Secure and Efficient Cloud Data Deduplication Supporting Dynamic Data Public Auditing

Hua Ma, Xiaoyu Han, Ting Peng and Linchao Zhang
(Corresponding author: Xiaoyu Han)

School of Mathematics and Statistics, Xidian University
No. 2, South Taibai Road, Xi'an 710071, P.R. China
(Email: xyhan0723@163.com)

(Received May 27, 2017; revised and accepted Oct. 21, 2017)

Abstract

Secure data deduplication, which can reduce the amount of storage cost in the cloud by eliminating duplicate data copies, has been widely used in industry and academia. At the same time, as the outsourced cloud storage server is not fully credible, it will cause data destruction when the user puts the the encrypted data in the cloud. Therefore, we present a secure and efficient cloud data deduplication scheme supporting dynamic data public auditing. Compared with the prior systems, our system has two advantages. Firstly, the proposed scheme has a high performance in terms of data equality test by using the decision tree, as the server can reduce the time complexity of deduplication equality test from linear to logarithmic over the whole data items in the database. Secondly, the proposed scheme can reduce the computation cost of searching a data block from linear to logarithmic by using relative index of a node and also support data modification, insertion, append and deletion procedure.

Keywords: Cloud Storage; Deduplication; Public Auditing

1 Introduction

Cloud computing is a pay-per-use model that provides available, convenient, and on-demand network access to configurable computing resource sharing pools (resources including networks, servers, storage, applications and services). These resources can be quickly provided with very little management effort or little interaction among the service provider.

Among them, cloud storage is a networked storage model that data is stored in a virtual storage pool, usually maintained by the third party. Cloud storage provides customers with a lot of benefits, including cost savings and simplified convenience, and so on. These benefits make more and more customers store their personal data in the cloud. The analysis shows that the amount of data will reach 40 trillion gigabytes in 2020, of which 70% are stored in the cloud. Among these remote stored files, most of them are identical. Data deduplication, a special data compression method, has been widely used in cloud to optimize storage space by reducing the amount of data storage [5, 14, 20]. Nevertheless, the problem, that the same file encrypted by different users lead to different ciphertexts, makes cross-user deduplication impossible. The convergence encryption(CE) can implement this, because the key is derived from the file, regardless of who performs the encryption algorithm. However, existing solutions in use unfortunately have either security or privacy issues. Such as offline dictionary attacks as the keys are derived deterministically from the file F ($i.e., K_A(= K_B) = H(F)$), where H is a conventional hash function used to map any length of messages into a fixed-length value.

At the same time, the users put the data in the cloud and need to consider the data security. Because the Cloud Service Provider (CSP) may take the initiative to delete the data that the client does not frequently access, so it is necessary for users to ensure the integrity of data in the cloud. In other words, any change about the data in the cloud without the consent of the user, such as data loss, corruption, modification, or disclosure, should be detected by the user. However, as the user's computing power is limited, the user decides to authorize the task of auditing to a Third Party Auditor (TPA), which enables that the user can efficiently perform integrity verifications even without the local copy of data. TPA replaces the user to perform the integrity of data and do not need to download the entire file.

We also consider that the user may want the cloud to protect their data to prevent unauthorized users from accessing. So we use a key distribution mechanism to process the convergence key [16, 26].

In this paper, we propose a secure and efficient cloud data deduplication scheme supporting public auditing. Our main contributions can be summarized as follows.

- A secure and efficient cloud data deduplication scheme is achieved [13, 24], and can reduce the comparison times of the equality-testing algorithm from linear to nearly logarithmic.

- An auditing entity can help clients audit the integrity of data stored in the cloud even without the local copy of data files [9]. At the same time, the computational cost of searching the data block can reduce efficiently from linear to logarithmic. In addition, the scheme supports the dynamic operation of data.

- A scheme of data deduplication by using a key encapsulation mechanism [16, 26], which means that the key of the file is re-encrypted with the attribute, and the user can decrypt the ciphertext only if the user has a private key associated with the attribute.

In Section 2, we give the relevant knowledge of deduplication and auditing. In Section 3, we give the bilinear pairings, computational Diffie-Hellman problem, Path-PRV-CDA2, modified Merkle Hash Tree, Boneh-Lynn-Shacham signature and equality test over deduplication decision tree. Section 4 gives system structure and the threat model. Our detailed scheme is shown in Section 5. The dynamic operation of the modified $\mathcal{MHT}$ tree will be showed in Section 6. The Section 7 gives the security. The Section 8 gives the performance analysis. We make a conclusion in Section 9.

