Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 3471312, 12 pages
https://doi.org/10.1155/2018/3471312

Research Article

Adjacency-Hash-Table Based Public Auditing for Data Integrity in Mobile Cloud Computing

Wenqi Chen,[1] Hui Tian,[1] Chin-Chen Chang,[2] Fulin Nan,[1] and Jing Lu[3]

[1] *College of Computer Science and Technology, National Huaqiao University, Xiamen 361021, China*
[2] *Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan*
[3] *Network Technology Center, National Huaqiao University, Xiamen 361021, China*

Correspondence should be addressed to Hui Tian; htian@hqu.edu.cn

Received 1 May 2018; Accepted 28 August 2018; Published 13 September 2018

Academic Editor: Ke Xiong

Cloud storage, one of the core services of cloud computing, provides an effective way to solve the problems of storage and management caused by high-speed data growth. Thus, a growing number of organizations and individuals tend to store their data in the cloud. However, due to the separation of data ownership and management, it is difficult for users to check the integrity of data in the traditional way. Therefore, many researchers focus on developing several protocols, which can remotely check the integrity of data in the cloud. In this paper, we propose a novel public auditing protocol based on the adjacency-hash table, where dynamic auditing and data updating are more efficient than those of the state of the arts. Moreover, with such an authentication structure, computation and communication costs can be reduced effectively. The security analysis and performance evaluation based on comprehensive experiments demonstrate that our protocol can achieve all the desired properties and outperform the state-of-the-art ones in computing overheads for updating and verification.

1. Introduction

Cloud storage is one of cloud computing services and provides a way to effectively store and manage big data [1]. In recent years, more and more individuals and businesses tend to outsource their data to the cloud, since outsourcing data can render the advantages of location independent resource pooling, flexible resources, universal network access, and usage-based pricing [2–6]. Although the benefits of cloud storage services are many and huge, it also faces a lot of challenges [2, 4]. For example, the security of data sharing and storage in the same group is an urgent issue to be solved in the cloud environment [7]. In addition, data deduplication in cloud storage is also one of the vital techniques to reduce the amount of storage space and save bandwidth [8, 9]. Particularly, due to the separation of data ownership and management, cloud users (data owners) cannot verify the integrity of their data in the traditional techniques, which leads to a trust gap between cloud users and the Cloud Service Provider (CSP). In addition, Cloud storage is also faced with many internal and external security threats [10–15] (e.g., byzantine failures, hacker attacks, etc.), which may lead to cloud data corruption or loss. To solve these concerns, the cloud data auditing whose purposes are to enhance the data security of cloud storage platforms and to improve mutual trust between users and the CSP is proposed.

The most core challenge of cloud data auditing is how to efficiently check the cloud data integrity. To address this problem, a provable data possession (PDP) protocol and a proof of retrievability (PoR) protocol have been provided, respectively, by work [16] and work [17]. In typical PDP protocols, the user first generates some metadata (such as block tags) for a data file to verify integrity of the data on cloud servers. Later, the user sends the file and metadata to the cloud servers and removes them from its local storage. PDP employs a challenge-response mode for the remote verification; i.e., the CSP can generate a proof for the verifier's challenge. Compared with the former, the latter (PoR) is a complementary protocol to PDP. In initial PoR protocols, the user first encodes the data file with error-correcting code before outsourcing data to the CSP. Therefore, the user can reconstruct the entire file from the CSP's partial response. The

PoR model focuses on static data. Compared with PoR, PDP is more suitable for dynamic data auditing; see [18–28].

The existing PDP protocols can be generally divided into two categories: private auditing and public auditing. In private auditing, the user is as an only verifier to remotely verify the data integrity with low overhead. Due to no trust between the user and the CSP, the user cannot provide convincing results for verification. What is more, it is not advisable for the user to conduct the audits for their data frequently, since one of the important motivations of outsourcing data is to reduce the user's burden of storage management. To address this problem, a public auditing protocol was first provided by Ateniese et al. [16], in which an independent authorized auditor (Third Party Auditor, TPA), not only the user, can remotely verify the data integrity. Therefore, the TPA can not only provide independent audit results, but also bear the communication overhead and computation costs in the entire verification phase. The public audit for cloud storage should also achieve some security and function requirements as follows:

(i) Privacy preserving: in public auditing, the TPA on behalf of the user periodically verifies the integrity of data on the cloud servers. Thus, auditing protocols should design a mechanism to ensure that the TPA cannot derive user's data contents from the collected information during the verification phase.

(ii) Batch auditing: batch auditing is defined as the TPA can deal with auditing tasks from multiple various users simultaneously, which not only reduces the numbers of communications between the TPA and the CSP during the auditing phase, but also enhances the verification efficiency.

(iii) Dynamic auditing: in the cloud storage environments, there are a lot of various application data (financial trade, social media, etc.), which need to be updated frequently. Therefore, dynamic data auditing is a significant function for cloud storage auditing.

For the dynamic data audit, Erway el at. [19] first provided an extended PDP protocol, named as dynamic provable data possession (DPDP), which introduced a dynamic authenticated data structure, rank-based authenticated skip list, to support data updating. Later, Wang el at. [20] proposed a protocol based on the BLS signature, which utilized Merkle Hash Tree (MHT) to achieve data updating. However, the above two protocols would cause heavy computational overhead of the TPA and large communication costs during the verification phase and the updating phase. Further, [23], [26], and [27], respectively, design the dynamic authenticated data structures, Index Hash Table (IHT), Dynamic Hash Table (DHT), and Doubly Linked Info Table (DLIT) to improve audit efficiency and the structures are stored in the TPA rather than the CSP, to reduce communication costs. Though the above protocols achieve auditing effectively, the methods still have some drawbacks. In [23], updating operations incur large computational overhead, especially insertion and delete operations. Thus, [26] and [27], respectively, design the structures to overcome the above drawbacks in [23]. However, search operations in [26, 27] are relatively inefficient in the verification phase and the updating phase.

In view of above problems, this paper introduces a novel dynamic data authenticated structure adjacency-hash table (AHT) in our public auditing protocol (AHT-PA). We employ the AHT to achieve dynamic auditing. Moreover, due to AHT stored in the TPA instead of CSP, its computational overhead and communication costs are significantly less than both the protocol based on the skip list [19] and the one using MHT [20]. In the verification phase and the updating phase, AHT-PA also outperforms the protocols [23, 26, 27]. We exploit the bilinear maps and Boneh-Lynn-Shacham (BLS) signatures to support batch auditing and employ random masking to achieve privacy preserving. Our contributions can be summarized as follows:

(1) We propose a novel public auditing protocol, which can simultaneously support the essential functions: privacy preserving, batch auditing, and dynamic data auditing.

(2) We introduce a novel dynamic structure, AHT, to save data properties for dynamic data auditing. With such structure, our protocol can effectively achieve the dynamic data auditing and the data updating.

(3) We prove the security of the presented protocol and justify the auditing performance by concrete experimental comparisons with the state of the arts. The results demonstrate that our protocol can efficiently achieve secure auditing and outperform the previous ones in computational overhead and communication costs.

