Provable Multiple-Replica Dynamic Data Possession for Big Data Storage in Cloud Computing

Huiying Hou[1], Jia Yu[1,2,3], and Rong Hao[1]
(Corresponding author: Rong Hao)
College of Computer Science and Technology, Qingdao University[1]
266071 Qingdao, China
Institute of Big Data Technology and Smart City, Qingdao University[2]
266071 Qingdao, China
State Key Laboratory of Information Security, Institute of Information Engineering[3]
Chinese Academy of Sciences, 100093 Beijing, China
(Email: hr@qdu.edu.cn)
(Received Feb. 21, 2017; revised and accepted June 20, 2017)

Abstract

In order to avoid the data loss in cloud storage, some users prefer to store multiple replicas on the cloud server. Multiple-Replica Provable Data Possession (MR-PDP) schemes are proposed to check the integrity of remote multiple-replica data. In most of the previous schemes, the user has to generate a homomorphism authenticator based on BLS signature or RSA signature for each block of each replica before uploading them to the cloud. This can incur high overhead for the user especially when the data is very big. In this paper, we make use of the algebraic signature technique to generate authenticator for each block of each replica. Because most operations of algebraic signature are XOR operations, it only needs minimal computation and communication cost. Moreover, we design a new data structure named Divided Map-Version Table (DMVT) to efficiently support full dynamic data operations. The performance analysis demonstrates that our scheme is very efficient for verifying the integrity of multiple-replica dynamic big data.

Keywords: Algebraic Signature; Dynamic Data; Multiple Replicas; Provable Data Possession

1 Introduction

Cloud storage brings enormous convenience to the cloud user. However, the user loses direct control of their data in the cloud storage system. In order to detect whether the user's data is unabridged, Provable Data Possession (PDP) schemes have been proposed [9, 10]. In 2007, Ateniese *et al.* [1] firstly presented the definition of Provable Data Possession (PDP). Subsequently, a lot of PDP schemes have been proposed such as [8, 11, 12, 15, 17–24, 26].

The above PDP schemes are designed for single replica of user data. If cloud server suffers from some irresistible disasters, the user may lose some important data permanently. Therefore, it is necessary for the user to store multiple replicas of important data on multiple servers.

In order to check the integrity of multiple copies of data, multiple-replica PDP schemes have been proposed [3]. In most schemes, one replication technology is used to generate multiple replicas in distributed storage system. However, this method cannot resist collusion attack of cloud servers. Cloud servers can make data owner believe that they truly stored all replicas while they only save one replica in fact. In order to solve this problem, Curtmola *et al.* [5] proposed a PDP scheme named Multiple-Replica Provable Data Possession (MR-PDP). In this scheme, multiple replicas of data are stored on multiple servers across multiple data center. Subsequently, other Multiple-Replica PDP schemes were proposed [7, 13, 27]. However, the above schemes can only support static data. In many practical applications, the data owner might frequently update the data stored in the cloud. In order to solve this problem, Ayad *et al.* [2] proposed a provable multi-replica data possession scheme supporting dynamic data. This scheme adopts the homomorphic authenticator to generate the tag for each data block. The data owner needs to generate $m \times n$ homomorphic authenticators if the data file is divided into n blocks and the cloud server stores m replicas. In the process of calculating homomorphic authenticator, there are a lot of modular multiplication operations. This may incur high computation overhead for the data owner. Furthermore,

this scheme cannot efficiently support dynamic data operations, especially for insert or delete operations. Therefore, how to design an efficient Multiple-Replica PDP scheme supporting dynamic data operations is an interesting topic.

The main contribution of this paper can be summarized as follows:

- We propose a provable multiple-replica dynamic data possession scheme for big data storage. We make use of algebraic signature to generate authenticator for each data block, which incurs very low computation overhead for the user.
- We design a new data structure - Divided Map-Version Table (DMVT) - to efficiently support full dynamic data operations (on data block level), such as insertion, deletion, modification, and append. When the large-scale outsourced data are frequently updated, the proposed scheme incurs minimum computation cost.

Organization. The rest of this paper is organized in the following way: In Section 2, we give the definition of system model and present our design goals. Then we introduce the algebraic signature and the proposed data structure DMVT in Section 3. In Section 4, we describe our proposed scheme in detail. The security and performance analysis of the proposed scheme is presented in Section 5. In Section 6, we give the conclusions.

