Dynamic-Hash-Table Based Public Auditing for Secure Cloud Storage

Hui Tian, *Member, IEEE*, Yuxiang Chen, Chin-Chen Chang, *Fellow, IEEE*,
Hong Jiang, *Fellow, IEEE*, Yongfeng Huang, *Senior Member, IEEE*,
Yonghong Chen, *Member, IEEE*, and Jin Liu, *Member, IEEE*

Abstract—Cloud storage is an increasingly popular application of cloud computing, which can provide on-demand outsourcing data services for both organizations and individuals. However, users may not fully trust the cloud service providers (CSPs) in that it is difficult to determine whether the CSPs meet their legal expectations for data security. Therefore, it is critical to develop efficient auditing techniques to strengthen data owners' trust and confidence in cloud storage. In this paper, we present a novel public auditing scheme for secure cloud storage based on dynamic hash table (DHT), which is a new two-dimensional data structure located at a third parity auditor (TPA) to record the data property information for dynamic auditing. Differing from the existing works, the proposed scheme migrates the authorized information from the CSP to the TPA, and thereby significantly reduces the computational cost and communication overhead. Meanwhile, exploiting the structural advantages of the DHT, our scheme can also achieve higher updating efficiency than the state-of-the-art schemes. In addition, we extend our scheme to support privacy preservation by combining the homomorphic authenticator based on the public key with the random masking generated by the TPA, and achieve batch auditing by employing the aggregate BLS signature technique. We formally prove the security of the proposed scheme, and evaluate the auditing performance by detailed experiments and comparisons with the existing ones. The results demonstrate that the proposed scheme can effectively achieve secure auditing for cloud storage, and outperforms the previous schemes in computation complexity, storage costs and communication overhead.

Index Terms—Cloud storage, cloud security, public auditing, dynamic hash table

1 Introduction

Cloud storage is an important branch of cloud computing [1], whose goal is to provide powerful and on-demand out-sourcing data services for users exploiting highly virtualized infrastructures [1], [2]. Due to the low-cost and high-performance of cloud storage, a growing number of organizations and individuals are tending to outsource their data storage to professional cloud services providers (CSP), which buoys the rapid development of cloud storage and its relative techniques in recent years. However, as a new cutting-edge technology, cloud storage still faces many security challenges [3]. One of the biggest concerns is how to determine whether a cloud storage system and its provider meet the legal expectations of customers for data security [4]. This is mainly caused by the following reasons. First, cloud users (data owners), who outsource their data in clouds, can no longer verify the integrity of their data via traditional techniques that are often employed in local storage scenarios. Second, CSPs, which suffer Byzantine failures occasionally, may choose to conceal the data errors from the data owners for their own self-interest [5]. What is more severe, CSPs might neglect to keep or even deliberately delete rarely accessed data that belong to ordinary customers to save storage space [6]. Therefore, it is critical and significant to develop efficient auditing techniques to strengthen data owners' trust and confidence in cloud storage, of which the core is how to effectively check data integrity remotely.

So far, many solutions have been presented to overcome this problem, which can be generally divided into two categories: private auditing and public auditing. Private auditing is the initial model for remote checking of data integrity [7], [8], in which the verification operation is performed directly between data owners and CSPs with relatively low cost. However, it cannot provide convincing auditing results, since the owners and CSPs often mistrust each other. Moreover, it is not advisable for the users to carry out the audit frequently, since it would substantially increase the overhead that the users may not afford. Thus, Ateniese et al. [9] first presented the public auditing scheme, in which the checking work is customarily done by an authorized third party auditor (TPA). Compared with the former, the latter can offer dependable auditing results and significantly reduce users' unnecessary burden by introducing an independent TPA. Thus, it is more rational and practical, and popularly believed to be the right direction of future development [2],

- *H. Tian, Y. Chen, Y. Chen, and J. Liu are with the College of Computer Science and Technology, National Huaqiao University, Xiamen 361021, China. E-mail: {htian, yxchen, jliu}@hqu.edu.cn, y.h.chen@hqu.edu. cn.*
- *C.-C. Chang is with the Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan. E-mail: alan3c@gmail.com.*
- *H. Jiang is with the Department of Computer Science and Engineering, University of Texas at Arlington, TX 76010. E-mail: hong.jiang@uta.edu.*
- *Y. Huang is the with Department of Electrical Engineering, Tsinghua University, Beijing 100084, China. E-mail: yfhuang@tsinghua.edu.cn.*

Manuscript received 25 June 2015; revised 20 Oct. 2015; accepted 21 Dec. 2015. Date of publication 25 Dec. 2015; date of current version 6 Oct. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2015.2512589

TABLE 1
Performance Comparison of Auditing Schemes for Cloud Storage

Schemes	Communication Overhead	Computation costs				Detection Probability
		Verification		Updating		
		Auditor	CSP	User/TPA	CSP	
PoRs [8]	$O(1)$	$O(c)$	$O(c)$	—	—	$1-(1-t)^{c}$
PDP [9]	$O(1)$	$O(c)$	$O(c)$	—	—	$1-(1-t)^{c}$
CPDP [13]	$O(c+s)$	$O(c+s)$	$O(c+s)$	—	—	$1-(1-t)^{c\cdot s}$
DAP [14]	$O(c)$	$O(c)$	$O(c\cdot s)$	$O(n)$	$O(w)$	$1-(1-t)^{c\cdot s}$
DPDP(skip list) [15]	$cO(logn)$	$cO(logn)$	$cO(logn)$	$wO(logn)$	$wO(logn)$	$1-(1-t)^{c}$
DPDP(MHT) [6]	$cO(logn)$	$cO(logn)$	$cO(logn)$	$wO(logn)$	$wO(logn)$	$1-(1-t)^{c}$
IHT-PA [16]	$O(c+s)$	$O(c+s)$	$O(c+s)$	$O(n)$	$O(w)$	$1-(1-t)^{c\cdot s}$
DHT-PA	$O(c)$	$O(c)$	$O(c)$ $(O(c\cdot s))$	$O(w)$	$O(w)$	$1-(1-t)^{c}(1-(1-t)^{c\cdot s})$

Note: *n is the whole number of blocks in a file; a block is divided into* s *segments;* c *is the number of the verified blocks when auditing a file;* w *is the number of updated blocks;* t *is the probability of the corrupted blocks/segments. In our DHT-PA, the communication overhead of each challenge is proportional to the number of the sampled blocks* c*, and the proof generated by CSP is a constant value, so the communication overhead can be considered as* O(c)*. In the verification phase, the costs for the proof generation in the CSP and the proof audit in the TPA are also proportional to the number of the sampled blocks* c*, so both the verification overheads for the CSP and that for the TPA are* O(c)*. If the segment strategy used to reduce the storage cost of block tags in the CSP is introduced, the verification overhead for the CSP can be considered as* O(c·s)*. However, the verification overhead for the TPA is still* O(c)*, because the segment strategy is transparent to the auditor. In the updating phrase, the computation overheads for data updating in the CSP and DHT updating in the TPA are proportional to the number of the blocks needed to modify, so they are both considered as* O(w)*. The probability of at least one of the uncorrected blocks (or segments) being picked by checking randomly sampled* c *blocks (or* c·s *segments) is* $1-(1-t)^{c}$ *(or* $1-(1-t)^{c\cdot s}$*).*

[3], [4], [5], [6], [9], [10], [11], [12], [13], [14], [15], [16]. In the public auditing, however, some vital problems as follows remain to be addressed or further gone into [5], [10].

- Privacy-preserving: data privacy protection (DPP) has always been an important topic for cloud storage. In the public auditing, the core of this problem is how to preserve uses' privacy while introducing a TPA. Although exploiting data encryption prior to outsourcing is an approach to mitigate the privacy concern in cloud storage [7], it cannot prevent data leakage during the verification process [11]. Thus, it is important for the cloud auditing to include a privacy-preserving mechanism independent to data encryption [11], [12].
- Batch auditing: to enhance the efficiency and enable the scalability of public auditing, the TPA should deal with multiple auditing tasks from various users in a fast and cost-efficient manner, i.e., support the batching auditing [13], [14].
- Dynamic auditing: as it is well known that a cloud storage system is not just a data warehouse, the users often need to update the data dynamically motivated by various application requirements. Therefore, it is significant for cloud storage auditing to support data dynamics [6], [15], [16].

For the dynamic data auditing, Erway et al. [15] first presented a dynamic provable data possession (DPDP) scheme, which extends the original PDP model [9] by introducing a rank-based authenticated skip list. Although their scheme cannot support public auditing, they reveal a general approach to achieve dynamic auditing, i.e., incorporating dynamic authenticated data structures with verification algorithms. Later, Wang et al. [6] presented a public auditing scheme based on Merkle Hash Tree (MHT), which can achieve the above auditing requirements. However, both the above schemes would incur heavy computational costs of the TPA and large communication overhead during the updating and verification processes. Further, Zhu et al. [16] proposed another public auditing scheme (IHT-PA) based on an index-hash table (IHT), which can effectively reduce both the computational costs and communication overhead by storing the data properties for auditing using the IHT in the TPA instead of the CSP. The IHT is the key of IHT-PA to support data dynamics, but it is inefficient in updating operations, especially insertion and deletion operations. The main reason is that these updating operations will lead to the adjustment of average $N/2$ elements, where N is the total number of all blocks, due to the sequence structure of the IHT. Moreover, the operations would change the sequence numbers of some blocks, and cause the recalculations of their tags, which would induce extra computational costs to the user and unnecessary communication overhead.

In view of these problems, this paper presents a public auditing scheme (DHT-PA) using a new data structure called dynamic hash table (DHT). Exploiting the DHT, our scheme can achieve dynamic auditing. Moreover, because DHT-PA migrates the authorized information from the CSP to the TPA, its computational costs and communication overhead are significantly smaller than the scheme based on skip list [15] and the one based on MHT [6] (see Table 1). DHT-PA also outperforms IHT-PA [16] in updating, as the times of updating operations on the DHT are much fewer than that on the IHT.

In addition, we extend DHT-PA to achieve privacy preserving by combining the homomorphic authenticator based on the public key with random masking generated by the TPA. Furthermore, we employ the well-known Boneh-Lynn-Shacham (BLS) signature and bilinear maps to achieve batch auditing. Specifically, our contribution in this work can be summarized as follows:

1. We present a novel public auditing scheme, which can completely support three vital functions, i.e., dynamic data auditing, privacy protection and batch auditing.