The rest of the paper is organized as follows: in Section 2, we review the related work concerning cloud storage auditing, particularly, regarding the dynamic data auditing. Then, we introduce the background and the necessary preliminaries for our work in Section 3. Section 4 gives the detailed description of our protocol. Section 5 presents the security proofs of our protocol, and Section 6 gives the comprehensive performance evaluations through experimental comparisons with some existing protocols. Finally, Section 7 gives the concluding remark of this paper.

2. Related Work

In recent years, many researchers have focused on cloud storage auditing. In 2007, Atenises et al. [16] proposed one of the earliest related works, "provable data possession (PDP)", which employs the based-RSA homomorphic authenticator to check the data integrity. At the same year, Juels el at. [17] presented a complementary protocol, "Proof of Retrievability (PoR)", which can not only check the correctness of data on cloud, but also ensure the retrievability of cloud data with an encoding method (error-correcting code). However, due to encoding the file before outsourcing to the CSP, the PoR model focuses on static data, such as archive data. Compared with PoR, PDP is more suitable for dynamic data auditing. As mentioned earlier, the public auditing has some advantages over private auditing. In public auditing, TPA

Table 1: Function comparison of auditing protocols.

Protocols	Public auditing	Privacy protection	Dynamic auditing	Batch auditing
PDP[16]	√	×	×	×
PoR[17]	×	⊗	×	×
IHT-PA[23]	√	√	√	⊙
DAP[22]	√	√	√	√
DPDP(skip list)[19]	×	⊗	√	×
DPDP(MHT)[20]	√	√	√	√
DHT-PA[26]	√	√	√	√
DLIT-PA[27]	√	×	√	√
AHT-PA	√	√	√	√

Note: "√" means "support"; "×" means "not support"; "⊗" means "no demand"; and "⊙" means "not mentioned".

can not only provide independent audit results, but also bear the communication overhead and computation costs for the entire verification phase. Therefore, it is considered a more practical model [15, 26]. Besides, public auditing should also achieve some security and function requirements, for example, privacy preserving, batch auditing, and dynamic auditing.

To overcome the data leakage to the TPA, Wang et al. [28] first provided a public auditing protocol for privacy preserving, where the CSP integrates the aggregate value of the data blocks with random masking. Therefore, this protocol can guarantee that the TPA cannot learn any knowledge of the user data during the verification phase. Later, [22, 23, 26] show that privacy preserving is indispensable in public auditing. Moreover, [15] and [28] are extended to preform audit tasks from multiple users simultaneously for better performance. In work [15] and work [28], the approach for batch auditing is that the CSP aggregates the data block tags generated by various users and then the TPA uses them and related block information responded from the CSP to verify the integrity of the cloud data.

For the auditing dynamic data, Erway el at. [19] provided an extended PDP protocol, named dynamic provable data possession (DPDP), which first introduced a dynamic authenticated data structure, rank-based authenticated skip list, to support data updating. Later, Wang el at. [20] proposed a protocol based on the BLS signature, which utilized Merkle Hash Tree to achieve data updating. However, the above two protocols would cause heavy computational overhead of the TPA and large communication costs during the verification phase and the updating phase. Further, [23], [26], and [27], respectively, designed the dynamic authenticated data structures, Index Hash Table (IHT), Dynamic Hash Table (DHT), and Doubly Linked Info Table (DLIT) to improve audit efficiency and to reduce communication costs by storing the structures in the TPA instead of the CSP. Though the above protocols can effectively achieve public auditing, the methods still have some drawbacks. In [23], updating operations incur large computational overhead, especially insertion and delete operations. Thus, [26] and [27], respectively, design the structure to overcome the above drawbacks in [23]. However, search operation in [26, 27] is relatively inefficient in the verification phase and the updating phase. Therefore, this paper introduces a novel dynamic data authenticated structure, adjacency-hash table (AHT), to achieve better auditing and updating efficiency.

To highlight the difference between our protocol and the existing ones, Table 1 shows comparison results of functions among them. It is clear that the presented protocol (AHT-PA) supports all the mentioned audit functions.

3. Background and Preliminaries

3.1. Problem Statement. As illustrated in Figure 1, we concentrate on designing an AHT-based public audit protocol which includes the following three entities: Users have large amounts of data and outsource their data to the cloud. Cloud Service Provider (CSP) has large-scale computing and storage devices and provides users with cloud storage services. Third Party Auditor (TPA) undertakes audit tasks for users and provides fair and objective audit results. Users outsource their data to the cloud to enjoy the reliability of data storage and high-performance services and to reduce its maintenance overhead. However, since the CSP manages their data on the cloud rather than users, users strongly desire to periodically check the integrity and correctness of their data.

As mentioned in the existing protocols [18, 19, 23, 26], the TPA is pointed out to be credible but curious. In other words, although the TPA can credibly perform the audit in the verification phase, it may be curious about the privacy information of users' data and even may try to derive the users' data contents. In addition, the CSP is considered as an untrustworthy party. For gaining benefits or maintaining their reputations, the CSP may hide the fact of data loss and even delete some data that users rarely access. In particular, the CSP may further launch three attacks to the TPA:

(i) Forge attack: the CSP may attempt to forge the data blocks and their corresponding tags to pass the audit.

(ii) Replacing attack: the CSP may attempt to pass the audit by replacing a corrupted block and its tag with another block and its corresponding tag.

(iii) Reply attack: the CSP may attempt to pass the audit using the proof messages generated previously.

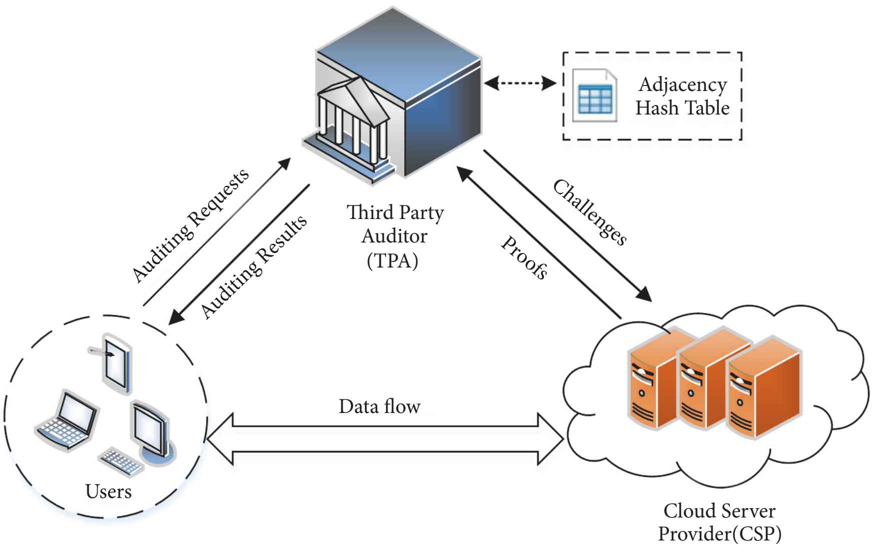

Figure 1: System architecture for public auditing.

To achieve the secure and efficient public auditing, our protocol aims to meet the following desired properties:

(1) Public auditing: it allows any authorized TPA to verify the correctness and integrity of user's data on the cloud servers.

(2) Blockless verification: it allows TPA to audit cloud data without retrieving the data blocks.