2 Problem Statement

2.1 System Model

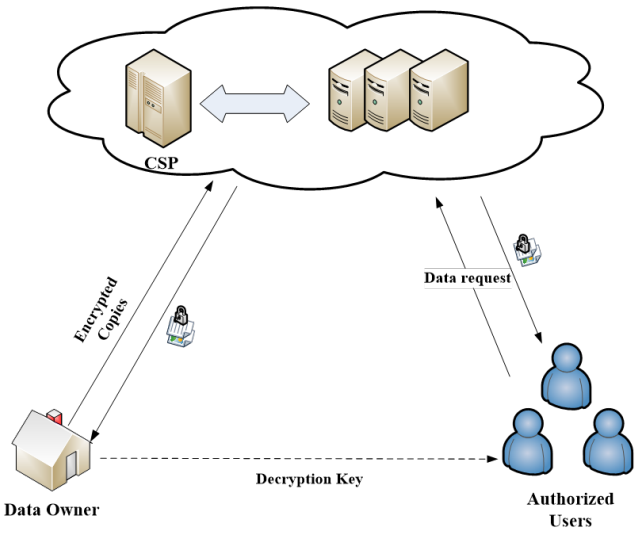

Figure 1: The system model

As shown in Figure 1, there are three types of entities in the system model: (1) Data Owner: an entity who can store large-scale data in the cloud, and then may perform modify, delete, insert, and append operations to update their outsourced data. (2) Cloud Storage Provider(CSP): an entity who provides storage service for the data owner and is in charge of managing the cloud servers. (3) Authorized Users: a collection of clients gain the access authorization from the data owner firstly, then they have the right to access the outsourced data and share the decryption key with the data owner. For the simplicity of description, we assume the data owner is in charge of checking the integrity of multiple replicas of data in our system model.

2.2 Design Goals

A Multiple-Replica PDP scheme should satisfy the following properties: (1) High efficiency: to allow data owner to efficiently check the integrity of multiple replicas of data. (2) Being against collusion attack: to ensure that colluded cloud servers cannot cheat users if they don't have all copies of data. (3) Supporting dynamic operations: to allow data owners to frequently update their outsourced data by performing insert, modify, delete, and append operations.

3 Preliminaries

In this section, we first introduce the algebraic signature technique used in our scheme; and then give the definition of our proposed data structure named DMVT.

3.1 Algebraic Signature

The algebraic signature is a type of hash functions with algebraic properties. The main property of algebraic signature is that the signature of the sum of some random file blocks is equal to the result of the sum of the signatures of the corresponding blocks. Therefore, we can compute the algebraic signature of the data block b_{ij} which is divided into s sectors:

$$S_\alpha(b_{ij}) = \sum_{k=1}^{s} b_{ijk} \cdot \alpha^{k-1}.$$

Note that α is an element in the Galois field. Moreover, b_{ij} denotes the j-th block of the i-th replica. There are some important properties of the algebraic signature [14]:

1) The algebraic signature of concatenation of message m_1 with l length and message m_2 can be computed as follows:

$$S_\alpha(m_1||m_2) = S_\alpha(m_1) \oplus l^\alpha S_\alpha(m_2).$$

2) The signature of the summation of several file blocks is equal to the summation of the signature of each block:

$$S_\alpha(b_{1j}+b_{2j}+...+b_{mj}) = S_\alpha(b_{1j})+S_\alpha(b_{2j})+...+S_\alpha(b_{mj}).$$

3.2 Divided Map-Version Table

The data structure named Divided Map-Version Table(DMVT) consists of two important components: (1) The logical index (L_j): a logical number of file blocks. (2) Version number (V_j): the current version of file blocks. The initial value of V_j is 1. When the data owner updates a data block, the corresponding V_j increases one. Assuming the file F is divided into 15 blocks, and 3 DMVTs are used to support dynamic operations, we display these 3 DMVTs in Figure 2. The DMVTs are stored on the data owner side.

DMVT1

Lj	Vj
1	1
2	1
3	1
4	1
5	1

{1,2,3,4,5}

DMVT2

Lj	Vj
6	1
7	1
8	1
9	1
10	1

{6,7,8,9,10}

DMVT3

Lj	Vj
11	1
12	1
13	1
14	1
15	1

{11,12,13,14,15}

Figure 2: An example of three DMVTs

4 The Proposed Scheme

4.1 Common Notations

$\phi : Z_q^* \times Z_q^* \to Z_q^*$, a pseudo-random function (PRF).

$\pi : Z_q^* \times \{1, 2, ..., n\} \to \{1, 2, ..., n\}$, a pseudo-random permutation (PRP).

$E_k(\cdot), D_k(\cdot)$: the encryption algorithm and the decryption algorithm of a symmetric cryptosystem with symmetric key k.

4.2 Scheme Description

The proposed scheme consists of six algorithms (*Setup*, *ReplicaGen*, *TagBlock*, *DataUpdate*, *Challenge*, *ProofGen*, *ProofVerify*).

Setup: Let G_1 be a multiplicative cyclic group generated by g with prime order q. The data owner randomly selects a secret key $k \xleftarrow{R} Z_q^*$, and computes a public key $y = g^x \in G_1$.

ReplicaGen: Assume that the file F is divided into n blocks $\{b_1, b_2, ..., b_n\}$. The data owner creates m differentiable replicas $\hat{F} = \{\hat{F}_i\}_{1 \le i \le m}$ that are stored on m cloud servers. A replica $\hat{F}_i$ is divided into n blocks $\hat{F}_i = \{\hat{b}_{ij}\}_{1 \le j \le n}$, where $\hat{b}_{ij} = E_k(i||b_j)$. Furthermore, the block $\hat{b}_{ij}$ is fragmented into s sectors with the same length. Denote block $\hat{b}_{ij} = \{\hat{b}_{ij1}, \hat{b}_{ij2}, ..., \hat{b}_{ijs}\}_{\substack{1 \le i \le m, \\ 1 \le j \le n}}$. So replica $\hat{F}_i = \{\hat{b}_{ijk}\}_{\substack{1 \le j \le n, , \\ 1 \le k \le s}}$, where each sector $\hat{b}_{ijk} \in Z_q$.