2. We design a new data structure named DHT to record data properties for auditing in the TPA, and by virtue of it, achieve rapid auditing and efficient data updating.
3. We formally prove the security of the proposed scheme, and evaluate its auditing performance by concrete experiments and comparisons with the state-of-the-art schemes. The results demonstrate that the proposed scheme can effectively achieve secure auditing in clouds, and outperforms the previous ones in computation complexity, storage costs and communication overhead.

The rest of the paper is organized as follows: In Section 2, we review the related work regarding cloud storage auditing. We introduce the background and necessary preliminaries for our work in Section 3, and present our scheme based on DHT in Section 4. We prove and analyze the security of our scheme in Section 5, which is followed by the comprehensive performance evaluation through experiments and comparisons with some existing schemes in Section 6. Finally, Section 7 gives the concluding remarks.

2 Related Work

In recent years, cloud storage auditing has attracted increasing attention. One of the earliest related work is "proof of retrievability (PoRs)" presented by Juels et al. [8] in 2007, which can check the correctness of data stored on the CSP and ensure data's retrievability with the use of error-correcting code. However, PoRs is a typical private auditing solution, and does not support auditing by the third party. In the same year, Ateniese et al. [9] first presented an original public auditing scheme, provable data possession (PDP), which employs homomorphic tags based on RSA and can remotely check the integrity of outsourced data by randomly sampling a few blocks from the file. As mentioned above, compared with the private auditing, the public auditing can provide dependable verification results and greatly reduce users' unnecessary overhead by introducing an independent TPA. Thus, it is believed to be more practical and promising.

Besides, there are some other significant concerns for cloud storage auditing, such as, privacy protection, batch auditing and dynamic auditing.

To address the data-leakage concern, Wang et al. [11], [12] first presented a privacy-preserving auditing protocol. By integrating the homomorphic authenticator with the random masking, this protocol can guarantee that the TPA could not obtain any knowledge on the data content stored in the cloud servers during the whole verification process. Particularly, the authors [10], [11], [12] observantly pointed out that privacy protection is indispensable for achieving the public auditability.

Moreover, Wang et al. [11], [12] extended their privacy-preserving auditing protocol into a multiuser setting to support batch verification for better efficiency. Later, Zhu et al. [13] proposed a cooperative PDP (CPDP) scheme exploiting the homomorphic verifiable response and hash index hierarchy to achieve batch auditing in multi-clouds scenarios. Further, Yang et al. [14] presented another public auditing scheme for both multi-clouds and multi-users without introducing any trusted organizer. Essentially, the batch auditing for multi-clouds is a task of the CSP that organizes the auditing information from different cloud servers [14]. However, in terms of the batch verification performed by the TPA, the difficult point is how to effectively handle multiple audit requests from different users [11], [12], [13], [14]. A practical solution for this problem is to first aggregate the different data block tags produced by various users and then verify them as a whole [11], [12], [14], which is also adopted in this work to achieve batch auditing.

To achieve dynamic data auditing, Erway et al. [15] extended the original PDP model [9] by introducing a rank-based authenticated skip list, and presented a dynamic provable data possession scheme. Their foremost contribution is to demonstrate a general pattern for dynamic data auditing, i.e., incorporating dynamic data structures with verification algorithms. Later, Wang et al. [6] presented another classic public auditing scheme for dynamic auditing using Merkle Hash Tree, which simultaneously supports privacy-preserving and batch verification. However, both the above schemes would incur heavy computational costs of the TPA and large communication overhead during the updating and verification processes [16]. Thus, Zhu et al. [16] proposed another public auditing scheme (IHT-PA) based on Index Hash Table (IHT). Compared with the former ones, this scheme organizes the data properties for auditing using the IHT, and stores them in the TPA instead of the CSP. Consequently, it can reduce the computational costs and communication overhead. Nevertheless, its updating operations (particularly, the insertion and deletion ones) are inefficient, because they would induce the adjustment of average $N/2$ elements in the IHT, where N denotes the number of all blocks, due to the sequence structure of the IHT. Moreover, the operations would inevitably change the sequence numbers of some blocks, and finally cause the recalculations of their tags, which would induce more extra computational costs of the CSP and unnecessary communication overhead. Therefore, in this paper, we are motivated to design a new data structure, DHT, to achieve more efficient data updating and auditing.

To highlight the differences among our scheme and the existing ones, Tables 1 and 2 respectively give the comparisons of performance and functions. It is not hard to see that the proposed scheme (DHT-PA) can support all the given auditing functions, and achieve data updating and various audits in a more cost-effective way.

3 Background and Preliminaries

3.1 Problem Statement

In this work, we concentrate on the design of an effective public auditing scheme based on the DHT illustrated in Fig. 1, which involves the following three entities: **User**, who stores a great quantity of data files in the cloud, can be an individual or a organization; **Cloud Service Provider**, who manages and coordinates a number of cloud servers to offer scalable and on-demand outsourcing data services for users; and **Third Party Auditor**, who can verify the reliability of the cloud storage services (CSS) credibly and dependably on behalf of the users upon request. Users can be relieved of the burden of storage and computation while enjoying the storage and maintenance service by

TABLE 2
Function Comparison of Auditing Schemes for Cloud Storage

Schemes	Public Auditing	Privacy Protection	Dynamic Auditing	Batch Auditing
PoRs [8]	×	—	×	×
PDP [9]	√	×	×	×
CPDP [13]	√	√	×	√
DAP [14]	√	√	√	√
DPDP(skip list) [15]	×	—	√	×
DPDP(MHT) [6]	√	√	√	√
IHT-PA [16]	√	√	√	○
DHT-PA	√	√	√	√

Note: *"√" means "support"; "×" means "not support"; "—" means "no demand"; and "○" means "not mentioned".*

outsourcing their data into the CSP. However, due to the loss of local possession of the data, they are keen to ensure the correctness and integrity of their data periodically. To obtain a convincing answer as well as alleviate the users' burden potentially induced by the frequent verification, the TPA is involved to check the integrity of the users' data stored in the cloud. However, in the whole verification process, the TPA is not expected to be able to learn the actual content of the users' data for privacy protection.

We assume the TPA is credible but curious. In other words, the TPA can perform the audit reliably, but may be curious about the users' data. In addition, the CSP is considered to be dishonest. That is to say, the CSP may choose to hide the fact of some data being corrupted motivated by self-interest. Specially, the CSP may launch the following attacks to the TPA:

- *Forge attack.* The CSP may forge the data blocks and/or their tags to deceive the verifier.
- *Replacing attack.* The CSP may want to pass the verification by replacing a required block and its tag, which have been corrupted, with another block and its corresponding tag.
- *Reply attack.* The CSP may attempt to pass the verification using the proof generated from the previous ones or other former information.

To enable secure and efficient public auditing for cloud storage, our scheme is designed to achieve the following objectives:

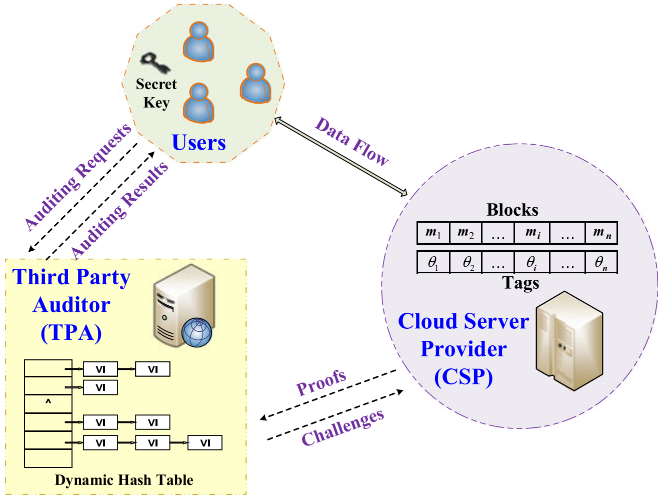

Secret Key
Users
Data Flow
Auditing Requests
Auditing Results
Third Party Auditor (TPA)
VI
^
Dynamic Hash Table
Blocks
m_1 m_2 … m_i … m_n
θ_1 θ_2 … θ_i … θ_n
Tags
Cloud Server Provider (CSP)
Proofs
Challenges

Fig. 1. System architecture.

1. Public auditing: anyone (not only the users) is allowed to have the capability to verify the correctness and integrity of the users' data stored in the cloud.
2. Storage correctness: the CSP, which does not correctly store users' data as required, cannot pass the verification.
3. Blockless verification: no data block needs to be retrieved by the TPA during the verification process.
4. Dynamic data auditing: dynamic data operations should be supported while the efficient public auditing is achieved.
5. Privacy preserving: the TPA cannot derive any actual content of users' data from the received auditing information.
6. Batch auditing: the TPA can handle multiple auditing tasks from various users in a fast and cost-efficient manner.
7. Lightweight: the verification should be performed with the minimum communication and computation overhead.

3.2 Dynamic Hash Table

As previously mentioned, it is popular to introduce an authenticated data structure to achieve dynamic auditing. The PDP based on skip list [15] and the MHT-based public auditing scheme [6] are typical representatives. However, they would incur heavy computational costs of the TPA and large communication overhead during the updating and verification processes. Thus, Zhu et al. [16] introduced a simple data structure (see Table 3), called Index Hash Table, to record the changes of data blocks and help to

TABLE 3
Index Hash Table

No.	B_i	V_i	R_i	
0	0	0	0	← Used to head
1	1	2	r'_1	← Update
2	2	1	r_2	
3	4	1	r_3	← Delete
4	5	1	r_5	
5	5	2	r'_5	← Insert
⋮	⋮	⋮	⋮	
n	n	1	r_n	
$n+1$	$n+1$	1	r_{n+1}	← Append

Note: "No." is the serial number, "B_i" is the block number, "V_i" is the version number, "R_i" is a random integer.