(3) Storage correctness: the CSP, who does not store the intact data as required, cannot pass the audit.

(4) Dynamic data audit: it allows the users to perform dynamic data operations (insertion, modification, and deletion) and achieves the efficient public auditing.

(5) Privacy preserving: it ensures that TPA cannot learn knowledge of users' data from the information collected during verification phase.

(6) Batch auditing: the TPA has the capability to deal with multiple auditing tasks from various users in a cost-effective way.

(7) Lightweight: it allows the TPA to perform public verification with minimum communication and computation costs.

3.2. Adjacency-Hash Table. As mentioned earlier, [19] and [20], respectively, introduced the authenticated skip list and the MHT to support public dynamic auditing. However, the above two protocols would cause heavy computational overhead of the TPA and large communication costs during the verification phase and the updating phase. Further, [26] and [27], respectively, designed Dynamic Hash Table (DHT) and Doubly Linked Info Table (DLIT) to improve audit efficiency and to reduce communication costs by storing the structures in the TPA rather than the CSP. Note that the DHT is a single linked table and the DLIT is a double linked table. Though the above protocols [26, 27] can achieve efficient auditing, the methods still have some drawbacks. Particularly, the search operation in [26, 27] is relatively inefficient in the verification phase and the updating phase. Therefore, this paper introduces a novel dynamic data authenticated structure, adjacency-hash table (AHT), to achieve better the auditing and updating efficiency.

The AHT is utilized by the TPA to store the latest version information (VI) of data blocks, as illustrated in Figure 2. The AHT is divided into file elements and the corresponding tables, called Adjacency Tables (AT). Every file element in file arrays includes the index number (NO_j), the file identifier (ID_j) of the given file (e.g., F_j), and a pointer indicating an AT. Each AT consists of block elements and a counter array whose every element contains a pointer indicating a block element and a value ($cValue_i$) which records the number of block elements after the corresponding pointer. Each file is organized by a file element and the corresponding AT. Each block element (e.g., the element corresponding the i-th block of the j-th file $m_{j,i}$) in the AT consists of the current version number of the block $v_{j,i}$, its time stamp $t_{j,i}$, and a pointer to the next node. Accordingly, the operations on the AHT are divided into file operations and block operations. To search the x-th ($1 \le x \le n$) block element, the TPA first determines the value a according to

$$\sum_{i=0}^{a} cValue_i < x \le \sum_{i=0}^{a+1} cValue_i, \qquad (1)$$

where i is the index of the counter array ($0 \le i \le n$) and the head element whose index is 0 in the counter array is used to indicate that $cValue_0$ should be set to 0. Note that the head is not drawn in Figure 2, because it is virtual. Further, the TPA calculates the distance dis, namely,

$$dis = x - \sum_{i=0}^{a} cValue_i. \qquad (2)$$

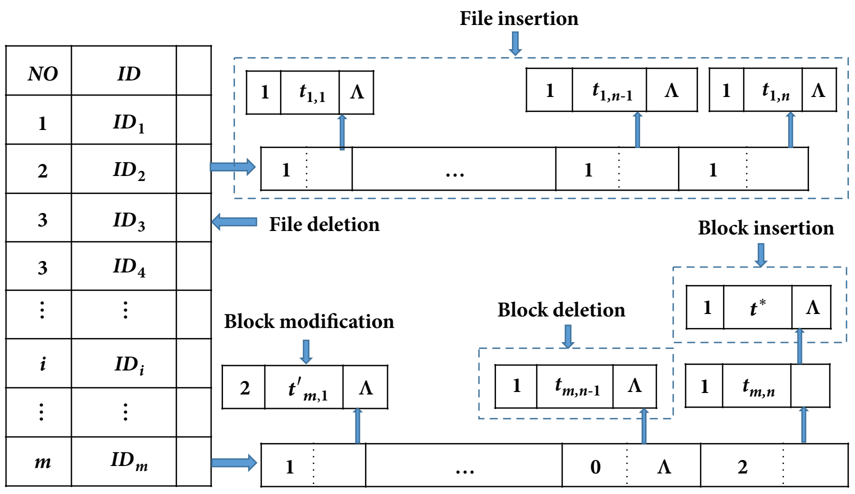

FIGURE 2: Adjacency-hash table (AHT).

Apparently, the x-th block element is the dis-th block element after the pointer of (a+1)-th element in the counter array. To insert a block element after an existing block, the TPA first tracks the given node (the given block element) and inserts the new node after it; the deletion of the given block element is to first track the given node and to delete it from the current AT. Besides, when the value ($cValue_i$) is equal to "0", the corresponding element in the counter array should be deleted. The search process of a file is to locate the file element according to its identifier or index; the insertion of the given file is to first insert a file element in the file array and then to construct a AT which includes related block elements; the deletion of the given file is to first remove the AT and to delete its file element; the modification of the given file is to update the file element and corresponding block elements.

3.3. Preliminaries. To facilitate understanding for readers, this section first introduces some necessary knowledge of cryptography for the presented protocol.

Bilinear Map. Let $\mathbb{G}$, $\mathbb{G}_T$ be multiplicative cyclic groups of a large prime order p. A map function e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ with the following properties: (1) Bilinear: $\forall u, v, h \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_p$, $e(u^a, u^b) = e(u, v)^{ab}$, and $e(u, v \cdot h) = e(u \cdot h, v) = e(u, v) \cdot e(u, h)$. (2) Computable: e is an efficient computable algorithm. (3) Nondegeneracy: if g is a generator of $\mathbb{G}$, then $e(g, g) \neq 1$.

BLS-Based Homomorphic Verifiable Authenticator (BLS-HVA). BLS-HVA is widely utilized for public auditing protocols [15, 18, 19, 22–24, 26–29], which can enable a public verifier to verify the cloud data integrity without downloading its original data. To be specific, BLS-HVAs are generated by BLS signatures in public auditing. Consequently, BLS-HVAs satisfy the properties as follows:

(i) Blockless verifiability [16]: constructing the proof in the BLS-HVA, the TPA can verify the cloud data integrity without its actual data content.

(ii) Homomorphism [16]: let $\mathbb{G}$ and $\mathbb{H}$ be multiplicative groups of a large prime order p, and "$\oplus$" and "$\otimes$" be operations in $\mathbb{G}$ and $\mathbb{H}$, respectively. If a map function f: $\mathbb{G} \longrightarrow \mathbb{H}$ satisfies homomorphism, then $\forall h_1, h_2 \in \mathbb{G}$, $f(h_1 \oplus h_2) = f(h_1) \otimes f(h_2)$.

(iii) Nonmalleability [30]: let σ_1 and σ_2 denote signatures on blocks m_1 and m_2, respectively, and β_1 and β_2 two random numbers in $\mathbb{Z}_p$. For the given block, $m' = \beta_1 m_1 + \beta_2 m_2$, a user, who does not know the private key sk, cannot generate the signature σ' of m' by combining σ_1 and σ_2.