TagBlock: Given the distinct data file replicas $\hat{F} = \{\hat{F}_i\}_{1 \le i \le m}$, where $\hat{F}_i = \{\hat{b}_{ij}\}_{1 \le j \le n}$, the data owner generates a tag T_{ij} for each block $\hat{b}_{ij}$ by computing:

$$\begin{aligned} T_{ij} &= S_\alpha(\hat{b}_{ijk}||F_{id}||j||L_j||V_j) \\ &= \sum_{k=1}^{s} (\hat{b}_{ijk}||F_{id}||j||L_j||V_j) \cdot \alpha^{j-1} \end{aligned}$$

where L_j is the logical number of the block at physical position j, V_j is the current version of the block, and F_{id} is the unique name of the file F. In order to avoid the replay attack, the data owner computes:

$$\begin{aligned} C_{ij} &= S_\alpha(F_{id}||j||L_j||V_j) \\ &= \sum_{k=1}^{s} (F_{id}||j||L_j||V_j) \cdot \alpha^{j-1} \end{aligned}$$

The data owner computes $T_j = \sum_{i=1}^{m} T_{ij}$ and $C_j = \sum_{i=1}^{m} C_{ij}$ to reduce the storage overhead and the communication overhead of cloud servers. Hence, the CSP only needs to store n tags for the replicas $\hat{F} = \{F_i\}_{1 \le i \le m}$. Denote the set C as $\{C_j\}_{1 \le j \le n}$ and the set T as $\{T_j\}_{1 \le j \le n}$. The data owner sends $\{C, \hat{F}, T\}$ to the CSP, and deletes the local replicas and tags.

DataUpdate: The dynamic operations include Block Modification (denoted by **BM**), Block Insertion (denoted by **BI**), Block Append, and Block Deletion (denoted by **BD**).

- **Block Modification:** Assume that the data owner wants to modify a block b_j with b'_j in file $F = \{b_1, b_2, b_3, ..., b_n\}$ for all file replicas $\hat{F} = \{F_i\}_{1 \le i \le m}$. The data owner does as follows:
  1) Finds the corresponding V_j, then updates $V_j = V_j + 1$. The data owner recomputes C_{ij}.
  2) Generates m differentiable blocks $\{\hat{b}'_{ij}\}$. Divides $\hat{b}'_{ij} = E_k(i||b'_j)$ intossectors $\{\hat{b}'_{ij1}, \hat{b}'_{ij2}, ..., \hat{b}'_{ijs}\}$.
  3) Computes a new tag for each block $\hat{b}'_{ij}$ as follows:

$$\begin{aligned} T'_{ij} &= \sum_{k=1}^{s} S_\alpha(\hat{b}'_{ijk} \,\|F_{id}\, \|j\, \|L_j\, \|V_j\,) \\ &= \sum_{k=1}^{s} (\hat{b}'_{ijk} \,\|F_{id}\, \|j\, \|L_j\, \|V_j\,) \cdot \alpha^{j-1} \end{aligned}$$

And computes an aggregated tag $T'_j = \sum_{i=i}^{m} T'_{ij}$.

4) Sends a modification message $< F_{id}, BM, j, \{\hat{b}'_{ij}\}_{\substack{1\le i\le m,\\ 1\le j\le n}}, T'_j >$ to the CSP.

When the CSP receives the modification message from the data owner, he replaces the block $\hat{b}_{ij}$ with $\hat{b}'_{ij}$ for $1 \le i \le m$, and replaces T_j in the set T with T'_j.

- **Block Insertion:** Assume that the data owner wants to insert a new block $\hat{b}$ after the j-th block in file $F = \{b_1, b_2, ..., b_n\}$, and the new file is $F' = \{b_1, b_2, ..., b_j, \hat{b}, ..., b_{n+1}\}$. The data owner does as follows:

1) Finds the location of the j-th block in the DMVT table according to the tuples of the range of L_j.
2) Inserts a new table entry $< L_{j+1}, V_{j+1} >=< n+1, 1 >$ in the DMVT after position j. And recomputes C_{ij}.
3) Generates m distinct blocks $\{\hat{b}_i\}_{1\le i\le m}$, where $\hat{b}_i = E_k(i||\hat{b})$. Each block of $\{\hat{b}_i\}_{1\le i\le m}$ is fragmented into s sectors.
4) Computes a new tag $\hat{T}_i$ for each block $\hat{b}_i$ as follows:

$$\hat{T}_i = \sum_{k=1}^{s} S_\alpha(\hat{b}_{ik}||F_{id}||j+1||L_{j+1}||V_{j+1}) = \sum_{k=1}^{s} (\hat{b}_{ik}||F_{id}||j+1||L_{j+1}||V_{j+1}) \cdot \alpha^{j-1}$$

And computes an aggregated tag $T' = \sum_{i=1}^{m} \hat{T}_i$.