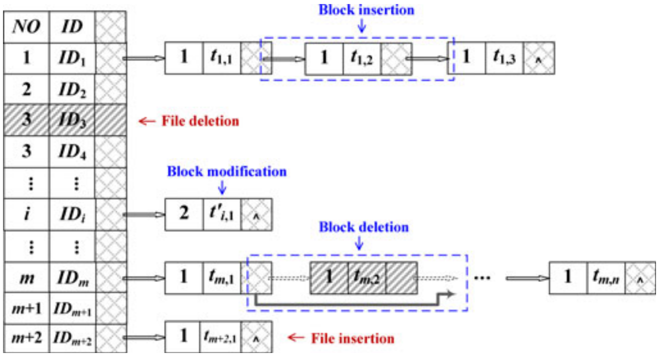


Fig. 2. Dynamic hash table.

generate the hash value of each bock in the verification process. The structure of the IHT is like a one-dimensional array, which contains index number, block number, version number and random value. The IHT-based scheme can also reduce the computational costs and communication overhead by storing the data properties for auditing using the IHT in the TPA instead of the CSP. Unfortunately, due to the sequence structure of the IHT, updating operations (particularly, insertion and deletion) on the IHT are inefficient, since they will lead to the adjustment of average $N/2$ elements, where N is the total number of all blocks. Moreover, during the insertion or deletion processes, the block numbers (B_i) of some blocks will be inevitably modified, which thereby will cause the regeneration of their corresponding block tags. That is obviously inefficient, and would cause more extra computational costs of users and unnecessary communication overhead. Therefore, we are motivated to design a new data structure, dynamic hash table, for better auditing efficiency.

The DHT, like the IHT, is employed by the TPA to track the latest version information (VI) of the user' data for auditing. However, differing from the IHT, the DHT is a two-dimensional data structure, as illustrated in Fig. 2. In the DHT, there are two kinds of basic elements, namely, file elements and block elements. Each file element consists of the index number (NO_i) of the given file (e.g., F_i), the File identifier (ID_i) and a pointer indicating its first block element, which is stored in an array-like structure. Each file is organized using a linked list with the corresponding file element as the header node. Each block element (e.g., the jth block of the ith file $m_{i,j}$) is one node of the corresponding file list, including the current version of the given block $v_{i,j,}$ its time stamp $t_{i,j}$ and a pointer indicating the next node. Accordingly, the operations on the DHT are divided into two categories: file operations and block operations, which both include search, insertion, deletion, and modification. Generally speaking, the block operations parallel those of the common linked list. To be specific, the search of a block is to locate the required element through visiting nodes from the first one in sequence; the insertion of a block after (before) an existing block is to first keep track of the given (previous) node and insert the new node after it; the deletion of a block is to first keep track of the required node and remove it from the current linked list. The search of a file is to locate the file element according to its index, and the other file operations would involve the manipulations on both file elements and block elements. Specifically, the insertion of a file involves inserting a file element into the file arrays and constructing a linked list that consists of corresponding block elements; the deletion of a file is to delete the linked list of the given file and its file element; the modification of a file is to update both the file element and related block elements.

Taking the advantages of linked lists, the DHT significantly outperforms the IHT in the insertion and deletion of blocks. Further, the insertion and deletion of a block are unable to cause the change of other VI records in the DHT. That is to say, the block tags, which include the hash values of the VI records, would not be influenced. Therefore, compared with the scheme based on the IHT, our scheme can effectively reduce the computational costs of the CSP and communication overhead in the updating process. Moreover, although the search operation on the DHT during the verification may cost more time than the IHT, it is too negligible to induce any material impact on the whole verification time. We will demonstrate the conclusion in the following text, and further prove that the verification time of our scheme is substantially smaller than that of the one based on the IHT.

3.3 Preliminaries

To make our paper self-contained, we would like first introduce some necessary cryptographic background for our proposed scheme.

Bilinear Map: Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of a large prime order p. A bilinear map is a map function e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties [19]: 1) Bilinearity: For ∀ $g_1, g_2, g_3 \in \mathbb{G}$ and ∀ a, $b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{a \cdot b}$, and $e(g_1, g_2 \cdot g_3) = e(g_2 \cdot g_3, g_1) = e(g_1, g_2) \cdot e(g_1, g_3)$; 2) Nondegeneracy: $e(g, g) \neq 1$, if g is a generator of $\mathbb{G}$; 3) Computability: e is efficiently computable.

Homomorphic Verifiable Authenticator (HVA): HVA is widely employed as a building block for public auditing [2], [4], [5], [6], [9], [10], [11], [12], [13], [14], [15], [16], [18], [19], [20], which allows a public auditor to verify the integrity of data stored in the cloud without accessing or downloading the original data. Generally, digital signatures (such as RSA-based signature and BLS-based signature) are used to generate HVAs. In this sense, HVAs can be considered as homomorphic verifiable signatures. In addition to unforgeability, HVAs satisfy the following properties:

- Blockless verifiability [9], [18]. Using HVAs, the TPA can verify the required file blocks without knowing their actual data content.
- Homomorphism [9]. Let $\mathbb{G}$ and $\mathbb{H}$ be multiplicative groups of a large prime order p, "$\oplus$" and "$\otimes$" be operations in $\mathbb{G}$ and $\mathbb{H}$. If a map function f: $\mathbb{G} \rightarrow \mathbb{H}$ satisfies homomorphism, then ∀ $g_1, g_2 \in \mathbb{G}$, $f(g_1 \oplus g_2) = f(g_1) \otimes f(g_2)$.
- Non-malleability [18]. Let σ_1 and σ_2 denote the signatures on blocks m_1 and m_2 respectively, α_1 and α_2 two random numbers in $\mathbb{Z}_p$. For the given block, $m' = \alpha_1 m_1 + \alpha_2 m_2$, a user, who does not know the private key sk, is not able to generate the signature σ' of m' by combining σ_1 and σ_2.

While wielding the same security strength (e.g. 80-bit security), a BLS-based signature (160 bit) is much shorter than an RSA-based signature (1,024 bit) [6], [18]. Therefore, BLS-based homomorphic verifiable authenticators (BLS-HVA) are

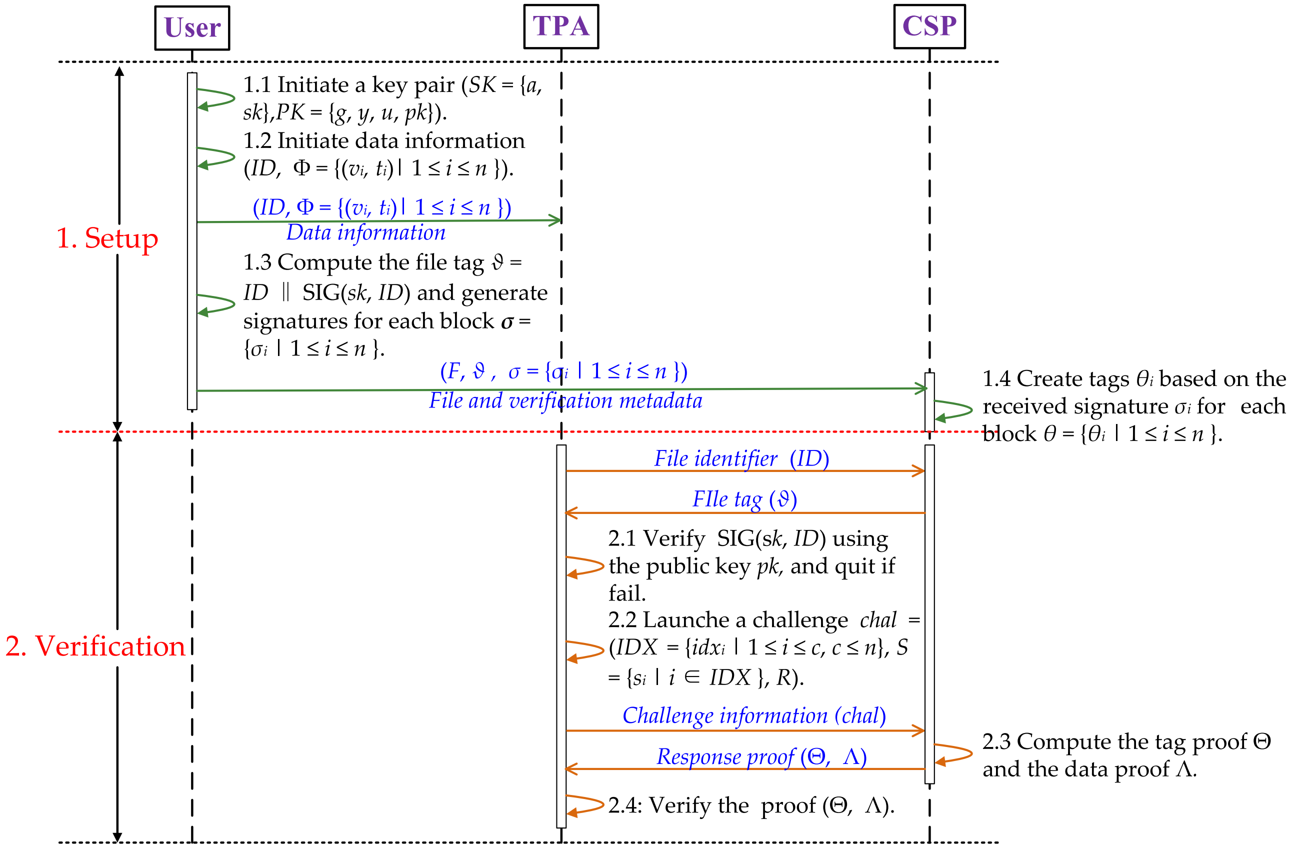

User
TPA
CSP
1. Setup
1.1 Initiate a key pair (SK = {a, sk},PK = {g, y, u, pk}).
1.2 Initiate data information (ID, Φ = {(v_i, t_i)| 1 ≤ i ≤ n }).
(ID, Φ = {(v_i, t_i)| 1 ≤ i ≤ n })
Data information
1.3 Compute the file tag ϑ = ID ‖ SIG(sk, ID) and generate signatures for each block σ = {σ_i | 1 ≤ i ≤ n }.
(F, ϑ , σ = {σ_i | 1 ≤ i ≤ n })
File and verification metadata
1.4 Create tags θ_i based on the received signature σ_i for each block θ = {θ_i | 1 ≤ i ≤ n }.
2. Verification
File identifier (ID)
FIle tag (ϑ)
2.1 Verify SIG(sk, ID) using the public key pk, and quit if fail.
2.2 Launche a challenge $chal$ = (IDX = {idx_i | 1 ≤ i ≤ c, c ≤ n}, S = {s_i | i ∈ IDX }, R).
Challenge information (chal)
Response proof (Θ, Λ)
2.3 Compute the tag proof Θ and the data proof Λ.
2.4: Verify the proof (Θ, Λ).

Fig. 3. Workflow of the dynamic verification with privacy-preserving.

more popularly adopted in the recent public schemes [6], [18], [19], [20]. Essentially, the BLS-HVA can be generated using the bilinear map as follows: Given a bilinear map e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, a secret key a and a pair of public key $\{g, y\} \in \mathbb{G}$, where $y = g^a$. The BLS-HVA (denoted by σ) for block m can be computed by $\sigma = (h(m))^a$, where h is a hash function. Finally, the TPA can verify the integrity of block m through checking whether $e(h(m), y) = e(\sigma, g)$ holds.