3.4. Secure Assumptions. The security of the present protocol is based on the following assumptions.

Computational Diffe-Hellman (CDH) Assumption. Let $\mathbb{G}$ be multiplicative cyclic groups of a large prime order p. Given g^a and g^b, where g is a generator of $\mathbb{G}$, and $a, b \in \mathbb{Z}_p$, it is computationally intractable to compute g^{ab}. For any probabilistic polynomial-time adversary $\mathscr{A}$, the probability of solving the CDH problem is negligible, namely,

$$Pr\left(\mathscr{A}_{\mathrm{CDH}}\left(g, g^a, g^b \in \mathbb{G}\right) \longrightarrow g^{ab} \in \mathbb{G} : \ \forall a, b \underset{R}{\in} \mathbb{Z}_p\right) \leq \varepsilon. \tag{3}$$

Discrete Logarithm (DL) Assumption. Let $\mathbb{G}$ be multiplicative cyclic groups of a large prime order p. Given h (such as $h = g^a$, where g is a generator of $\mathbb{G}$, and $a \in \mathbb{Z}_p$), it is computationally intractable to compute a. For any probabilistic polynomial-time adversary $\mathscr{A}$, the probability of solving the DL problem is negligible, namely,

$$Pr\left(\mathscr{A}_{\mathrm{DL}}\left(g, h \in \mathbb{G}\right) \longrightarrow a \in \mathbb{Z}_p, \ \text{s.t. } h = g^a\right) \leq \varepsilon. \tag{4}$$

4. The Proposed Protocol Based on AHT

In this section, we will present the core of our protocol based on AHT, which consists of the dynamic verification protocol

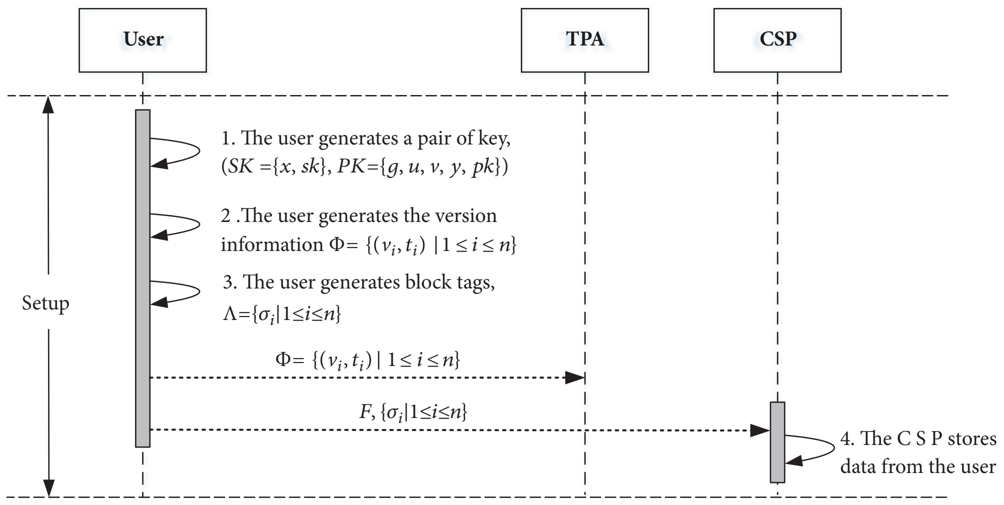
User
TPA
CSP
Setup
1. The user generates a pair of key, (SK ={x, sk}, PK={g, u, v, y, pk})
2 .The user generates the version information $\Phi = \{(v_i, t_i) \mid 1 \le i \le n\}$
3. The user generates block tags, $\Lambda=\{\sigma_i|1\le i\le n\}$
$\Phi = \{(v_i, t_i) \mid 1 \le i \le n\}$
$F, \{\sigma_i|1\le i\le n\}$
4. The C S P stores data from the user

Figure 3: The workflow of the setup phase.

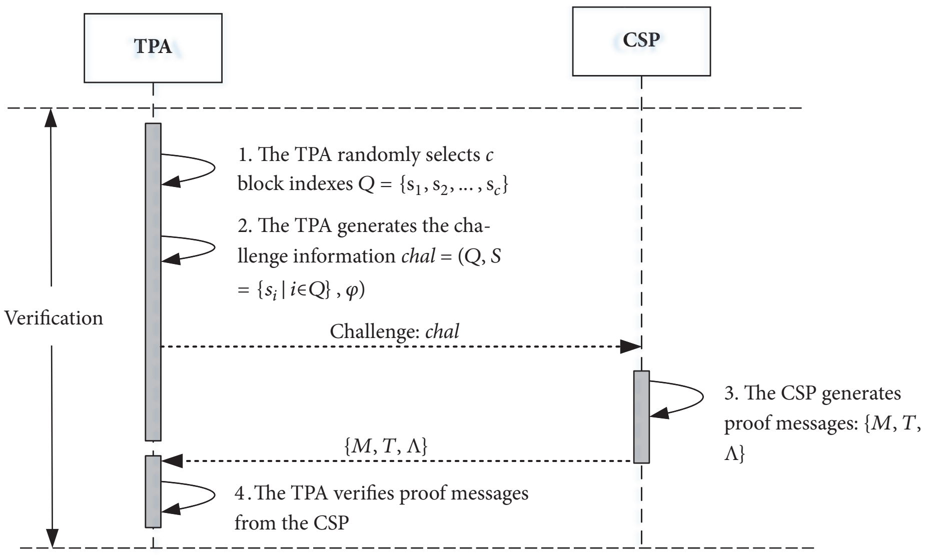
TPA
CSP
Verification
1. The TPA randomly selects c block indexes $Q = \{s_1, s_2, \dots, s_c\}$
2. The TPA generates the challenge information $chal = (Q, S = \{s_i \mid i\in Q\}, \varphi)$
Challenge: $chal$
3. The CSP generates proof messages: $\{M, T, \Lambda\}$
$\{M, T, \Lambda\}$
4.The TPA verifies proof messages from the CSP

Figure 4: The workflow of the verification phase.

with privacy protection described in Section 4.1, the updating operations detailed in Section 4.2, and the batch verification protocol in Section 4.3.

4.1. Dynamic Verification with Privacy Preserving. Let $\mathbb{G}$, $\mathbb{G}_T$ be multiplicative cyclic groups of a large prime order p, and g be the generator of $\mathbb{G}$. A map function e is defined as $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. $H(\cdot) : \{0,1\}^* \longrightarrow \mathbb{Z}_p$ is a secure hash function. The file outsourced to the cloud is denoted as F, which is divided into n blocks, namely, $F = \{m_1, m_2, \dots, m_n\}$. Our dynamic auditing protocol involves the following algorithms: *KeyGen* and *TagGen* in the setup phase (see Figure 3), and *Challenge*, *ProofGen*, and *Verify* in the verification phase (see Figure 4).

KeyGen (Key Generation). The user performs *KeyGen* to generate public and secret keys, ($PK = \{g, y, v, u, pk\}$, SK = $\{x, sk\}$), where (pk, sk) is a key pair for signature, v, $u \in \mathbb{G}$ are the random elements, $x \in \mathbb{Z}_p$ is a random number, and $y = g^x$.