2 Related Works

In this section, we will elaborate on the work related to our scheme from two aspects.

2.1 Secure Deduplication

Convergent encryption, which can not only guarantee the confidentiality of data, but also achieve data deduplication. Specifically, each data file will be encrypted with a so-called convergence key that derived from the cryptographic hash value of the file contents. This gives rise to the same data file to be encrypted to get the same ciphertext, which makes it possible to reduce the redundant data. Bellare *et al.* [5] proposed a novel cryptographic primitive called Message-Locked Encryption (MLE), where the formal definition and security model of deduplication is given. However, both of them do not meet semantic security because of content-guessing attack. If the adversary wants to specify a distribution of plaintexts, the tag that is deterministic from the message will leak unnecessary information. To strengthen the security, Abadi *et al.* [1] proposed a modified primitives, named random Message-Locked Encryption (R-MLE) which can support an equality-testing algorithm defined on the ciphertexts. However, the equality-testing algorithm is very inefficient, which is the only problem. To enhance the security of deduplication and protect the data confidentiality, Bellare *et al.* [4] proposed an improved deduplication scheme, which can resist the online brute-force attack by running an oblivious pseudo-random function (OPRF). In the scheme, users generate the key by the aid of a secret parameter which is produced by a third-part key server. To prevent the data from being leaked to the third-part server, Bellare and Keelveedhi extended their prior work and proposed a new primitive named Interactive Message-Locked Encryption (iMLE) [3]. In their model, security depends on whether the message is related to the public parameter or not. After that, Jiang *et al.* [13] introduced a new primitive called μR-MLE2, which generates a key that does not depend on the third party, and can resist content-guessing attack. The most important is that the time of deduplication test is reduced from linear to logarithmic.

2.2 Integrity Auditing

Ateniese *et al.* [2] put forward the concept of provable data possession (PDP), which makes sure that the cloud server owns indeed the data and does not need to download all the data. PDP implements mainly through randomly extracting a set of data blocks from the cloud server. Zhang *et al.* [28] proposed a secure provable data possession scheme, which can resist forgery attack and replay attack, under the Chosen-Target-CDH problem and the CDH problem. Proof of retrievability (POR) [19] is also a part of the data integrity verification. Compared with PDP, POR can also achieve the recovery of data. Ling *et al.* [17] proposed an efficient and secure one time password authentication scheme for wireless sensor networks. Xu and Chang [25] recommended to use the polynomial commitment to reduce the communication cost of [19]. Stefanov *et al.* [21] made a protocol to the frequent changes in the authentication file system. Hsien *et al.* [10] presented a survey of data integrity based on public auditability and provided the approach to analyze security and efficiency. Hwang *et al.* [11] proposed a scheme with a mechanism to locate the problematic data blocks when the cloud data as a whole fails the auditing. Ming *et al.* [27] implemented a more secure and efficient auditing mechanism in the cloud. Cao *et al.* [8] showed that there are two flaws in one scheme for cloud storage auditing with verifiable outsourcing of key updates. However, these protocols only adapt to the static operation of the data and do not support the dynamic one. Liu *et al.* [18] considered dynamic data update when the data is shared with multiple users, but only described the inserted operation. The scheme [12, 23] supports the dynamic operation of the data, the time of integrity verification is $O(m)$. The computational complexity of integrity verification is proportional to the amount of data in the cloud. When the amount of the data in the cloud increases, the computational complexity will increase linearly. Garg *et al.* [9] improved it by using Relative Index and Time Stamped Merkle Hash Tree ($\mathcal{RITS}--\mathcal{MHT}$), which can guarantee that it can not only support the dynamic operation of the data, but the computation cost of searching the node can reduce from linear to sub-linear.

3 Preliminaries

In this section, we give bilinear groups and computational Diffie-Hellman problem, Path-PRV-CDA2. At the same time, in order to achieve the integrity of the data, we give the modified Merkle Hash Tree ($\mathcal{MHT}$), Boneh-Lynn-Shacham ($\mathcal{BLS}$) signature and equality test over deduplication decision tree.