5) Sends an insert message $< F_{id}, BI, j, \{\hat{b}_i\}_{1\le i\le m}, T' >$ to the CSP.

Upon receiving the insert message, the CSP inserts the new block $\hat{b}$ for each file replica to generate new file replicas $\{\hat{F}_i\}_{1\le i\le m}$, and constructs the new file replicas $\{\hat{F}'_i\}_{1\le i\le m}$. The CSP also inserts T' after the position j of aggregated tags.

- **Block Append:** The append operation of the data block is equivalent to performing an insert operation after the last block of the file.
- **Block Deletion:** If the data owner hopes to remove the block at position j from all replicas, and he deletes the entry at position j from the DMVT. Meanwhile the number of elements in L_i decreases one. He sends the deletion request $< F_{id}, BD, j, null, null >$ to the CSP.

Upon receiving the deletion request, the CSP deletes the blocks $\{\hat{b}_{ij}\}_{1\le i\le m}$ and T_j from T. The CSP outputs the new file replicas $\hat{F}' = \{\hat{F}'_i\}_{1\le i\le m}$ and a new set $T' = \{T_1, T_2, ..., T_{j-1}, T_{j+1}, ..., T_{n-1}\}$.

Figure 3 shows the changes of the DMVTs for different dynamic operations. The file F is divided into 15 blocks, and 3 DMVTs are used to support dynamic operations. As shown in Figure 3(a), the initial value of Vi is 1. As shown in Figure 3(b), when the data owner modifies f[9], $V9$ is incremented by 1. To insert a new block after f[9], Figure 3c shows that a new entry $< 16, 1 >$ is inserted after the position 9, where 16 is the logical index the new inserted block, and 1 is the version number of the new inserted block. As shown in Figure 3(d), the append operation of the block is equivalent to performing an insert operation after the last block of the file. Deleting a block f[3] requires deleting the table entry $< L3, V3 >$ and shifting all subsequent entries one position up (as shown in Figure 3(e)).

Challenge:

1) The data owner chooses a random value c as the number of the challenged blocks.
2) And then picks two random numbers $k_1 \underleftarrow{R} Z_q^*, k_2 \underleftarrow{R} Z_q^*$.
3) Finally, the data owner sends the challenge $chal = (c, k_1, k_2)$to the CSP.

ProofGen:

1) After receiving the challenge from the data owner, the CSP computes $l_t = \pi_{k_1}(t)$ and $a_t = \phi_{k_2}(t)$ for $1 \le t \le c$.
2) And then the CSP computes $\mu = \sum_{t=1}^{c} T_{l_t} \oplus C_{l_t}$ and $\sigma_k = \sum_{t=1}^{c} \hat{b}_{i,l_t,k}$ for $1 \le k \le s$.
3) Finally, the CSP returns a proof (σ, μ).

ProofVerify: Upon receiving the proof from the CSP, the data owner does as follows:

Firstly, computes $l_t = \pi_{k_1}(t), a_t = \phi_{k_2}(t)$ for $1 \le t \le c$. And then checks whether the following verification equation holds or not.

$$S_\alpha(\sigma) \stackrel{?}{=} \mu.$$

If this equation holds, it means that the CSP properly stores all the replicas of the file. Otherwise, not.

