System Security*, vol. 14, no. 1, p. 12, 2011.

Table 4: Comparison of computing time for CSP and TPA under different s and c

s=1	**Our scheme**			**[30]**		
Number of sampled blocks c	300	460	500	300	460	500
CSP computing time (ms)	142.28	217.10	233.69	142.37	203.78	227.81
TPA computing time (ms)	143.37	218.66	238.30	147.02	227.75	257.98
s=10	**Our scheme**			**[30]**		
Number of sampled blocks c	300	460	500	300	460	500
CSP computing time (ms)	153.50	227.66	244.85	168.68	245.38	258.29
TPA computing time (ms)	148.97	225.62	241.31	151.89	226.96	255.81

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *ACM Conference on Computer and Communications Security*, pp. 598–609, 2007.

[4] J. L. Baer and B. Schwab, "A comparison of tree-balancing algorithms," *Communications of the Acm*, vol. 20, no. 5, pp. 322–330, 1977.

[5] A. F. Barsoum and M. A. Hasan, "On verifying dynamic multiple data copies over cloud servers," *Iacr Cryptology Eprint Archive*, vol. 2011, 2011.

[6] D. Boneh and M. K. Franklin, "Identity based encryption from the weil pairing," *Crypto*, vol. 32, no. 3, pp. 213–229, 2001.

[7] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "Mr-pdp: Multiple-replica provable data possession," in *The International Conference on Distributed Computing Systems*, pp. 411–420, 2008.

[8] B. Dan, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

[9] C. C. Erway, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Transactions on Information and System Security (TISSEC'15)*, vol. 17, no. 4, p. 15, 2015.

[10] M. Etemad and A. Küpcü, "Transparent, distributed, and replicated dynamic provable data possession," in *International Conference on Applied Cryptography and Network Security*, pp. 1–18, 2013.

[11] M. T. Goodrich, R. Tamassia, and A. Schwerin, "Implementation of an authenticated dictionary with skip lists and commutative hashing," in *DARPA Information Survivability Conference & Exposition II*, pp. 68–82 vol.2, 2001.

[12] W. Hsien, 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.

[13] M. S. Hwang, C. C. Lee, T. H. Sun, "Data error locations reported by public auditing in cloud storage service," *Automated Software Engineering*, vol. 21, no. 3, pp. 373–390, Sep. 2014.

[14] 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.

[15] A. Juels and B. S. Kaliski, "Pors: Proofs of retrievability for large files," in *ACM Conference on Computer and Communications Security*, pp. 584–597, 2007.

[16] J. Katz and Y. Lindel, *Introduction to Modern Cryptography: Principles and Protocols (1 edition)*, British: Chapman and Hall/CRC, 2007.

[17] D. E. Knuth, *The Art of Computer Programming, Volume3: Sorting and Searching (2nd Edition)*, The US: Addison-Wesley Professional, 1998.

[18] Y. Li, Y. Yu, B. Yang, G. Min, and H. Wu, "Privacy preserving cloud data auditing with efficient key update," *Future Generation Computer Systems*, vol. 78, 2016.

[19] 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.

[20] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, and R. Kotagiri, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Transactions on Parallel & Distributed Systems*, vol. 25, no. 9, pp. 2234–2244, 2014.

[21] W. Mao, *Modern Cryptography: Theory and Practice*, The US: Prentice Hall PTR, 2003.

[22] T. Mell and P. Grance, "Draft nist working definition of cloud computing," *Referenced on*, vol. 53, no. 6, pp. 50–50, 2009.

[23] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Th Conference on Advances in Cryptology*, pp. 369–378, 1987.

[24] C. Papamanthou and R. Tamassia, "Time and space efficient algorithms for two-party authenticated data structures," *Information and Communications Security*, pp. 1-15, 2007.

[25] D. Pointcheval and J. Stern, "Provably secure blind signature schemes," *Proceedings of Asiacrypt*, vol. 1163, pp. 252–265, 1996.

[26] W. Pugh, "Skip lists : A probabilistic alternative to balanced tress," *Communications of the Acm*, vol. 33, no. 6, pp. 668–676, 1990.

[27] H. Shacham and B. Waters, "Compact proofs of retrievability," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 90–107, 2008.

[28] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in *Hotos'07: Workshop on Hot Topics in Operating Systems*, 2007. (`https://www.usenix.org/legacy/event/hotos07/tech/full_papers/shah/shah_html/hotos11web.html`)

[29] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, and R. Hao, "Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium," *Journal of Network & Computer Applications*, vol. 82, pp. 56–64, 2017.

[30] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[31] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel & Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[32] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," *Computers & Electrical Engineering*, vol. 40, no. 5, pp. 1703–1713, 2014.

[33] C. Xu, Y. Zhang, Y. Yu, X. Zhang, and J. Wen, "An efficient provable secure public auditing scheme for cloud storage," *Ksii Transactions on Internet & Information Systems*, vol. 8, no. 11, pp. 4226–4241, 2014.

[34] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *Journal of Systems & Software*, vol. 113, no. C, pp. 130–139, 2016.

[35] J. Yao, S. Chen, S. Nepal, D. Levy, and J. Zic, "Truststore: Making amazon s3 trustworthy with services composition.," in *Ieee/acm International Conference on Cluster, Cloud and Grid Computing*, pp. 600–605, 2010.

[36] Y. Zhu, G. J. Ahn, H. Hu, S. S. Yau, H. G. An, and S. Chen, "Dynamic audit services for outsourced storages in clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 227–238, 2013.

[37] Y. Zhu, H. Hu, G. J. Ahn, and S. S. Yau, "Efficient audit service outsourcing for data integrity in clouds," *Journal of Systems & Software*, vol. 85, no. 5, pp. 1083–1095, 2012.

[38] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, 2012.

Biography

Haichun Zhao received his Bachelor's and Master's degrees in Computer Science and Technology from School of Computer Science at Inner Mongolia Normal University, P. R. China. He is currently a Ph.D. candidate in School of Computer and Communication Engineering at University of Science and Technology Beijing. His current research interests are in information security and cloud computing.

Xuanxia Yao received her B.S. degree from Jiangsu University, M.S. and Ph.D. degree from University of Science and Technology Beijing (USTB), China. She is an associate professor in School of Computer and Communication Engineering, USTB. She is the author of one book, more than 30 articles. Her research interests include network and information security, trusted computing, the security issues in internet of things, cloud computing, blockchain and big data as well.

Xuefeng Zheng is Professor and Ph.D. Supervisor in School of Computer and Communication Engineering at University of Science and Technology Beijing, P.R. China. His research interests include the security of sensor net, cryptosystem and network security.