3.4 Security Assumptions

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

Computational Diffe-Hellman (CDH) Problem. Let $\mathbb{G}$ be a multiplicative cyclic group of a large prime order p and two random numbers $a, b \in \mathbb{Z}_p$, given $(g, g^a, g^b) \in \mathbb{G}$, compute the value $g^{ab} \in \mathbb{G}$.

Definition 1 (CDH Assumption). *For any probabilistic polynomial-time adversary $\mathfrak{A}$, the probability of solving the CDH problem is negligible, namely,*

$$Pr(\mathsf{A}_{\mathrm{CDH}}(g, g^a, g^b \in \mathbb{G}) \rightarrow g^{ab} \in \mathbb{G} : \forall a, b \underset{\mathrm{R}}{\in} \mathbb{Z}_p) \leq \varepsilon. \quad (1)$$

That is to say, the CDH problem is computationally intractable, or impossible to be solved in limited time.

Discrete Logarithm (DL) Problem. Let $\mathbb{G}$ be a multiplicative cyclic group of a large prime order p with a generator g, given $h \in \mathbb{G}$, compute $a \in \mathbb{Z}_p$, such that $h = g^a$.

Definition 2 (DL Assumption). *For any probabilistic polynomial-time adversary $\mathfrak{A}$, the probability of solving the DL problem is negligible, namely,*

$$Pr(\mathsf{A}_{\mathrm{DL}}(g, h \in \mathbb{G}) \rightarrow a \in \mathbb{Z}_p, \text{s.t. } h = g^a) \leq \varepsilon. \quad (2)$$

That is to say, the DL problem is computationally intractable, or impossible to be solved in limited time.

4 The Proposed Scheme Based on DHT

In this section, we will present our scheme based on DHT, which consists of the dynamic verification protocol with privacy-preserving described in Section 4.1, the dynamic updating operations detailed in Section 4.2, and the batch verification protocol introduced in Section 4.3.

4.1 Dynamic Verification with Privacy-Preserving

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be multiplicative cyclic groups of a large prime order p, and e be a bilinear map $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. H is a secure hash function with $H : \{0,1\}^* \rightarrow \mathbb{G}_1$; assume that the file (denoted by F) to be outsourced to the CSP is divided into n blocks, i.e., $F = \{m_1, m_2, \ldots, m_n\}$. Our dynamic auditing scheme (as shown in Fig. 3) involves two phases: setup and verification. The setup phase can be completed by the following steps:

STEP 1 (Key Initiation): The user generates a key pair $(SK = \{a, sk\}, PK = \{g, y, u, pk\})$, where $(sk,\ pk)$ is a random key pair of the user for signature, $a \in \mathbb{Z}_p$ is a random number, g and u are the random elements of $\mathbb{G}_1$ and $y = g^a$.

STEP 2 (Data Information Initiation): The user sends the data information $(ID, \Phi = \{(v_i, t_i) | 1 \leq i \leq n\})$ to the TPA, where ID is the unique identifier of $F, \Phi = \{(v_i, t_i)|$

$1 \leq i \leq n\}$ is the set of all blocks' VI, and v_i and t_i are respectively the version and timestamp of the block m_i. Upon receiving the data information, the TPA will add it into the DHT.

STEP 3 (Signature Generation): For each block m_i, the user generates a signature σ_i with the public key *u*, which can be described as follows:

$$\sigma_i = H(v_i||t_i) \cdot u^{(m_i + H(v_i||t_i))}. \tag{3}$$

Moreover, to ensure the integrity of the unique file identifier *ID*, the user computes the file tag $\vartheta = ID||\mathrm{SIG}(sk, ID)$, where SIG(*sk*, *ID*) is the signature on *ID* under the private key *sk*. Let the set of all blocks' signatures be $\sigma = \{\sigma_i|1 \leq i \leq n\}$. The user uploads F, ϑ and σ to the CSP, and deletes them from the local storage.

STEP 4 (Tag Generation): For each block m_i, the CSP further creates a tag θ_i based on the received signature σ_i using the bilinear map *e*, namely,

$$\theta_i = e(\sigma_i, y). \tag{4}$$

Let the set of all block tags be $\boldsymbol{\theta} = \{\theta_i|1 \leq i \leq n\}$. At last, the CSP should store the verification metadata $(\vartheta, \boldsymbol{\theta})$ along with the file $F = \{m_1, m_2, \ldots, m_n\}$.

Note that we assume that the CSP creates a tag θ_i for each block m_i here. However, it is not the best choice. Since each tag is an element of $\mathbb{G}_1$, the *n* tags for *n* blocks would cost a great deal of extra storage space, which is evidently uneconomic in terms of the pay-as-you-go pricing model. Therefore, the segment strategy is popularly adopted to reduce the space cost [14], [16], [22]. Specifically, each data block is further divided into *s* segments, i.e., $m_i = \{m_{i,1}, m_{i,2}, \ldots, m_{i,s}\}$, and its corresponding signature is calculated as follows:

$$\sigma_i = H(v_i||t_i) \cdot \prod_{j=1}^{s} u^{(m_{i,j} + H(v_i||t_i))}, \tag{5}$$

instead of Eq. (3). In this way, the storage overhead can be reduced to $1/s$ of the original one [14], [16]. However, it is worthwhile to emphasize that each signature and each tag still correspond to a block rather than a segment. In other words, the segment strategy performed by the user is just an approach to reduce the storage overhead of tags in the CSP, and it is transparent (invisible) to the TPA.

The workflow of verification phrase is as follows:

STEP 1 (File Identifier Check): The TPA first retrieves the file tag ϑ, and verifies the signature SIG(*sk*, *ID*) using the user's public key *pk*. If the verification fails, the TPA quits the verification by emitting FALSE; otherwise, it recovers the file identifier *ID* and goes to STEP 2.

STEP 2 (Challenge): The TPA launches a challenge by sending the challenge information $chal = (IDX = \{idx_i|1 \leq i \leq c, c \leq n\}, S = \{s_i|i \in IDX\}, R)$ to the CSP, where $IDX = \{idx_i|1 \leq i \leq c\}$ is the index set of the blocks to be checked, $S = \{s_i|i \in IDX\}$ is the set of random numbers belonging to $\mathbb{Z}_p$, *c* is the total number of the blocks to be checked, and *R* is a random masking calculated by $R = y^r$ (*r* is a random number belonging to $\mathbb{Z}_p$ here).

STEP 3 (Proof Generation): Upon receiving the challenge, the CSP would produce a response proof of data storage correctness, which consists of the tag proof and the data proof. The tag proof Θ is essentially the aggregated authenticator of all the tags to be checked, i.e.,

$$\Theta = \prod_{i \in IDX} \theta_i^{s_i}. \tag{6}$$

To generate the data proof, the CSP first computes the linear combination (denoted by *M*) of sampled blocks required in *chal*, i.e.,

$$M = \sum_{i \in IDX} s_i \cdot m_i. \tag{7}$$

Then, the CSP computes the data proof Λ as follows.

$$\Lambda = e(u, R)^M. \tag{8}$$

Finally, the CSP sends (Θ, Λ) back to TPA as the proof.

STEP 4 (Proof Check): To perform the verification, the TPA first computes the hash values for all VI of the challenged data blocks, and obtains the challenge hash H by combining them:

$$\mathrm{H} = \prod_{i \in IDX} H(v_i, t_i)^{s_i}. \tag{9}$$

Further, the TPA can verify the proof by checking the following equation:

$$\Lambda \cdot e(\mathrm{H} \cdot u^{\sum_{i \in IDX} s_i \cdot H(v_i||t_i)}, R) \stackrel{?}{=} \Theta^r. \tag{10}$$

If the equation holds, it outputs TRUE; otherwise, FALSE. The correctness of the above verification equation can be demonstrated as follows.

$$\begin{aligned}
&\Lambda \cdot e\left(\mathrm{H} \cdot u^{\sum_{i \in IDX} s_i \cdot H(v_i||t_i)}, R\right) \\
&= e(u, R)^{\sum_{i \in IDX} s_i \cdot m_i} \cdot e\left(\prod_{i \in IDX} H(v_i||t_i)^{s_i} \cdot u^{\sum_{i \in IDX} s_i \cdot H(v_i||t_i)}, R\right) \\
&= e\left(u^{\sum_{i \in IDX} s_i \cdot m_i}, R\right) \cdot e\left(\prod_{i \in IDX} H(v_i||t_i)^{s_i} \cdot u^{\sum_{i \in IDX} s_i \cdot H(v_i||t_i)}, R\right) \\
&= e\left(\prod_{i \in IDX} H(v_i||t_i)^{s_i} \cdot u^{\sum_{i \in IDX} s_i \cdot (m_i + H(v_i||t_i))}, g^{r \cdot a}\right) \\
&= e\left(\prod_{i \in IDX} \left(H(v_i||t_i) \cdot u^{(m_i + H(v_i||t_i))}\right)^{s_i}, g^{r \cdot a}\right) \\
&= e\left(\prod_{i \in IDX} \sigma_i^{s_i}, g^{r \cdot a}\right) \\
&= e\left(\prod_{i \in IDX} \sigma_i^{s_i}, g^{a}\right)^{r} \\
&= e\left(\prod_{i \in IDX} \sigma_i^{s_i}, y\right)^{r} \\
&= \prod_{i \in IDX} \theta_i^{s_i \cdot r} \\
&= \Theta^r.
\end{aligned}$$

So far, we have presented the verification process of the proposed dynamic auditing protocol. However, we would also like to introduce the following design considerations for efficiency and security.

Blockless verification: To achieve the public auditing without having to retrieve the original data block, we employ the HVA technique as described in Section 3.3 to generate block tags that are unforgeable metadata for data blocks. Accordingly, we just need to authenticate the block tags instead of original data blocks in the verification process. Moreover, although all PKC-based HVAs (such as RSA-HVA and BLS-HVA) can be employed in our scheme, we prefer to choose BLS-HVA for the shorter length of each block tag [6], [18].

Sampling verification: Given the huge amounts of data outsourced in the cloud, it is inadvisable to challenge all data blocks for checking the integrity. Instead, it is more affordable and practical for both the TPA and the CSP to achieve high-accuracy verification by only checking a portion of the data file, which is the so-called sampling verification. The previous studies [6], [9], [11], [12] have demonstrated the rationality and feasibility of this strategy. Generally, if t fractions of the given data are corrupted, the detection probability of the verification by checking random sampled c blocks is $P = 1 - (1 - t)^c$. Particularly, when $t = 0.01$, the TPA only needs to verify 460 (300) randomly chosen blocks to discover this corruption with probability larger than 99 percent (95 percent).