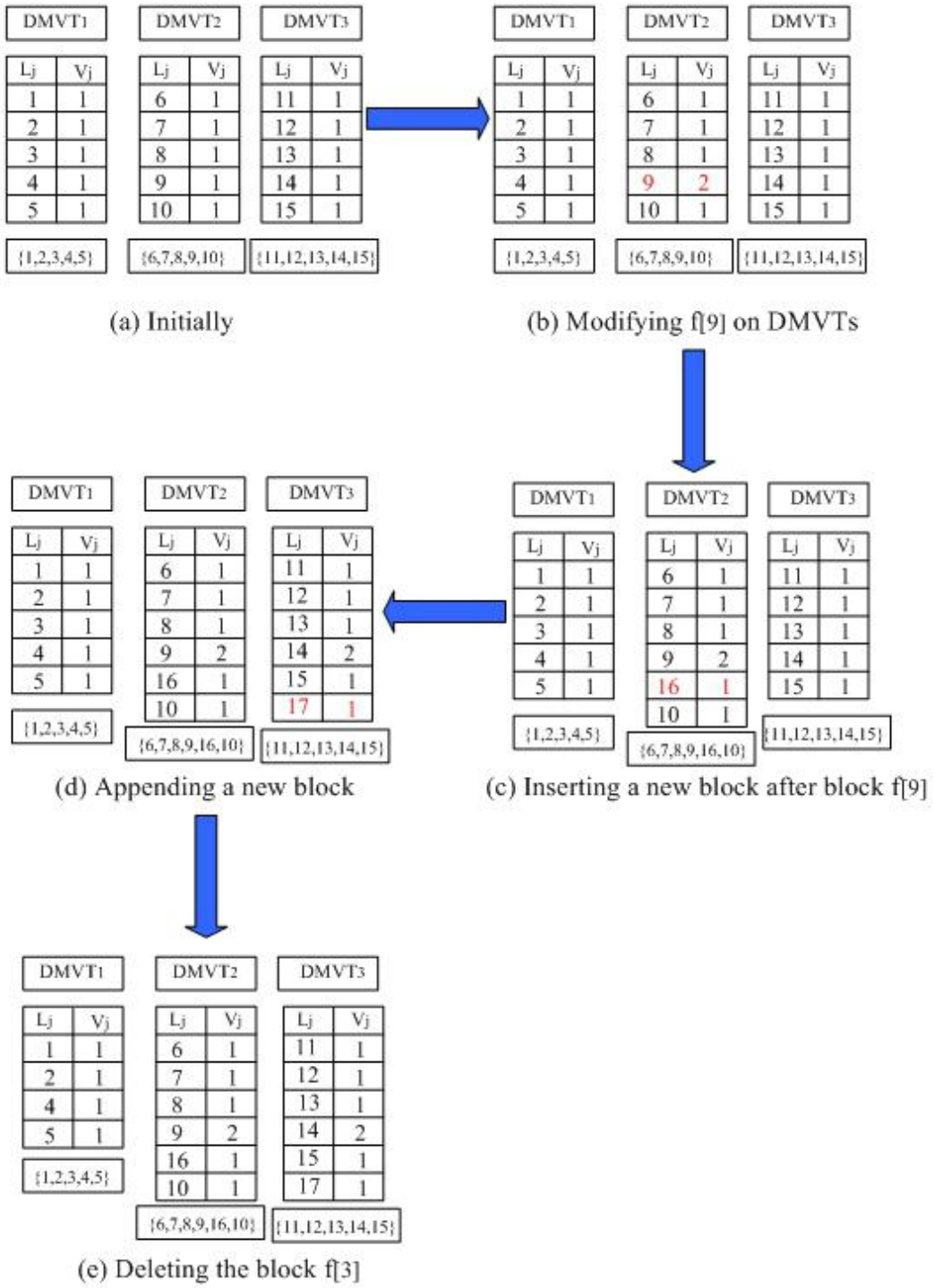

Figure 3: The changes of the DMVTs for different dynamic operations

5 The Security and Performance Analysis

5.1 The Correctness and Security Analysis

5.1.1 The Correctness Analysis

We analyze the correctness and the security of the proposed scheme. In the ProofVerify stage, we firstly extend by using the properties of the algebraic signature as follows:

$$\begin{aligned}\mu &= \sum_{t=1}^{c} T_j \oplus C_j \\ &= \sum_{t=1}^{c}\sum_{k=1}^{s}(S_\alpha(\hat{b}_{ijk}||F_{id}||j||L_j||V_j) \oplus S_\alpha(F_{id}||j||L_j||V_j)) \\ &= \sum_{t=1}^{c}\sum_{k=1}^{s} S_\alpha(\hat{b}_{ijk}) \oplus l^\alpha \oplus S_\alpha(F_{id}||j||L_j||V_j) \\ &\quad \oplus S_\alpha(F_{id}||j||L_j||V_j)) \\ &= \sum_{t=1}^{c}\sum_{k=1}^{s} S_\alpha(\hat{b}_{ijk}).\end{aligned} \tag{1}$$

And then we demonstrate the correctness of the above verification equation as follows:

$$\begin{aligned}S_\alpha(\sigma) &= S_\alpha(\sum_{t=1}^{c}\sum_{k=1}^{s}\hat{b}_{ijk}) \\ &= \sum_{t=1}^{c} S_\alpha(\sum_{k=1}^{s}\hat{b}_{ijk}) \\ &= \sum_{t=1}^{c}\sum_{k=1}^{s} S_\alpha(\hat{b}_{ijk}) \\ &= \mu.\end{aligned}$$

5.1.2 The Security Analysis

Theorem 1. *(The auditing soundness)* In the proposed scheme, the cloud can pass the verification only if it actually stores intact data.

Proof. If the cloud passes the verification but does not possess the intact data, it means that the cloud can forge the valid algebraic signature for any message. Algebraic signature condenses a large block into a bit string. The bit string can be made long enough to make an accidental almost impossible to happen. For example, a 64 bits signature will suffer a collision with probability 2^{-64} and a 256 bits signature with probability 2^{-256}. It is probabilistically impossible for a site that does not know any secret to generate a coherent set of signatures. As a result, the algebraic signature is secure enough for checking the integrity of multiple replicas. □

Theorem 2. *(The resisting collusion attack of cloud servers)* In the proposed scheme, the cloud cannot make data owner believe that they truly stored all replicas, but in real they only save one replica.

Proof. In our scheme, the data owner creates m differentiable replicas $\hat{F} = \{\hat{F}_i\}_{1\le i\le m}$ that are stored on m cloud servers. The cloud sever can not know the content of replicas stored on other cloud servers. The proof (σ, μ) generated by the CSP will be valid and will pass the verification equation $S_\alpha(\sigma) \stackrel{?}{=} \mu$ only if all copies are intact. Thus, when there is one or more corrupted copies, the whole auditing procedure fails. So, the cloud cannot make data owner believe that they truly stored all replicas, but in real they only save one replica. □

Theorem 3. *(Detectability)* Our proposed auditing scheme is $(\frac{m}{n}, 1-(\frac{n-1}{n})^c)$ detectable if the cloud stores a file with n blocks including m bad (deleted or modified) blocks, and c blocks are challenged.