TagGen (Tag Generation). For each block m_i ($i = 1, 2, \dots, n$), the user generates the signature σ_i:

$$\sigma_i = \left(u^{H(v_i \| t_i)} \cdot v^{m_i}\right)^x, \quad (5)$$

where v_i is the version number of m_i, and t_i is its time stamp, and '$\|$' is the connection operation. This signature, called block tag [16], should be uploaded to the cloud for verification along with the corresponding block. All version information (i.e., v_i and t_i) will be sent to the TPA for storing them in the

AHT. Moreover, the user generates a file tag ϑ to ensure the integrity of the file identifier ID:

$$\vartheta = ID \parallel SIG(sk, ID), \tag{6}$$

where $SIG(sk, ID)$ is the signature on ID with sk and sends it along with file identifier ID to the CSP.

Challenge. The TPA first retrieves the file tag ϑ and verifies the signature $SIG(sk, ID)$ with the public key pk. If the verification is failed, the TPA directly stops the verification by emitting FALSE; otherwise, it generates the following information: $chal = (Q = \{idx_i \mid 1 \le i \le c, c \le n\}, S = \{s_i \mid i \in Q\}, \varphi)$, where $Q = \{idx_i \mid 1 \le i \le c\}$ is the set of the block indexes to be verified, $S = \{s_i \mid i \in Q\}$ is the set of random numbers, and s_i is randomly selected form $\mathbb{Z}_p, \varphi = g^{\tau}$, and $\tau \in \mathbb{Z}_p$ is the random number. In particular, the TPA computes $\eta = y^{\tau}$ for the verification. Upon completion, the TPA sends the challenge information $chal$ to the CSP.

ProofGen (Proof Generation). While receiving the challenge, the CSP would produce a response proof for the verification, which consists of the tag proof, the block proof, and an auxiliary auditing factor. For the challenged block, the CSP generates the tag proof,

$$T = \prod_{i \in Q} e(\sigma_i, \varphi)^{s_i}, \tag{7}$$

and the block proof

$$M = \sum_{i \in Q} m_i \cdot s_i + r, \tag{8}$$

where $r \in \mathbb{Z}_p$, called random mask, is used for protecting the data privacy. Moreover, the CSP calculates the auxiliary auditing factor

$$\Lambda = e(v, y)^{-r}. \tag{9}$$

Upon completion, the CSP sends (T, M, Λ) back to the TPA as the response for the challenge.

Verify. To verify the response messages returned from the CSP, the TPA can perform the following equation:

$$\Lambda^{\tau} \cdot e\left(u^{\sum_{i \in Q} H(v_i \| t_i) \cdot s_i} \cdot v^{M}, \eta\right) = T. \tag{10}$$

If it holds, the algorithm outputs TRUE, otherwise, FALSE.

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

$$\begin{aligned} &\Lambda^{\tau} \cdot e\left(u^{\sum_{i \in Q} H(v_i \| t_i) \cdot s_i} \cdot v^{M}, \eta\right) \\ &\quad = e(v, \eta)^{-r} \cdot e\left(u^{\sum_{i \in Q} H(v_i \| t_i) \cdot s_i} \cdot v^{\sum_{i \in Q} m_i \cdot s_i + r}, \eta\right) \\ &\quad = e\left(u^{\sum_{i \in Q} H(v_i \| t_i) \cdot s_i} \cdot v^{\sum_{i \in Q} m_i \cdot s_i + r} \cdot v^{-r}, \eta\right) \\ &\quad = \prod_{i \in Q} e\left(\left(u^{H(v_i \| t_i)} \cdot v^{m_i}\right)^{x \cdot s_i}, \varphi\right) = \prod_{i \in Q} e(\sigma_i, \varphi)^{s_i} = T \end{aligned} \tag{11}$$

4.2. Dynamic Updating. To support the efficient updating operations for data blocks and files, we design the AHT in our protocol. The specific operations of block consist of block modification ($\mathscr{B}_{modify}$), block insertion ($\mathscr{B}_{insert}$), and block deletion ($\mathscr{B}_{delete}$) as follows.

Block Modification. Suppose the i-th block m_i of the file F will be modified to m'_i. The user first generates the corresponding version information (v'_i, t'_i) and then sends $U_{\mathrm{TPA}} = (F, \mathscr{B}_{modify}, i, v'_i, t'_i)$ to the TPA. Upon receipt, the TPA updates the AHT. Simultaneously, the user generates the new signature σ'_i for m'_i according to (5) and then sends $U_{\mathrm{CSP}} = (F, \mathscr{B}_{modify}, i, m'_i, \sigma'_i)$ to the CSP. Upon receiving, the CSP directly modifies m_i and σ_i as indicated.

Block Insertion. Suppose a new block m^* of the file F will be inserted after m_i. The user first generates the corresponding version information (v^*, t^*) and sends an insertion request $U_{\mathrm{TPA}} = (F, \mathscr{B}_{insert}, i, v^*, t^*)$ to the TPA. Upon receiving, the TPA performs the insertion request as indicated in the AHT. Meanwhile, the user generates the new signature σ^* for m^* according to (5) and then sends $U_{\mathrm{CSP}} = (F, \mathscr{B}_{modify}, i, m^*, \sigma^*)$ to the CSP. Once receiving the request, the CSP inserts the new block m^* after m_i and the new tag σ^* behind σ_i.

Block Deletion. Suppose the i-th block m_i of the file F will be deleted. The user sends a deletion request $U_{\mathrm{TPA}} = (F, \mathscr{B}_{delete}, i)$ to the TPA. Upon receipt, the TPA executes the deletion request to delete the corresponding version information in the AHT. Moreover, the user sends a deletion request $U_{\mathrm{CSP}} = (F, \mathscr{B}_{delete}, i)$ to the CSP. Upon receiving, the CSP directly deletes m_i and σ_i as indicated.

The updating operations on file include the file appending and the file deletion, which are very straightforward. We suppose that a new file F^* will be appended. The user needs to execute the algorithm *TagGen* once again. Moreover, while deleting a file F, the user first sends deletion instructions to the TPA and the CSP, respectively. Once receiving the requests, the TPA will delete the file element and its corresponding AT in the AHT, and the CSP will delete the file F and all of its tags.

4.3. Batch Verification. In reality, the TPA may simultaneously handle multiple audit tasks from different users' delegations. To achieve the minimum communication and computation costs, the batch auditing is introduced to deal with multiple auditing tasks from various users' delegations.

Suppose that the TPA sends w challenges for w users' delegations to the CSP. Once receipt, the CSP first calculates the tag proof (T_k), the data proof (M_k), and the auxiliary auditing factor (Λ_k) and then computes the aggregate tag proof T_{B} according to the following equations.

$$T_{\mathrm{B}} = \prod_{k=1}^{w} T_k. \tag{12}$$

Finally, the CSP responds with $(T_{\mathrm{B}}, \{M_k, \Lambda_k\}_{1 \le k \le w})$.

Table 2: Communication costs comparison.