Privacy preservation: To prevent privacy leakage, in the process of generating the data proof Λ, the CSP would blind the linear combination M of sampled blocks with the random masking R provided by the TPA and the public key u. Although the TPA knows Λ, R and u, it is computationally infeasible for the TPA to obtain users' privacy data M under the DL assumption.

Data freshness: For each data block m_i, the user stores its VI (v_i, t_i) in the DHT located at the TPA, and generates a signature σ_i involving both m_i and (v_i, t_i) as Eq. (5) or (3) in the setup phrase. In the verification phrase, the TPA would employ the VI of each challenged data block to verify the proof, i.e., check whether Eq. (10) holds. Since the VI of each block cannot be forged, the presented protocol can guarantee that the challenged data reflects the latest updates, if the verification is passed.

In addition, let us consider the following derivation: by substituting Eq. (7) into Eq. (8), we can get

$$\Lambda = e(u, R)^M = e\left(u^{\sum_{i \in IDX} s_i m_i}, R\right) = \prod_{i \in IDX} e(u^{m_i}, R)^{s_i}. \quad (11)$$

According to this equation, it seems that the CSP may compute the data proof Λ as $\prod_{i \in IDX} e(u^{m_i}, R)^{s_i}$, and even keep u^{m_i} instead of m_i to pass the audit, which is a common issue existing also in the state-of-the-art schemes like DAP [14]. In the dynamic auditing scheme, in fact, this dishonest behavior can be easily discovered, because the dynamic data keep updating continually. That is to say, in the dynamic audit, the CSP could not pass the verification without actually storing the very version of the data. However, for the archived data that is not updated frequently, the CSP may indeed have the chance to pass the verification using u^{m_i}. Inspired by this, we plan to design another better method targeted for archived data in the future, which is beyond the scope of this paper. By the way, we would like to point out that it could be more effective to employ different and more proper auditing methods for diverse data.

4.2 Dynamic Updating

To support the efficient management of dynamic data, we design updating operations on the DHT for data blocks and files respectively. The updating operations of data blocks include block modification (M_{B}), block insertion (J_{B}) and block deletion (D_{B}) as follows.

Block modification: Suppose the ith block m_i of the file F needs to be modified to m_i'. To update the records in the DHT, the user first generates the data information (v_i', t_i') for m_i', and sends an update request $U_{\mathrm{TPA}}(F, M_{\mathrm{B}}, i, v_i', t_i')$ to the TPA. Upon receiving $U_{\mathrm{TPA}}(F, M_{\mathrm{B}}, i, v_i', t_i')$, the TPA finds the ith node in the linked list of F in the DHT, and replaces v_i and t_i with v_i' and t_i'. To update the data stored in the cloud, the user first generates the signature σ_i' for m_i' according to Eq. (3), and sends an update request $U_{\mathrm{CSP}}(F, M_{\mathrm{B}}, i, m_i, m_i', \sigma_i')$ to the CSP. Upon receiving $U_{\mathrm{CSP}}(F, M_{\mathrm{B}}, i, m_i, m_i', \sigma_i')$, the CSP obtains the new file version F' by replacing the block m_i in F with m_i', and the new tag set $\boldsymbol{\theta}'$ by replacing the tag θ_i in $\boldsymbol{\theta}$ with θ_i' generated based on σ_i' according to Eq. (4).

Block insertion: Suppose that a block m^* needs to be inserted into the file F after the ith block m_i. To update the records in the DHT, the user first generates the data information (v^*, t^*) for m^*, and sends an update request $U_{\mathrm{TPA}}(F, J_{\mathrm{B}}, i, v^*, t^*)$ to the TPA. Upon receiving $U_{\mathrm{TPA}}(F, J_{\mathrm{B}}, i, v^*, t^*)$, the TPA finds the ith node in the linked list of F in the DHT, and behind it inserts a new node including (v^*, t^*). To update the data stored in the cloud, the user first generates the signature σ^* for m^* according to Eq. (3), and sends an update request $U_{\mathrm{CSP}}(F, J_{\mathrm{B}}, i, m^*, \sigma^*)$ to the CSP. Upon receiving $U_{\mathrm{CSP}}(F, J_{\mathrm{B}}, i, m^*, \sigma^*)$, the CSP obtains the new file version F' by inserting m^* after m_i in F, and the new tag set $\boldsymbol{\theta}'$ by inserting the tag θ^* generated based on σ^* according to Eq. (4) after θ_i in $\boldsymbol{\theta}$.

Block deletion: Suppose the ith block m_i of the file F needs to be deleted. The user sends an update request $U_{\mathrm{TPA}}(F, D_{\mathrm{B}}, i)$ to the TPA. Upon receiving $U_{\mathrm{TPA}}(F, D_{\mathrm{B}}, i)$, the TPA finds and deletes the ith node in the linked list of F in the DHT. Moreover, the user sends an update request $U_{\mathrm{CSP}}(F, D_{\mathrm{B}}, i)$ to the CSP. Upon receiving the request, the CSP obtains the new file version F' and the new tag set $\boldsymbol{\theta}'$ by deleting m_i in F and θ_i in $\boldsymbol{\theta}$, respectively.

Compared with the block updating operations, the file updating operations, including file appending and file deletion, are relatively straightforward. If the user wants to append a new file F^*, he (or she) only needs to conduct STEP 2 to STEP 4 in the setup phase once again. To delete a file F, the user can send deletion requests to the TPA and the CSP respectively. Upon receiving the requests, the TPA would delete the file element and the whole linked list of F in the DHT, and the CSP would delete the all data of F and its verification metadata $(\vartheta, \boldsymbol{\theta})$.

4.3 Batch Verification

The core of the batch auditing is how to concurrently handle multiple verification tasks from different users. Specifically, this is equivalent to the verification of many signatures on different messages by different users. Thus, we introduce the aggregate BLS signature technique from bilinear maps [21] to achieve the batch verification, which the idea behind is to aggregate all the signatures by different users on various data blocks into a single short one and verify it for only one time to reduce the communication cost in the verification process.

Assume that there are k challenges launched by k different users. Upon receiving these challenges, for each user (e.g. the ith one, $i = 1, 2, \ldots, k$), the CSP first computes the tag proof (Θ_i) and the data proof (Λ_i), and then obtains the aggregate tag proof $\Theta_{\mathscr{B}}$ and the aggregate data proof $\Lambda_{\mathscr{B}}$ according to Eqs. (10) and (11), respectively.

$$\Theta_{\mathscr{B}} = \prod_{i=1}^{k} \Theta_i. \tag{12}$$

$$\Lambda_{\mathscr{B}} = \prod_{i=1}^{k} \Lambda_i. \tag{13}$$

At last, the CSP sends $\Theta_{\mathscr{B}}$ and $\Lambda_{\mathscr{B}}$ to the TPA as the response proof. Upon receiving the proof, the TPA computes the challenge hash H_i for each user, and gets the batch challenge hash $\mathrm{H}_{\mathscr{B}}$ by combining the challenge hash values of all k users, namely,

$$\mathrm{H}_{\mathscr{B}} = \prod_{i=1}^{k} \prod_{j \in IDX_i} H(v_{i,j}||t_{i,j})^{s_{i,j}}, \tag{14}$$

where $IDX_i = \{idx_{i,j} | 1 \leq j \leq c\}$ is the index set of the blocks to be checked for the ith user; $v_{i,j}$ and $t_{i,j}$ are respectively the version and timestamp of the block $m_{i,j}$ challenged by the ith user; and $S_i = \{s_{i,j} \mid j \in IDX_i\}$ is the set of random numbers for the ith user. Further, the TPA can verify the proof by checking the following equation:

$$\Lambda_{\mathscr{B}} \cdot \prod_{i=1}^{k} e\left(\mathrm{H}_{\mathscr{B}} \cdot u_i^{\sum_{j \in IDX} s_{i,j} \cdot H(v_{i,j}||t_{i,j})}, R_i\right) \stackrel{?}{=} \Theta^r_{\mathscr{B}}, \tag{15}$$

where u_i is the public key of the ith user, and R_i is the random masking for the ith user. If this equation holds, all the challenged files are proved to be correctly stored in the cloud; otherwise, there is a high possibility that one or some of them are incorrect. The correctness of the above verification equation for batch auditing can be demonstrated as follows.

$$\begin{aligned}
&\Lambda_{\mathscr{B}} \cdot \prod_{i=1}^{k} e\left(\mathrm{H}_{\mathscr{B}} \cdot u_i^{\sum_{j \in IDX} s_{i,j} \cdot H(v_{i,j}||t_{i,j})}, R_i\right) \\
&= \prod_{i=1}^{k} e(u, R_i)^{\sum_{j \in IDX} s_{i,j} \cdot m_{i,j}} \cdot \prod_{i=1}^{k} e\left(\prod_{j \in IDX_i} H\big(v_{i,j}||t_{i,j}\big)^{s_{i,j}} \cdot u_i^{\sum_{j \in IDX} s_{i,j} \cdot H\left(v_{i,j}||t_{i,j}\right)}, R_i\right) \\
&= \prod_{i=1}^{k} e\left(u^{\sum_{j \in IDX} s_{i,j} \cdot m_{i,j}}, R_i\right) \cdot \prod_{i=1}^{k} e\left(\prod_{j \in IDX_i} H\big(v_{i,j}||t_{i,j}\big)^{s_{i,j}} \cdot u_i^{\sum_{j \in IDX} s_{i,j} \cdot H\left(v_{i,j}||t_{i,j}\right)}, R_i\right) \\
&= \prod_{i=1}^{k} e\left(\prod_{j \in IDX} H\big(v_{i,j}||t_{i,j}\big)^{s_{i,j}} \cdot u^{\sum_{j \in IDX} s_{i,i} \cdot \left(m_{i,j} + H\left(v_{i,j}||t_{i,j}\right)\right)}, g^{r \cdot a_i}\right) \\
&= \prod_{i=1}^{k} e\left(\left(\prod_{j \in IDX} H\big(v_{i,j}||t_{i,j}\big)^{s_{i,j}} \cdot u^{\sum_{j \in IDX} s_{i,i} \cdot \left(m_{i,j} + H\left(v_{i,j}||t_{i,j}\right)\right)}\right)^{a_i}, g^r\right) \\
&= \prod_{i=1}^{k} e\left(\left(\prod_{j \in IDX} H\big(v_{i,j}||t_{i,j}\big) \cdot u^{\left(m_{i,j} + H\left(v_{i,j}||t_{i,j}\right)\right)}\right)^{a_i \cdot s_{i,j}}, g^r\right) \\
&= \prod_{i=1}^{k} e\left(\prod_{j \in IDX} \sigma_{i,j}^{a_i \cdot s_{i,j}}, g\right)^r \\
&= \prod_{i=1}^{k} e\left(\prod_{j \in IDX} \sigma_{i,j}^{s_{i,j}}, g^{a_i}\right)^r \\
&= \prod_{i=1}^{k} \left(\prod_{j \in IDX} \theta_{i,j}^{s_{i,j}}\right)^r \\
&= \prod_{i=1}^{k} \Theta_i^r \\
&= \Theta^r_{\mathscr{B}}.
\end{aligned}$$

5 Security Analysis

We will evaluate the security of the proposed scheme by analyzing the effectiveness of attack prevention strategies in this section.