Proof. Assume that the cloud stores a file with total n blocks including m bad (deleted or modified) blocks. The number of challenged blocks is c. Thus, the bad blocks can be found out if and only if at least one of the challenged blocks chosen by the verifier matches the bad blocks. We use a discrete random variable X to denote the number of blocks selected by the challenger that matches the block-tag pairs changed by the adversary. We use PX to denote the probability that at least one block chosen by the challenger matches the blocks changed by the adversary. So

$$\begin{aligned}P_X &= P\{X \ge 1\} \\ &= 1 - P\{X = 0\} \\ &= 1 - \frac{n-m}{n}\frac{n-1-m}{n-1} \times ... \times \frac{n-c+1-m}{n-c+1}.\end{aligned}$$

We can get $P_X \ge 1-(\frac{n-m}{n})^c$. Thus, the proposed auditing scheme is $(\frac{m}{n}, 1-(\frac{n-m}{n})^c)$ detectable if the cloud stores a file with n blocks including m bad (deleted or modified) blocks, and c blocks are challenged. □

5.2 Performance Analysis

Compared with schemes [2,6,13,25], the proposed scheme is more efficient and has two advantages. In the following paragraphs, we will thoroughly explain why our scheme has two advantages over these schemes.

1) In this work, the computation overhead on the data owner is greatly reduced. In schemes [2, 6, 13, 25], the data owner uses BLS signature to generate a homomorphism authenticator for each block of each replica. In the process of calculating homomorphic authenticator, there are a lot of modular multiplication operations. This incurs high computation overhead for the data owner. So the data owner needs to

be powerful enough to perform these costly computations when the data are outsourced. However, in the real world, the data owner (*eg*, using PDAs and mobile phones) may possess low computation capabilities. Observing this fact, we select to use the algebraic signature technique to generate authenticator for each block of each replica in this paper. Because most operations of algebraic signature are XOR operations, the computation overhead on the data owner is minimal.

2) Our scheme efficiently supports full dynamic data operations. As shown in Figure 5, our proposed scheme is more efficient than the scheme [2] when the data owner frequently performs data update. In the scheme [2], the computation overhead during the insert and delete operations is O(n), where n is the number of the file blocks. In our scheme, the data owner only needs to shift a part of the outsourced data blocks ($\frac{n}{k} - i$) that incurs only O($\frac{n}{k}$) computation overhead on the data owner side when he inserts or deletes a data block. In the scheme [13], the data owner uses fully homomorphic encryption algorithm to generate multiple copies and to support data block dynamic operations. We know that fully homomorphic encryption algorithm can incur heavy computing burden on data owner and is inefficiency according to the scheme [4]. So, our proposed scheme is more efficient than the scheme [13]. The scheme [25] supports batch verification based on identity but not allows data owner stores multiple replicas on the cloud and not supports data block dynamic operations. In the scheme [6], the cloud uses skip list to support data block dynamic operations and the computation overhead during the insert and delete operations is O(n). Moreover, in this scheme, data file F is split into blocks, and each data block is split into sectors. In the scheme [6], data file F is split into blocks, but the data block is not split into sectors. In fact, the fragment operation of data blocks can reduce the number of data authenticators. Obviously, our scheme is more efficient than the scheme [6].

Here we analyze the storage and communication overhead of our proposed scheme. For a concrete example of using our scheme, we consider an algebraic signature is 256 bits, a 102MB file F is divided into 125,00 blocks (each block is 8KB) and the file F has 10 replicas. In the Setup stage, the data owner only needs to store one secret key k (160 bits). During the *TagBlock* stage, the data owner stores the file and its tags on cloud servers. The additional storage is less than 4MB. In the *challenge* phase, the data owner sends challenge message to server, and the size of this message is about 480 bits. If the server deletes at least 1% of F, the data owner can detect server misbehavior with probability over 99% by asking proof for 460 blocks. The response of server is less than 8KB.

5.2.1 Experiment Results

With the help of Pairing-Based Cryptography (PBC) library [16], we evaluate the proposed scheme in several experiments. We conduct these experiments on a Linux server with Intel processor running at 2.70 GHz and 4 GB memory. We choose a bilinear map that uses a supersingular curve to achieve the fast pairing operations. Therefore, the base field is 160bits, the size of an element in Z_q^* is 20 bytes, and the size of an element is 128 bytes. In our experiments, the file is set to 102MB consisting of 125,00 blocks, and has 10 replicas.