3.1 Bilinear Pairings

Suppose that G and G_T are the multiplication cycle groups of prime order p. When the map $e : G \times G \rightarrow G_T$ satisfies the following properties, we say that e is a bilinear map [6, 22]:

1) **Bilinearity**: For all $u, v \in G, a, b \in Z_p$, we can deduce that $e(u^a, v^b) = e(u, v)^{ab}$;

2) **Non-degeneracy**: $e(g, g) \neq 1$;

3) **Computability**: For all $u, v \in G$, we can compute $e(u, v)$ in an efficient algorithm.

3.2 Computational Diffie-Hellman Problem

The Computational Diffie-Hellman problem (CDH) [6] is that, given $g, g^a, g^b \in G$, it is difficult to compute g^{ab}, where a, b is the random number in Z_p^* and g is the generator of G.

3.3 Path-PRV-CDA2

In the decision tree, the user interacts with the server, calculates 1 bit path decision and sends it to the server. The definition of Path-PRV-CDA2 [13] is based on the definition of PRV-CDA2 [1].

3.4 Modified Merkle Hash Tree

Modified $\mathcal{MHT}$ [9] is a binary search tree, where each node stores two parts of information, one is the hash value of the data block and the other is the relative index of the node. Apart from the leaf nodes, the hash value of each node is the concatenation of the hash value of its child nodes. By verifying the hash value of root node, we can assure the integrity of data. In modified $\mathcal{MHT}$, the relative index of leaf nodes is set to 1. Suppose that the index value of a node's two child nodes is r_a and r_b, then its index value is $r_a + r_b$.

Figure 1 shows a modified $\mathcal{MHT}$ with 8 leaf nodes. For example, if the auditor wants the cloud server to prove the integrity of the data block $d[3]$, the cloud server will give the auditor the Auxiliary Information (AI) as $AI(d[3])$:

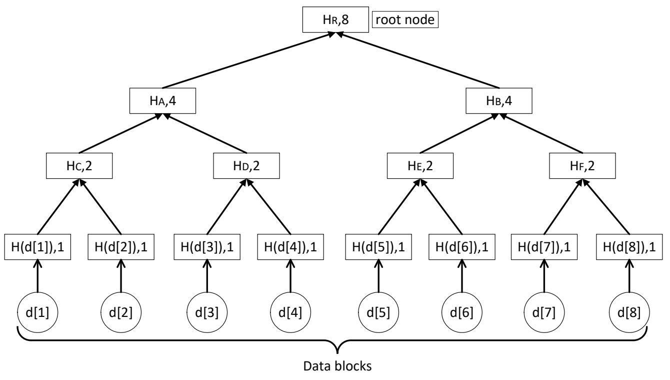


Figure 1: Modified merkle hash tree ($\mathcal{MHT}$)

$\{(H_C, L), H(d[3]), (H(d[4]), R), (H_B, R)\}$. Now the auditor verifies the integrity by using the root node H_R as follows:

Compute $H_D \leftarrow (H(d[3]) \parallel H(d[4]))$.

Compute $H_A \leftarrow (H_C \parallel H_D)$.

Finally compute root $H_R \leftarrow (H_A \parallel H_B)$.

3.5 Boneh-Lynn-Shacham Signature

Boneh-Lynn-Shacham Signature ($\mathcal{BLS}$) [7] is a short signature scheme, which is used for authentication the signer of a message. Its security is based on the CDH problem. $\mathcal{BLS}$ signature can aggregate multiple signatures about many blocks of message into a single signature. Therefore, $\mathcal{BLS}$ signature of n data blocks $(d[1], d[2], \cdots, d[n])$ in file F are generated as follows:

$$\phi = \prod_{i=1}^{n} (H(d[i]))^k$$

$H : \{0, 1\}^* \rightarrow G$ is a hash function and G is a prime order group having generator g. The $\mathcal{BLS}$ signature specifically contains three parts in the following:

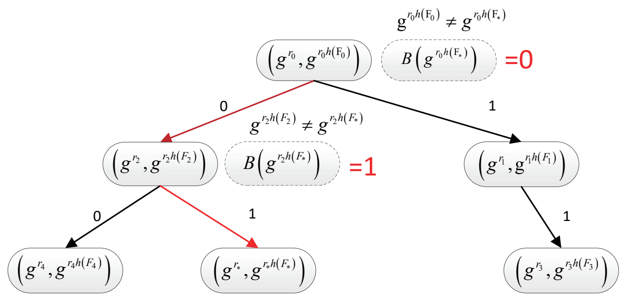
Let $k \in Z_p$ be the private key and g^k is the public key.

Generate a signature ϕ for file F: $\phi \rightarrow H(F)^k$.