Theorem 1 (Unforgeability of BLS-HVAs). *For any adversary, it is computationally infeasible to forge an HVA under*

BLS signature scheme, if the computational Diffe-Hellman (CDH) assumption in bilinear groups holds.

Proof. This theorem follows from Wang's work [6], where it has been proven that the HVA scheme is existentially unforgeable, in that BLS short signature scheme is secure with the assumption that the CDH problem is hard in bilinear groups [17]. Therefore, we omit the detailed proof here. □

Theorem 2 (Unforgeability of the proof). *In the DHT-PA scheme, it is computationally infeasible for the CSP to forge an auditing proof to pass the verification.*

Proof. To prove the theorem, we first design the following game according to the security game definition in [22]: The TPA sends a challenge $chal = (IDX = \{idx_i | 1 \leq i \leq c, c \leq n\}, S = \{s_i | i \in IDX\}, R)$ to the CSP. To pass the verification with Eq. (9), the auditing proof on the correct data file F sent back from the CSP should be (Θ, Λ). However, the CSP generates a proof on an incorrect data file F' as (Θ, Λ'), where $F' \neq F, \Lambda' = e(u, R)^{M'}, M' = \sum_{i \in \mathrm{IDX}} (\xi'_i)$, and $\xi'_i = s_i \cdot m'_i$. Define $\Delta\xi_i = \xi'_i - \xi_i$ for $i \in IDX$, and at least one element of $\{\Delta\xi_i\}_{i \in \mathrm{IDX}}$ is nonzero. If this proof can still pass the verification performed by the TPA, then the CSP wins the game; otherwise, it fails.

Assume that the CSP wins the game, then, according to Eq. (9), we have

$$e(u, R)^{M'} \cdot e\left(\prod_{i \in IDX} H(v_i, t_i)^{s_i} \cdot u^{\sum_{i \in IDX} s_i \cdot H(v_i || t_i)}, R \right) = \Theta^r. \tag{16}$$

Moreover, (Θ, Λ) is the correct proof, so we have

$$e(u, R)^{M} \cdot e\left(\prod_{i \in IDX} H(v_i, t_i)^{s_i} \cdot u^{\sum_{i \in IDX} s_i \cdot H(v_i || t_i)}, R \right) = \Theta^r. \tag{17}$$

According to the properties of bilinear maps described in Section 3.3, we can learn that

$$u^{\sum_{i \in IDX} \xi_i} = u^{\sum_{i \in IDX} \xi'_i} \Rightarrow u^{\sum_{i \in IDX} \Delta\xi_i} = 1. \tag{18}$$

Let $\mathbb{G}$ be a multiplicative cyclic group of a large prime order p, then for two elements $h_1, h_2 \in \mathbb{G}$, $\exists\, x \in \mathbb{Z}_p$ such that $h_2 = h_1^x$. Moreover, given h_1 and h_2, u can be generated as $u = h_1^{\alpha} h_2^{\beta} \in \mathbb{G}$, where α and β are random values belonging to $\mathbb{Z}_p$. Accordingly, we have

$$u^{\sum_{i \in IDX} \Delta\xi_i} = h_1^{\alpha \sum_{i \in IDX} \Delta\xi_i} \cdot h_2^{\beta \sum_{i \in IDX} \Delta\xi_i} = 1. \tag{19}$$

Obviously, for the given $h_1, h_2 \in \mathbb{G}$, $h_2 = h_1^x$, we can find a solution for the DL problem $x = -\alpha/\beta$ unless $\Sigma_{i \in IDX}(\Delta\xi_i) = 0$ or $\beta = 0$. However, as we defined above, at least one of element in $\{\Delta\xi_i\}_{i \in IDX}$ is nonzero, and β is a random element of $\mathbb{Z}_p$, meaning the probability of β being equal to 0 is $1/p$. Therefore, we can find a solution for the DL problem with a non-negligible probability of $1 - 1/p$, which contradicts the DL assumption described in Section 3.4. This completes the proof of the theorem. □

Theorem 3 (Immunity from replacing attacks). *In the DHT-PA scheme, the CSP is not able to pass the verification by replacing a specified block and its tag, with another block and its corresponding tag.*

Proof. We define the replacing-attack game as follows: The TPA sends a challenge $chal = (IDX = \{idx_i | 1 \leq i \leq c, c \leq n\}, S = \{s_i | i \in IDX\}, R)$ to the CSP. The CSP sends back auditing proof (Θ', Λ'). In the process of generating the proof, the information of the jth ($j \in IDX$) block is replaced with that of the kth ($k \neq j$) block. If this proof can still pass the verification performed by the TPA, then the CSP wins the game; otherwise, it fails.

According to the properties of bilinear maps, the left-hand side of Eq. (9) can be rewritten as follows.

$$\begin{aligned}
&\Lambda' \cdot e\Big(\mathrm{H} \cdot u^{\sum_{i \in IDX} s_i \cdot H(v_i || t_i)}, R\Big) \\
&= e(u, R)^{\sum_{i \in IDX \& i \neq j} (s_i \cdot m_i) + s_j \cdot m_k} \cdot e\left(\prod_{i \in IDX} H(v_i || t_i)^{s_i} \cdot u^{\sum_{i \in IDX} s_i \cdot H(v_i || t_i)}, R \right) \\
&= e(u^{\sum_{i \in IDX \& i \neq j} (s_i \cdot m_i) + s_j \cdot m_k}, R) \cdot e\left(\prod_{i \in IDX} H(v_i || t_i)^{s_i} \cdot u^{\sum_{i \in IDX} s_i \cdot H(v_i || t_i)}, R \right) \\
&= e\left(\prod_{i \in IDX} H(v_i || t_i)^{s_i} \cdot u^{\left(\sum_{i \in IDX \& i \neq j} (s_i \cdot m_i) + s_j \cdot m_k + \sum_{i \in IDX} s_i \cdot H(v_i || t_i)\right)}, R \right) \\
&= e\left(\prod_{i \in IDX} H(v_i || t_i)^{s_i} \cdot u^{\left(\sum_{i \in IDX \& i \neq j} s_i \cdot (m_i + H(v_i || t_i)) + s_j \cdot \left(m_k + H\left(v_j || t_j\right)\right)\right)}, R \right).
\end{aligned}$$

Further, the right-hand side of Eq. (9) can be developed to

$$\begin{aligned}
(\Theta')^r &= \prod_{i \in IDX \& i \neq j} \theta_i^{s_i \cdot r} \cdot \theta_k^{s_j \cdot r} \\
&= e\left(\prod_{i \in IDX \& i \neq j} \sigma_i^{s_i} \cdot \sigma_k^{s_j}, R \right) \\
&= e\left(\prod_{i \in IDX \& i \neq j} \left(H(v_i || t_i) \cdot u^{(m_i + H(v_i || t_i))}\right)^{s_i} \cdot \left(H(v_k || t_k) \cdot u^{(m_k + H(v_k || t_k))}\right)^{s_j}, R \right) \\
&= e\left(\prod_{i \in IDX} H(v_i || t_i)^{s_i} \cdot u^{\left(\sum_{i \in IDX \& i \neq j} s_i \cdot (m_i + H(v_i || t_i)) + s_j \cdot (m_k + H(v_k || t_k))\right)}, R \right).
\end{aligned}$$

If the verification is assumed to be passed, then $H(v_j \parallel t_j) = H(v_k \parallel t_k)$, i.e., $v_j = v_k$ and $t_j = t_k$. However, we have defined that $k \neq j$, so t_j must not be identical to t_k. That is to say, the equation $H(v_j \parallel t_j) = H(v_k \parallel t_k)$ does not hold absolutely. Thus, we can conclude that the CSP cannot win the game. This completes the proof of the theorem. □

Theorem 4 (Immunity from replay attacks). *In the DHT-PA scheme, it is impossible for the CSP to pass the verification using the former information.*

Proof. We define the replay-attack game as follows: The TPA sends a challenge $chal = (IDX = \{idx_i | 1 \leq i \leq c, c \leq n\}, S = \{s_i | i \in IDX\}, R)$ to the CSP. The CSP sends back auditing proof (Θ', Λ'). In the process of generating the proof, the information of the jth ($j \in IDX$) block is substituted with its previous information (We differentiate the

TABLE 4
Comparison of Communication Costs

Schemes	Communication Costs	
	Verification	Updating
DPDP(MHT) [6]	$cO(logn)$	$O(logn)$
DPDP(skip list) [15]	$cO(logn)$	$O(logn)$
DAP [14]	$O(c)$	$O(1)$
IHT-PA [16]	$O(c)$	$O(1)$
DHT-PA	$O(c)$	$O(1)$

Note: n *is the whole number of the blocks in a file; and* c *is the number of the verified blocks when auditing a file.*

TABLE 5
Comparison of Storage Costs

Schemes	Storage Costs	
	CSP	TPA
DPDP(MHT) [6]	$\mu \cdot (2^{logn+1} - 1 + n)$	—
DPDP(skip list) [15]	$\mu \cdot (2^{logn+1} - 1 + n)$	—
DAP [14]	$n \cdot \mu$	$v_1 \cdot n$
IHT-PA [16]	$n \cdot \mu$	$v_2 \cdot n$
DHT-PA	$n \cdot \mu$	$v_3 \cdot (n+1)$

Note: n *is the number of the data blocks; μ is the bit length of every element in $\mathbb{Z}_p$, v_1 v_2, and v_3 are respectively the bit lengths of each record in the ITable, the IHT and the DHT.*

previous parameters from the correct ones with a single quote mark, e.g., m'_j and m_j are the old block and the correct block respectively). If this proof can still pass the verification performed by the TPA, then the CSP wins the game; otherwise, it fails.