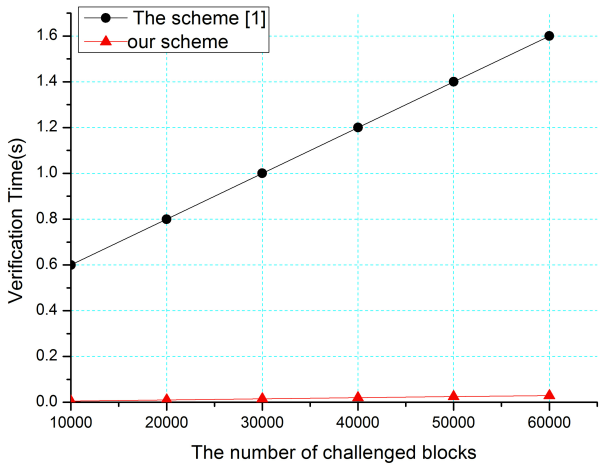

Figure 4: Computation overhead of data owner in verification phase

In our scheme, the most of operations in verification phase are XOR operations. In scheme [2], the most of operations are multiplication operations. We show these two schemes verification time with different number of the challenged blocks in Figure 4. We can see that the verification time in scheme [2] is about 0.6s with the 10,000 challenged blocks. In contrast, the verification time in our scheme is only about 5ms with the 10,000 challenged blocks. Therefore, the verification time in our scheme is remarkable efficient than that in scheme [2].

In Figure 5, we demonstrate the efficiency of the scheme [2] and our scheme when the data owner frequently performs data update. In our scheme, 10 DMVTs are used to support insert or delete operation. In the experiment, we consider computation time of inserting or deleting a block(i) with the number of updated blocks increasing from 100 to 1000. When insert or delete a block(i) in the scheme [2], the data owner firstly looks for the precise position of the block(i); and then shifts (n-i) blocks. This process will incur high computation cost, when the data owner frequently performs data update. Our scheme overcomes this weakness because 10 DMVTs

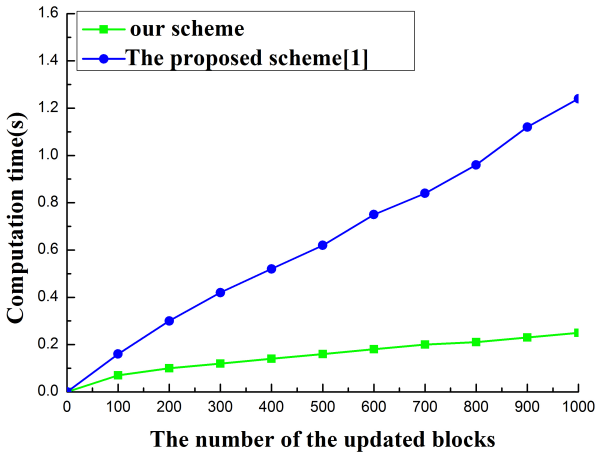

Figure 5: Comparison of computation cost of frequent data update

are used to enhance the efficiency. When the data owner frequently performs data update, our proposed scheme is more efficient than the scheme [2].

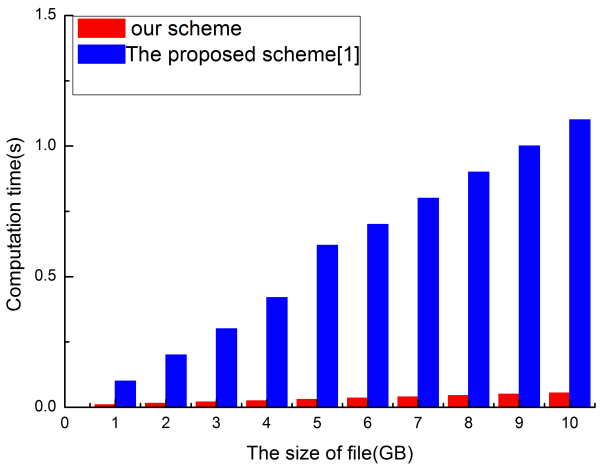

Figure 6: Comparison of computation cost when number of update requests is 100

In Figure 6, we show the computation cost of dynamic data update with the file size from 1GB to 10GB. Assume that 100 blocks are inserted or deleted. In scheme [2], the data owner needs to shift a large number of data blocks, so it will incur high computation overhead. When the data size increases from 1GB to 10GB, the computation time increases from 0.1s to 1.1s. In contrast, our scheme can remarkably reduce the computation overhead. For a 10GB file, the computation time is only 0.05s in our scheme. Therefore, our scheme is very efficient for large-scale files.

As shown in Table 1, the storage space of file copies

Table 1: Storage and communication overheads in our scheme and schemes [2, 3, 26]

Costs		Our Scheme	The Scheme[1]	The Scheme [2]	The Scheme[17]
Storage	File Copies	$m\|F\|$	$m\|F\|$	$m\|F\|$	$m\|F\|$
	CSP Overhead	$m\|F\|+2n\times256$ bits	$m\|F\|+(n+s)\times256+64$ bits	$m\|F\|+(n+s)\times256+64$ bits	$(4m+1)\times n\times256$ bits
	Verifier Overhead	2×256 bits	2×256 bits	2×256 bits	2×256 bits
Communi-cation	Challenge	$256+\log_2(c)$ bits	$256+\log_2(c)$ bits	$256\times\log_2(c)$ bits	512 bits
	Response	2×256 bits	$257+256sm$ bits	$257+256sm$ bits	$256+1024$ bits

($|F|$ is the size of the file F; m is the number of file copies; n is the number of data blocks; s is the number of sectors of one data block; c is the number of the challenged blocks.)

in our scheme is equal to that in schemes [2, 3, 26]. The CSP overheads in both our scheme and schemes [2, 3, 13] are linear in n. The verifier overheads and the size of the challenge message in our scheme are equal to those in the scheme [2, 3]. Especially, the size of response message in our scheme is almost constant and far less than that in schemes [2, 3]. Compared with the scheme [13], the size of challenge message and response message are less than that in scheme [13].