Protocols	Verification phase	Updating phase
DPDP(skip list)[6]	$cO(\log n)$	$O(\log n)$
DPDP(MHT)[7]	$cO(\log n)$	$O(\log n)$
IHT-PA[10]	$O(c)$	$O(1)$
DHT-PA[13]	$O(c)$	$O(1)$
DLIT-PA[14]	$O(c)$	$O(1)$
AHT-PA	$O(c)$	$O(1)$

To verify the response messages, the TPA checks if the following equation holds:

$$\prod_{k=1}^{w}\left(\Lambda_k^{\tau_k}\cdot e\left(u_k^{\sum_{i\in Q_k}H(v_{k,i}\|t_{k,i})\cdot s_{k,i}}\cdot v_k^{M_k},\eta_k\right)\right)=T_{\mathrm{B}},\tag{13}$$

where $v_{k,i}$ and $t_{k,i}$ are the version number of m_i and its time stamp for the k-th user, u_k, and v_k are the public keys of the k-th user, and τ_k, η_k, $Q_k=\{idx_{k,i}\mid 1\le i\le c\}$ and $S_k=\{s_{k,i}\mid i\in Q_k\}$ belong to the challenge information for the k-th user. If (13) holds, the integrity of all the challenged files can be ensured. Otherwise, one or some of them are corrupted. The correctness of the above batch verification can be demonstrated as follows:

$$\begin{aligned}\prod_{k=1}^{w}\left(\Lambda_k^{\tau_k}\cdot e\left(u_k^{\sum_{i\in Q_k}H(v_{k,i}\|t_{k,i})\cdot s_{k,i}}\cdot v_k^{M_k},\eta_k\right)\right)&=\prod_{k=1}^{w}T_k\\&=T_{\mathrm{B}}\end{aligned}\tag{14}$$

5. Security Analysis

We will evaluate the security of the presented protocol with proofs of the following theorems.

Theorem 1 (unforgeability of BLS-HVA). *In our protocol, it is computationally infeasible for any adversary to forge a valid BLS-HVA if the computational Diffe-Hellman (CDH) assumption in bilinear groups holds.*

Proof. As demonstrated in the security analysis of [21], the BLS-HVA is effectively unforgeable when the CDH problem is hard in bilinear groups [31]. Thus, the proof is omitted here. □

Theorem 2 (unforgeability of proof). *The presented protocol can efficiently resist the forging attacks generated by the CSP. In other words, it is impossible for the CSP to forge effective proofs to pass the auditing verification.*

Proof. To respond for a challenge, the CSP sends a proof message (T, M, Λ) back to the TPA. If the auxiliary auditing factor Λ is fake, the verification equation (10) does not hold, even though the other proofs are valid. As demonstrated in Theorem 1, BLS-HVAs are unforgeable. Therefore, the tag proof T cannot be forged. Finally, we just need to prove that the block proof M is unforgeable.

To prove this, we first define the following game: The TPA sends a fake proof message $P^*=(T,M^*,\Lambda)$, where

$$M=\sum_{i\in Q}m_i\cdot s_i+r\neq M^*=\sum_{i\in Q}m_i^*\cdot s_i+r.\tag{15}$$

If the CSP can still pass the verification, then he/she wins this game; otherwise, he/she does not. Assume that the CSP wins this game, then

$$\begin{aligned}&\Lambda^{\tau}\cdot e\left(u^{\sum_{i\in Q}H(v_i\|t_i)\cdot s_i}\cdot v^{M^*},\eta\right)\\&\quad=\Lambda^{\tau}\cdot e\left(u^{\sum_{i\in Q}H(v_i\|t_i)\cdot s_i}\cdot v^{\sum_{i\in Q}m_i^*\cdot s_i+r},\eta\right).\end{aligned}\tag{16}$$

Moreover, for the valid proofs, we have

$$\begin{aligned}&\Lambda^{\tau}\cdot e\left(u^{\sum_{i\in Q}H(v_i\|t_i)\cdot s_i}\cdot v^{M},\eta\right)\\&\quad=\Lambda^{\tau}\cdot e\left(u^{\sum_{i\in Q}H(v_i\|t_i)\cdot s_i}\cdot v^{\sum_{i\in Q}m_i\cdot s_i+r},\eta\right).\end{aligned}\tag{17}$$

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

$$\sum_{i\in Q}m_i\cdot s_i+r=\sum_{i\in Q}m_i^*\cdot s_i+r,\tag{18}$$

which contradicts the above assumption. That is to say, the block proof is unforgeable. This accomplishes the proof of the theorem. □

The security of our protocol for resisting replacing and replay attacks is similar to the work [27]. Thus, we omit the corresponding proofs here.

6. Performance Evaluation

In this section, we will evaluate the performance of our protocol (AHT-PA) and compare it with the state of the arts.

6.1. Communication Costs. In this section, the communication costs of AHT-PA protocol are analyzed and compared during the verification phase (i.e., challenge and response) and updating phase. In the verification phase, the challenge and response messages between the TPA and the CSP bring communication overhead of $O(c)$, where c is denoted as the number of challenged blocks. Moreover, during the updating phase, the user should send an updating request to the CSP and the TPA, respectively, which costs $O(1)$.

Table 2 presents the communication costs of some protocols during the verification phase and updating phase, where

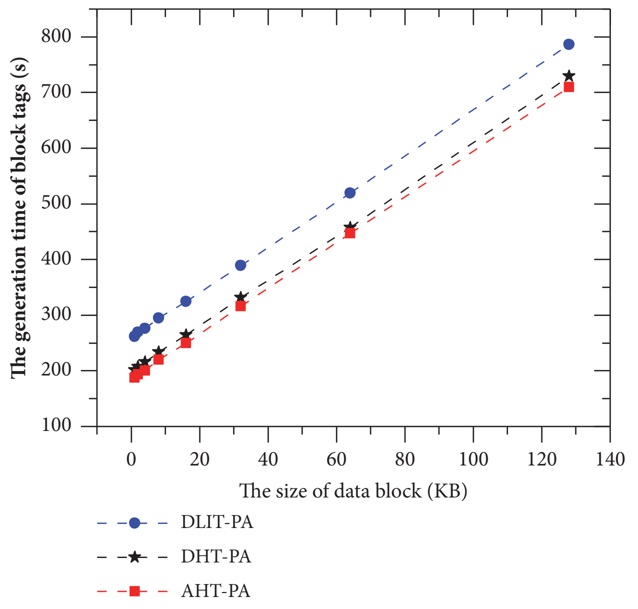

FIGURE 5: The tag generation time for different block sizes (the number of data blocks = 50000).

n represents the number of data blocks in a given file and *c* is the number of challenged blocks. Obviously, our protocol only requires a small amount of communication overhead and is substantially much efficient than the other protocols.

6.2. Computational Costs. The computational costs of AHT-PA protocol are evaluated and presented in this section. Thus, we implement all algorithms in the AHT-PA based on Pairing Base Cryptography (PBC) Library (0.5.14). The algorithms in experiments are evaluated on a DELL workstation with an Intel Xeon E3-1225v5 3.30 GHz, 16 GB DDR4-2133 ECC (2x8GB) RAM, and 2TB 7200 RPM SATA 1st HDD. We run the programs under a Linux (ubuntu 16.04.2 LTS x64) operating system, whose kernel version is 4.8.0 and use a MNT d159 curve, which has a 160-bit group order. The final results are the averages of 20 runs.