The proof of this theorem is very similar to that of Theorem 3. It is not hard to infer that the equation $H(v_j \parallel t_j) = H(v'_j \parallel t'_j)$ should hold, if the verification is assumed to be passed. However, it is absolutely impossible that the current timestamp t_j is identical with the old one t'_j. Thus, the CSP cannot win the game. This completes the proof of the theorem. □

6 Performance Evaluation

In this section, we will evaluate the performance of our scheme (DHT-PA) and compare it with the state-of-the-art schemes.

6.1 Communication Costs

As mentioned above, it is a popular strategy for achieving dynamic data auditing to incorporate a certain special data structure with verification algorithms. Besides DHT-PA, there are many typical schemes, such as, the DPDP based on MHT [6], the DPDP based on skip list [15], DAP [14], and IHT-PA [16]. Table 4 shows the communication costs of all the five schemes during the verification and updating, from which we can learn that the communication costs of the first two schemes are apparently more (*logn* times) than those of the others. The reasons for that are twofold. First, the former two schemes store all the metadata for auditing in the CSP while the others save the metadata except the tags in the TPA. In other words, the latter three schemes can reduce the communication costs compared to the former ones by migrating the auditing metadata except the tags from the CSP to the TPA. Second, for verifying a data block, the former two schemes involve *logn* times more metadata queries than the others, which also suggests that the data structures used in the latter three schemes are simpler but more effective than the former ones.

6.2 Storage Costs

We further compare the storage costs (not including the users' data) of DHT-PA with the previous four schemes. Without loss of generality, we take a data file *F* with *n* blocks as an example to analyze storage costs in the CSP and the TPA.

In the DPDP based on MHT and the DPDP based on skip list, there is no storage cost in the TPA (in fact, the DPDP based on skip list does not involve any TPA, since it is a private auditing protocol), and the storage costs in the CSP stem from the requirements for storing the tags and the auditing metadata organized with the MHT or the skip list. Specifically, the storage costs for the tags are $n{\cdot}\mu$, where μ is the bit length of every element in $\mathbb{Z}_p$; the storage costs for the MHT (or skip list) are $(2^{\log n+1} - 1){\cdot}\mu$, because there are $2^{\log n+1} - 1$ nodes in the MHT (or skip list). In the other schemes, the CSP stores the tags, and the TPA stores the auditing metadata organized with a data structure. Therefore, their storage costs in the CSP are $n{\cdot}\mu$. However, their storage costs in the TPA are different due to their distinct data structures. Specifically, let the bit length of each record be v_1 in the metadata table (called ITable) of DAP, v_2 in the IHT, and v_3 in the DHT, then, 1) the storage costs of DAP in the TPA are $v_1 \cdot n$, since there is a record for each block of the file in the ITable; 2) the storage costs of IHT-PA in the TPA are $v_2 \cdot n$ for the same reason above; and 3) the storage costs of DHT-PA in the TPA are $v_3{\cdot}(n + 1)$, because there are n records for all data blocks and one record for the file in the DHT. We list all their storage costs in Table 5.

Note that in these schemes, $\mu = 160$ bits (due to using the BLS signature), $v_1 = v_2 = 16$ bits, and $v_3 = 12$ bits. Accordingly, we can learn that: 1) although the latter three schemes migrate the auditing metadata stored in the data structures from the CSP and to the TPA for the sake of reducing the communication costs and enhancing the verification efficiency, their whole storage costs are still fewer than those of the former two schemes. 2) Particularly, DHT-PA induces the fewest storage costs, indicating the structural advantage of the DHT.

6.3 Computational Costs

In this section, we would like to further evaluate the computational costs of DHT-PA and compare them with IHT-PA and DAP (We do not include the DPDP based on MHT and the DPDP based on skip list in the comparison experiments, because they involve more metadata than IHT-PA, DAP and DHT-PA, and apparently need more computational costs). We simulate the computations involved in these schemes on an HP workstation with an Intel Core i5-3470 CPU at 3.2 GHz, 8 GB RAM and 7200 RPM 500 GB Serial ATA drive with a 32 MB buffer. All algorithms are implemented using the Pairing-Based Cryptography (PBC) library version 0.5.14. We employ an MNT d159 curve,

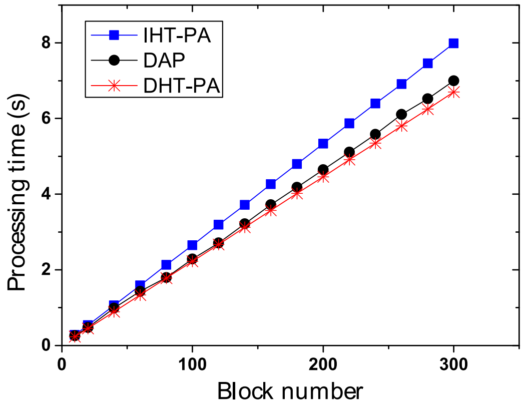


Fig. 4. The user's processing time for different block numbers in the setup phase (block size = 50 KB).

which has a 160-bit group order. Thus, p in the experiments is a 160-bit length prime. Moreover, in all experiments, we typically set the segment number to be 20, i.e., $s = 20$. All the statistical results are the averages of 20 trials.

6.3.1 Computational Costs in the Setup Phase

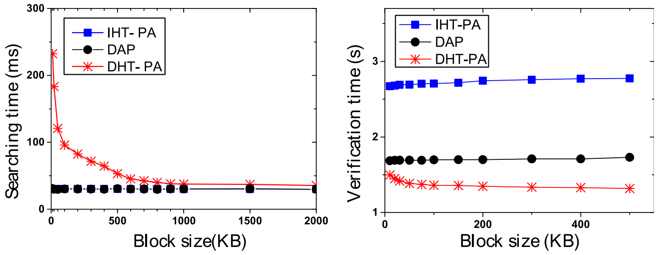
Fig. 4 shows the experimental result of the user's processing time for the different numbers of data blocks with the same size (e.g. 50 KB) in the setup phase, from which we can learn that: 1) the user's processing time is proportional to the block number; and 2) to handle the same number of data blocks, DHT-PA spends less time than IHT-PA and DPA. That is to say, the computational costs of the user in DHT-PA are fewer than those in IHT-PA and DPA.

6.3.2 Computational Costs in the Verification Phase

As mentioned in Section 3.2, the search time on the DHT is slightly more than that on the IHT used in both IHT-PA and DPA. However, as shown in Fig. 5a, the search time on DHT decreases with the increase of the block size, and the search time on the DHT and the IHT has insignificant gaps, when the block size is larger than 1,000. Further, we can learn from Fig. 5b that the verification time of DHT-PA is much less than IHT-PA and DPA, which suggests that the advantages of the verification algorithms in DHT-PA significantly outweigh the disadvantages induced by the searching operation.

6.3.3 Computational Costs in the Updating Phase

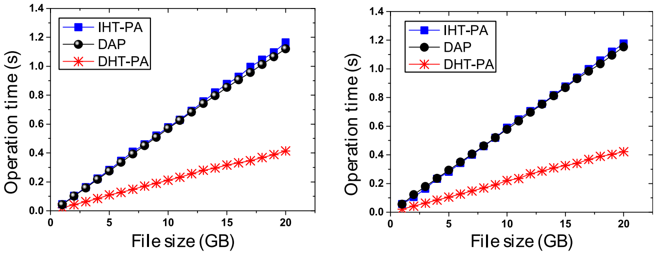
Figs. 6 and 7 show the computational costs of block / file operations on the IHT and the DHT in the updating phase, from which we can learn that: 1) Due to employing the same data structure (i.e, the IHT), the operation time of IHT-PA and DAP is almost identical. As mentioned in [14], however, the whole updating time of DAP would be less than that of IHT-PA, because DAP can effectively avoid the regeneration of block tags while performing insertion or deletion operations, compared with IHT-PA. 2) The time of the block operations both on the DHT and the IHT increases with the file size, but the time of operations on the DHT is much less than that on the IHT. 3) For the file operations, the time of operations on the IHT and the DHT are in proportion to the size of the file. Compared with the former, the latter has a much smaller initial value (for the file size equal to 1 GB) but a faster growth rate. Specifically, for file appending (deletion) operations, the time of operations on the DHT is less than that on the IHT when the file size is not larger than 23 GB (15 GB). Moreover, the smaller the file size, the larger the gap between the time of operations on the DHT and that on the IHT. In practice, the data files with small sizes are decidedly in the thumping majority. In this sense, DHT-PA would outperform IHT-PA and DPA in the whole efficiency of the updating process in the TPA.



(a). The searching time during the verification.

(b). The verification time with different block size.

Fig. 5. The experimental results for the searching time and verification time in the verification phase (file size = 1 GB).



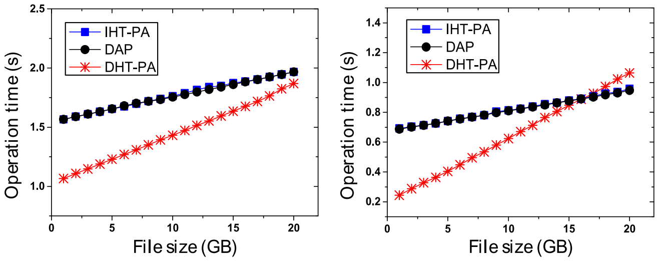
(a). Time of the block insertion operations for different file sizes

(b). Time of the block deletion operations for different file sizes

Fig. 6. Time of the updating operations for blocks (block size = 50 KB).

6.3.4 Computational costs of batch auditing

We also evaluate the performance of DHT-PA in the batch auditing scenario and compare it with DAP. The experimental results, as shown in Fig. 8, suggests that: 1) the batch auditing can not only simultaneously handle different verifications from multiple-users, but also reduce the computational costs on the TPA side compared with performing the individual auditing many times; and 2) the batch auditing protocol in DHT-PA is more efficient than that in DAP.



(a). Time of file appending operations for different file sizes

(b). Time of file deletion operations for different file sizes

Fig. 7. Time of the updating operations for files (block size = 50 KB).