6 Conclusion

In this paper, we propose a provable multiple-replica dynamic data possession for big data storage in cloud computing. In our scheme, we use the algebraic signature to reduce the computation and communication overhead on the data owner side. Meanwhile, in order to achieve efficient dynamic operation, we design a new data structure DMVT. The experimental results demonstrate that our scheme is efficient.

Acknowledgments

This research is supported by National Natural Science Foundation of China (61572267, 61272425, 61402245), the Open Project of the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences(2017-MS-21, 2016-MS-23).

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable data possession at untrusted stored", in *Proceeding of ACM CCS*, pp. 598-609, 2007.

[2] A. F. Barsoum and M. A. Hasan, "Provable Multireplica Dynamic Data Possession in Cloud Computing Systems," *IEEE Transactions on Information Forensics and Security*, pp. 485-497, 2015.

[3] A. F. Barsoum and M. A. Hasan, *Provable Possession and Replication of Data over Cloud Servers*, Cryptographic Research, The University of Waterloo, USA, 2010.

[4] G. Craig, "Computing on encrypted data," *Lecture Notes in Computer Science*, vol. 5888, pp. 477-477, 2009.

[5] R. Curtmola, O. Khan, R. Burns and G. Ateniese, "MR-PDP: Multiple-replica provable data possession," in *Proceedings of the 28th International Conference on Distributed Computing Systems*, pp. 411-420, 2008.

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

[7] Z. Hao and N. Yu, "A multiple-replica remote data possession checking protocol with public verifiability," in *Proceedings of 2nd International Symposium Data*, Privacy, E-Commerce, Sep., pp. 84-89, 2010.

[8] W. Hsien, C. Yang and M. S. Hwang, "A Survey of Public Auditing for Secure Data Storage in Cloud Computing," *International Journal of Network Security*, vol. 18, no. 1, pp. 133-142, 2016.

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

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

[11] 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 *Proceeding of 16th IEEE International Conference Computational Science and Engineering (CSE'13)*, pp. 1128-1135, 2013.

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

[13] M. Li, L. Wang and J. Wei, "Distributed data possession provable in cloud," *Distributed Parallel Databases*, vol. 35, pp. 1-21, 2017.

[14] W. Litwin and T. Schwarz, "Algebraic signatures for scalable distributed data structures," in *Twentieth IEEE International Conference on Data Engineering*, pp. 412-423, 2004.

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

[16] Pairing-Based Cryptography(PBC) library, Feb. 9, 2018. (`https://crypto.stanford.edu/pbc/howto.html`)

[17] W. Shen, J. Yu, R. Hao and X. Wang, "A public cloud storage auditing scheme with lightweight authenticator generation," in *IEEE International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 36-39, 2015.

[18] C. Wang, Q. Wang, K. Ren and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *The Proceedings of IEEE INFOCOM'10*, pp. 525-533, 2010.

[19] C. Wang, S. 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.

[20] H. Wang, D. He and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1165-1176, 2016.

[21] G. Yang, J. Yu, W. Shen, Q. Su, F. Zhang and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *Journal of Systems and Software*, vol. 113, pp. 130-139, 2016.

[22] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1362-1375, 2016.

[23] J. Yu, K. Ren, C. Wang and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1167-1179, 2015.

[24] J. Yuan and S. Yu, "Efficient public integrity checking for cloud data sharing with multi-user modification," in *Proceedings of IEEE INFOCOM*, pp. 2121-2129, 2014.

[25] F. Zhou, S. Peng, J. Xu and Z. Xu, "Identity-based batch probable data possession," in *International Conference on Provable Security*, pp. 112-129, 2016.

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

[27] Y. Zhang, J. Ni, X. Tao, Y. Wang and Y. Yu, "Provable multiple replication data possession with full dynamics for secure cloud storage," *Concurrency Computation*, vol. 28, no. 4, pp. 1161-1173, 2016.

Biography

HuiYing Hou received B.S. degrees in School of Computer Science and Technology from Qingdao University, China, in 2013. She will receive M.S. degree in the college of Computer Science and Technology from Qingdao University, China, in 2016. Her research is cloud computing security.

Rong Hao received Master degree in Institute of Network Security from Shandong University. She is working in the College of Computer Science and Technology, Qingdao University. Her research interest is information security.

Jia Yu received the M.S. and B.S. degrees in School of Computer Science and Technology from Shandong University, China, in 2003 and 2000, respectively. He received Ph. D. degree in Institute of Network Security from Shandong University, China, in 2006. Since 2012, he has been a full professor and the department director of information security, at Qingdao University, China. He was a visiting professor with the Department of Computer Science and Engineering, the State University of New York at Buffalo from 2013 to 2014. His research interests include cloud computing security, key evolving cryptography, digital signature, and network security. He has published over 100 academic papers. He is the reviewer of more than 30 international academic journals.