Computational Costs for Generating Tags. Figures 5 and 6 indicate the comparison results of the time of tag generation for different block sizes and for different numbers of data blocks, respectively, from which we can learn that (1) the generation time for the user is proportional to the block size or block number; (2) to deal with the same block size or same block number in the above two scenarios, AHT-PA takes less time than DHT-PA and DLIT-PA. In other words, the computation overhead of tag generation in AHT-PA is less than those in DHT-PA and DLIT-PA.

Computational Costs for Verification. Figure 7 indicates the experimental results of the verification time for different numbers of challenged blocks, from which we can learn that the verification time increases rapidly with the number of challenge blocks in the DHT-PA and DLIT-PA, but the verification time of AHT-PA remains stable and is much less than the previous two protocols.

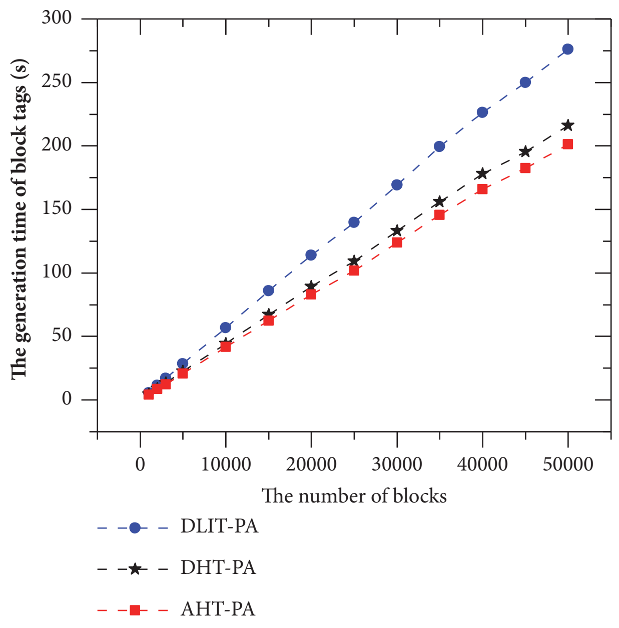

FIGURE 6: The time of tag generation for different numbers of blocks (block size = 4KB).

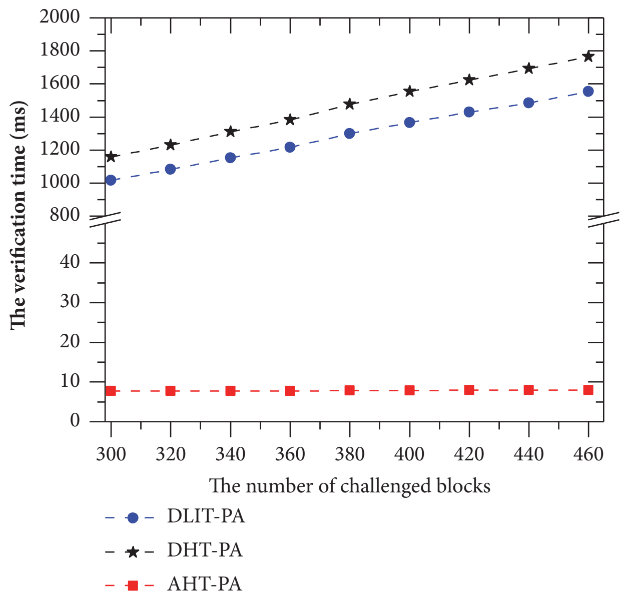

FIGURE 7: The verification time for different numbers of challenged blocks (the number of total data blocks = 50000, the block size = 4KB).

Computational Costs for Batch Auditing. In the batch auditing scenario, we will evaluate the performance of AHT-PA and compare it with DHT-PA and DLIT-PA. The comparison results, as shown in Figure 8, demonstrate that (1) three protocols can simultaneously handle various audits from multiple users and (2) at the same number of auditing tasks, the average audit time per task in AHT-PA is significantly less than in DHT-PA and DLIT-PA. That is to say, the batch auditing protocol in AHT-PA is much more efficient than those in DHT-PA and DLIT-PA.

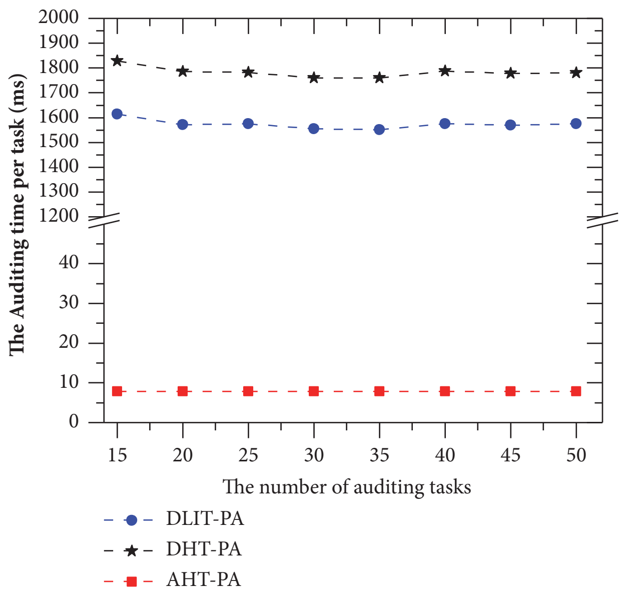

Figure 8: The average audit time per task for the various numbers of auditing tasks (the number of challenged blocks = 460, the total number of data blocks for each file = 50000, and the block size = 4KB).

Search Efficiency. We design two experiments on a single file to evaluate block-search efficiency of AHT-PA. The first experiment is performed under various total numbers of data blocks from 2×10^4 to 2×10^5 with 5000 challenged blocks, and another is performed under various numbers of challenged blocks with 2×10^5 data blocks. In the two experiments, we add an extra comparison item for DLIT-PA, named DLIT-PA (opt), whose results were obtained by first sorting the index set of challenged blocks to get its ascending set and then counting the corresponding frequency for searching block elements. As is well known, to locate a block element in the single linked list or double linked list, it is necessary to visit the whole list from the first element in the first search round. After that, if we previously sort the index set of the required blocks, the searching element in the single linked list or double linked list can start from the current element to the next element behind the current element. In this sense, in terms of search frequency, DLIT-PA (opt) is more efficient than DLTI-PA and identical to DHT-PA.

The results of the experiment under 5000 challenged blocks are shown in Figure 9, from which we can learn that AHT-PA outperforms the other protocols. Moreover, the larger the number of data blocks, the greater the search frequency gap. Figure 10 gives the experimental results under 2×10^5 data blocks. Apparently, the search frequency in AHT-PA slowly rises with the increasing number of challenged blocks. However, for the same number of challenged blocks, the search frequency in AHT-PA is much less than the ones in other protocols.

In summary, the public auditing protocol proposed in this paper can achieve better performance. To be specific, in the tag generation phase and verification phase, the computation cost is less than those of the state of the arts, while achieving better block-search efficiency in the verification phase and the updating phase.