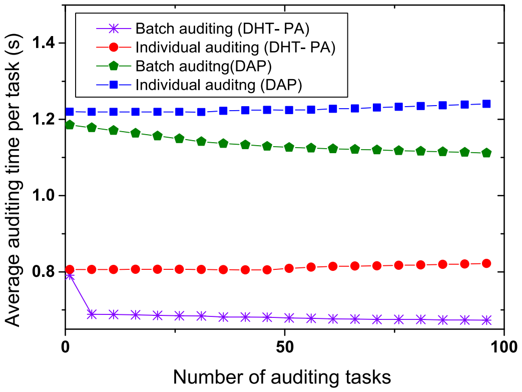


Fig. 8. Performance comparison between individual auditing and batch auditing. The average auditing time per task is computed by dividing total auditing time by the number of auditing tasks.

7 Conclusions

Nowadays, cloud storage, which can offer on-demand outsourcing data services for both organizations and individuals, has been attracting more and more attention. However, one of the most serious obstacles to its development is that users may not fully trust the CSPs in that it is difficult to determine whether the CSPs meet their legal expectations for data security. Therefore, it is critical and significant to develop efficient auditing techniques to strengthen data owners' trust and confidence in cloud storage. In this paper, we are motivated to present a novel public auditing scheme for secure cloud storage using dynamic hash table, which is a new two-dimensional data structure used to record the data property information for dynamic auditing. Differing from the existing works, our scheme migrates the auditing metadata excerpt the block tags from the CSP to the TPA, and thereby significantly reduces the computational cost and communication overhead. Meanwhile, exploiting the structural advantages of the DHT, our scheme can also achieve better performance than the state-of-the-art schemes in the updating phase. In addition, for privacy preservation, our scheme introduces a random masking provided by the TPA into the process of generating proof to blind the data information. Moreover, our scheme further exploits the aggregate BLS signature technique from bilinear maps to perform multiple auditing tasks simultaneously, of which the principle is to aggregate all the signatures by different users on various data blocks into a single short one and verify it for only one time to reduce the communication cost in the verification process. We formally prove the security of our scheme, and evaluate the auditing performance by detailed experiments and comparisons with the existing ones. The results demonstrate that our scheme can effectively achieve secure auditing in clouds, and induce significantly fewer costs of storage, communication and computation than the previous schemes.

Moreover, we would like to point out that no single method can achieve perfect audits for all types of cloud data, just as no standard has a universal validity. Thus, it may be a new trend to design a more effective scheme, including different audit strategies for various types of cloud data, which is also the direction for our future work.

Acknowledgments

The authors sincerely thank the anonymous reviewers and editors for their constructive comments that greatly improved the manuscript. This work was supported in part by Natural Science Foundation of China under Grant Nos. U1405254, U1536115, 61302094, 61370007 and 61202468, Natural Science Foundation of Fujian Province of China under Grant No. 2014J01238, Research Program for Outstanding Young Teachers in Higher Education Institutions of Fujian Province of China under Grant No. MJK2015–54, Education and Science Research Program for Young and Middle-aged Teachers of Fujian Province of China under Grant No. JA13012, Promotion Program for Young and Middle-aged Teacher in Science & Technology Research of Huaqiao University under Grant No. ZQN-PY115, and Program for Science & Technology Innovation Teams and Leading Talents of Huaqiao University under Grant No. 2014 KJTD13.

References

[1] H. Dewan and R. C. Hansdah, "A survey of cloud storage facilities," in *Proc. 7th IEEE World Congress Serv.*, Jul. 2011, pp. 224–231.

[2] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 220–232, Apr.–Jun. 2012.

[3] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, 2012.

[4] J. Ryoo, S. Rizvi, W. Aiken, and J. Kissell. "Cloud security auditing: Challenges and emerging approaches," *IEEE Security Privacy*, vol. 12, no. 6, pp. 68–74, 2014.

[5] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," *IEEE Netw.*, vol. 24, no. 4, pp. 19–24, 2010.

[6] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2011.

[7] F. Sebé, J. Domingo-Ferrer, A. Martínez-Ballesté, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1034–1038, Aug. 2008.

[8] A. Juels and B. S. Kaliski Jr., "PoRs: Proofs of retrievability for large files," in *Proc. ACM Conf. Comput. Commun. Security*, 2007, pp. 584–597.

[9] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 598–609.

[10] 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

[11] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving public auditing for data storage security in cloud computing," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

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

[13] Y. Zhu, H. Hu, G. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multi-cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2231–2244, Dec. 2012.

[14] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013.

[15] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Comput. Comm. Security*, 2009, pp. 213–222.

[16] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C.-J. Hu, "Dynamic audit services for outsourced storage in clouds," *IEEE Trans. Serv. Comput.*, vol. 6, no. 2, pp. 227–238, Apr.–Jun. 2013.

[17] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” in *Proc. 7th Int. Conf. Theory Appl. Cryptol. Inf. Security: Adv. Cryptol.*, 2001, vol. 2248, pp. 514–532.

[18] B. Wang, B. Li, and H. Li, “Panda: Public auditing for shared data with efficient user revocation in the cloud,” *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 92–106, Jan./Feb. 2015.

[19] C. Liu, R. Ranjian, X. Zhang, C. Yang, D. Georgakopoulos, and J. Chen, “Public auditing for big data storage in cloud computing– a survey,” in *Proc. 16th IEEE Int. Conf. Comput. Sci. Eng.*, 2013, pp. 1128–1135.

[20] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, and K. Ramamohanarao, “Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2234–2244, Sep. 2014.

[21] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, “Aggregate and verifiably encrypted signatures from bilinear maps,” in *Proc. 22nd Int. Conf. Theory Appl. Cryptograph. Techn.*, 2003, pp. 416–432.

[22] H. Shacham and B. Waters, “Compact proofs of retrievability,” in *Proc. 14th Int. Conf. Theory Appl. Cryptol. Inf. Security: Adv. Cryptol.*, 2008, pp. 90–107.

Hui Tian received the BSc degree and the MSc degree in computer science in 2004 and 2007 from the Wuhan Institute of Technology, Wuhan, China, and the PhD degree in 2010 in computer science from the Huazhong University of Science and Technology, Wuhan, China. He is now an associate professor in the College of Computer Science and Technology, National Huaqiao University, Xiamen, China. His present research interests include network and information security, cloud computing security, and digital forensics. He has published more than 30 papers in refereed proceedings of conferences, journals and books, and got two patents. He is a member of IEEE and ACM, a senior member of China Computer Federation (CCF) and a member of the Technical Committee on Internet of CCF.

Yuxiang Chen received the BSc degree in software engineering in 2010 from the University of South China, Hengyang, China. She is currently working toward the MSc degree in computer science from the National Huaqiao University, Xiamen, China. Her interests are in the areas of cloud computing security, with current focus on secure auditing for data outsourcing in public clouds.

Chin-Chen Chang received both the BSc degree in applied mathematics in 1977 and the MSc degree in computer and decision sciences in 1979 from the National Tsinghua University, Hsinchu, Taiwan, and the PhD degree in computer engineering in 1982 from the National Chiao Tung University, Hsinchu, Taiwan. His current title is chair professor at the Department of Information Engineering and Computer Science, Feng Chia University, from Feburary 2005. Prior to joining the Feng Chia University, he was an associate professor in the Chiao Tung University, professor in the National Chung Hsing University, chair professor in the National Chung Cheng University. His present research interests include computer cryptography and information security, cloud computing, data engineering, and database systems. He has over 850 publications in major journals and international conferences in these areas. Since his early years of career development, he consecutively won Outstanding Youth Award of Taiwan, Outstanding Talent in Information Sciences of Taiwan, AceR Dragon Award of the Ten Most Outstanding Talents, Outstanding Scholar Award of Taiwan, Outstanding Engineering Professor Award of Taiwan, Chung-Shan Academic Publication Awards, Distinguished Research Awards of National Science Council of Taiwan, Top Fifteen Scholars in Systems and Software Engineering of the Journal of Systems and Software, etc. He is currently a fellow of the IEEE and a fellow of the IEE, United Kingdom.

Hong Jiang received the BSc degree in computer engineering in 1982 from the Huazhong University of Science and Technology, Wuhan, China, the MSc degree in computer engineering in 1987 from the University of Toronto, Toronto, Canada, and the PhD degree in computer science in 1991 from the Texas A&M University, College Station, Texas. Since Sep. 2015, he has been Nedderman professor and Department chair of Computer Science And Engineering in the University of Texas at Arlington, Arlington, TX. From 2013 to 2015, he was a program director in the CCF Division of the CISE Directorate at the National Science Foundation. From 1991 to 2015, he was Willa Cather professor in the Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, Nebraska, where from 2001 to 2007, he served as Vice Department chair. His present research interests include cloud computing, big data computing, cloud storage and computer storage systems, parallel I/O, computer architecture, and performance evaluation. He serves as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. He has over 200 publications in major journals and international conferences in these areas, including IEEE-TPDS, IEEE-TC, ACM-TOS, ACM TACO, JPDC, ISCA, MICRO, FAST, USENIX ATC, USENIX LISA, SIGMETRICS, MIDDLEWARE, ICDCS, IPDPS, OOPLAS, ECOOP, SC, ICS, HPDC, ICPP, etc. He is a fellow of the IEEE.

Yongfeng Huang received the BSc degree in applied physics in 1989 from the Hubei University, Wuhan, China, the MSc degree and the PhD degree in computer science, in 1994 and 2000, respectively, from the Huazhong University of Science and Technology, Wuhan, China. He is now a professor in the Department of Electronic Engineering, Tsinghua University, Beijing, China. His present research interests include cloud computing, cloud storage, network security, and next-generation Internet. He has published five books and over 80 research papers on computer network and multimedia communications. He is a senior member of the IEEE and China Computer Federation (CCF), and a member of the Technical Committee on Internet of CCF.

Yonghong Chen received the BSc degree in mathematics in 1997 from the Hubei Institute for Nationalities, Enshi, China, the MSc degree in applied mathematics in 2000 and the PhD degree in 2005 in systems and control engineering from the Chongqing University, Chongqing, China. He is now a professor in the College of Computer Science and Technology, National Huaqiao University, Xiamen, China. His present research interests include network and multimedia information security, cloud computing security, and applied cryptography. He is a member of the IEEE.

Jin Liu received the BSc degree in mechanical engineering in 2007 from the Wuhan University of Technology, Wuhan, China, and the PhD degree in 2013 in computer science from the Huazhong University of Science and Technology, Wuhan, China. He is now an assistant professor in the College of Computer Science and Technology, National Huaqiao University, Xiamen, China. His present research interests include cloud storage security, digital forensics, and information hiding. He is a member of the IEEE.