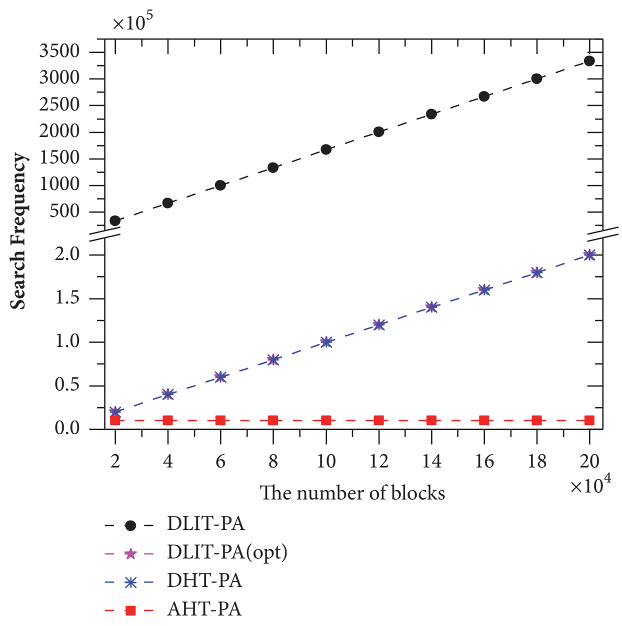

Figure 9: The search frequency under 5000 challenged blocks.

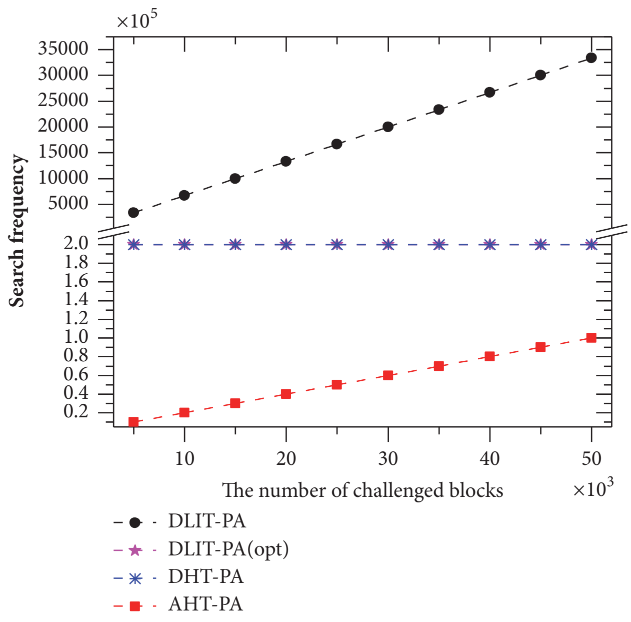

Figure 10: The search frequency under 2×10^5 data blocks.

7. Conclusions

In this paper, we present a novel auditing protocol for cloud storage. Differing from the state of the arts, we design a new structure, called adjacency-hash table, to support efficient data updating as well as reduce computational costs. Moreover, to achieve privacy preserving, our protocol employs the random masking technique to prevent the TPA from learning the users' data contents in the verification phase. Sufficient formal proofs indicate that our protocol is secure. The theoretical analysis and the experimental results show that our protocol is feasible and efficient and outperforms the state of the arts in both the computation overhead and the communication costs.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. U1405254 and U1536115, the Natural Science Foundation of Fujian Province of China under Grant No. 2018J01093, the Program for New Century Excellent Talents in Fujian Province University under Grant No. MJK2016-23, the Program for Outstanding Youth Scientific and Technological Talents in Fujian Province University under Grant No. MJK2015-54, the Promotion Program for Young and Middle-Aged Teachers in Science and Technology Research of Huaqiao University under Grant No. ZQN-PY115, the Opening Project of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security under Grant No. AGK201710, Subsidized Project for Postgraduates' Innovative Fund in Scientific Research of Huaqiao University, and Program for Science & Technology Innovation Teams and Leading Talents of Huaqiao University under Grant No. 2014KJTD13.

References

[1] H. Wang, Z. Zheng, L. Wu, and P. Li, "New directly revocable attribute-based encryption scheme and its application in cloud storage environment," *Cluster Computing*, vol. 20, no. 3, pp. 2385–2392, 2017.

[2] M. N. O. Sadiku, S. M. Musa, and O. D. Momoh, "Cloud computing: Opportunities and challenges," *IEEE Potentials*, vol. 33, no. 1, pp. 34–36, 2014.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[4] K. Yang and X. Jia, "Data storage auditing service in cloud computing: challenges, methods and opportunities," *World Wide Web*, vol. 15, no. 4, pp. 409–428, 2012.

[5] Z. Xia, X. Wang, X. Sun, Q. Liu, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[6] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Transactions on Communications*, vol. E98B, no. 1, pp. 190–200, 2015.

[7] J. Shen, T. Zhou, X. Chen, J. Li, and W. Susilo, "Anonymous and Traceable Group Data Sharing in Cloud Computing," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 4, pp. 912–925, 2018.

[8] J. Li, Y. K. Li, X. Chen, P. P. C. Lee, and W. Lou, "A Hybrid Cloud Approach for Secure Authorized Deduplication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1206–1216, 2015.

[9] J. Li, X. Chen, X. Huang et al., "Secure distributed deduplication systems with improved reliability," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3569–3579, 2015.

[10] Q. Jiang, J. Ma, and F. Wei, "On the security of a privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE Systems Journal*, 2016.

[11] D. A. B. Fernandes, L. F. B. Soares, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, "Security issues in cloud environments: A survey," *International Journal of Information Security*, vol. 13, no. 2, pp. 113–170, 2014.

[12] L. F. B. Soares, D. A. B. Fernandes, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, "Cloud security: State of the art," *Security, Privacy and Trust in Cloud Systems*, vol. 9783642385865, pp. 3–44, 2014.

[13] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward Efficient Multi-Keyword Fuzzy Search over Encrypted Outsourced Data with Accuracy Improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

[14] Q. Jiang, M. K. Khan, X. Lu, J. Ma, and D. He, "A privacy preserving three-factor authentication protocol for e-health clouds," *Journal of Supercomputing*, vol. 72, no. 10, pp. 3826–3849, 2016.

[15] 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.

[16] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 598–609, Virginia, Va, USA, November 2007.

[17] A. Juels and B. S. Kaliski Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 584–597, ACM, Alexandria, VA, USA, November 2007.

[18] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 220–232, 2012.

[19] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*, pp. 213–222, ACM, Chicago, Ill, USA, November 2009.

[20] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 5789, pp. 355–370, 2009.

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

[22] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 9, pp. 1717–1726, 2012.

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

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

[25] H. Jin, H. Jiang, and K. Zhou, "Dynamic and Public Auditing with Fair Arbitration for Cloud Data," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 680–693, 2018.

[26] H. Tian, Y. Chen, C.-C. Chang et al., "Dynamic-Hash-Table Based Public Auditing for Secure Cloud Storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 701–714, 2017.

[27] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2402–2415, 2017.

[28] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the IEEE INFO-COM*, pp. 525–533, March 2010.

[29] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Advances in Cryptology—ASIACRYPT 2008: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 2008*, vol. 5350 of *Lecture Notes in Computer Science*, pp. 90–107, Springer, Berlin, Germany, 2008.

[30] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Transactions on Services Computing*, vol. 8, no. 1, pp. 92–106, 2015.

[31] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology—ASIACRYPT 2001*, vol. 2248 of *Lecture Notes in Computer Science*, pp. 514–532, Springer, Berlin, Germany, 2001.