Input the file F and signature ϕ, verify $e(\phi, g) \overset{?}{=} e(H(F), g^k)$.

3.6 Equality Test Over Deduplication Decision Tree

Jiang *et al.* [13] proposed an interactive deduplication decision tree in Figure 2. Generally speaking, the user requests the server to return the tag of current node. The user checks whether there is a copy, if it is, users just get a link related to file from the cloud without uploading the encrypted data. Otherwise, the user uploads the data to the cloud. In general, give $h(F)$, some relevant information and 1-bit path decision $b = B(g^{r_i \cdot h(F)})(B(x)$ denotes



$g^{r_0 h(F_0)} \neq g^{r_0 h(F_*)}$
$(g^{r_0}, g^{r_0 h(F_0)})$
$B(g^{r_0 h(F_*)})$
=0
0
1
$g^{r_2 h(F_2)} \neq g^{r_2 h(F_*)}$
$(g^{r_2}, g^{r_2 h(F_2)})$
$B(g^{r_2 h(F_*)})$
=1
$(g^{r_1}, g^{r_1 h(F_1)})$
0
1
1
$(g^{r_4}, g^{r_4 h(F_4)})$
$(g^{r_*}, g^{r_* h(F_*)})$
$(g^{r_3}, g^{r_3 h(F_3)})$

Figure 2: Deduplication decision tree

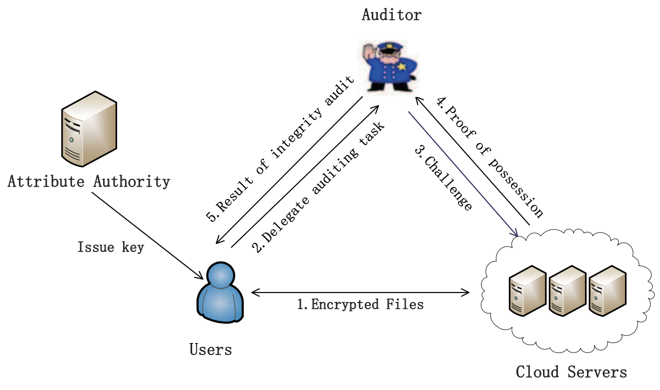


Figure 3: The system model

the bitwise exclusive or of the digest of x). If $b = 0$, the server moves it to the left child node. Else if $b = 1$, the server moves it to the right child node. The algorithm can be described as follows.

- User $\leftrightarrow$ Server: The user owns file F_* and requests the cloud server to return the tag τ_i of the current node, $\tau_i = (g^{r_i}, g^{r_i h(F_i)})$. (In general, if the node is the root one in the tree, and the corresponding tag is $\tau = (g^{r_0}, g^{r_0 h(F_0)})$.
- User: The user detects whether the cloud has the same one as himself. That is to say, the user verifies whether the equation $g^{r_i h(F_*)} \overset{?}{=} g^{r_i h(F_i)}$ holds.
- User $\rightarrow$ Server: When the equation is established, it means that the user finds a duplication. Otherwise, he calculates $b = B(g^{r_i h(F_*)}) \in \{0, 1\}$ and gives it to the server.
- Server: When the server gets 0, the user moves the node to the left child. Otherwise, the node is transferred to the right child. The user then continues to interact with the server and finds that one of the following cases occurs. One is that, when the server traverses the entire tree, it does not find the same data as the user. The other is that, the server finds a copy of the data.

4 Problem Formulation

4.1 System Architecture

In our system, we include four entities: the user, Cloud Service Provider (CSP), Third Party Auditor (TPA) and attribute authority (AA). The relationship among them is shown in Figure 3.

- The user has a lot of files and will generally put them in the cloud to save their own storage and reduce the computing cost.
- Cloud Service Provider (CSP) is a system that provides file storage and business access capabilities truthfully. Users often pay the cloud to enjoy this service.
- The Third Party Auditor (TPA) helps the user detect the integrity of the outsourced file in the cloud, since the cloud may delete files that is rarely used by the user or that has been corrupted to save storage.
- The attribute authority (AA) is an entity which can send the user a key related to the user's attribute to encapsulate the encryption key.

Our system contains two phases: data deduplication phase and public auditing phase.

1) Data deduplication phase: At this stage, the user detects whether CSP has the same file. We mainly have two situations to consider.
    - When CSP has already stored the data, users do not have to upload the file and only obtain a link to the file.
    - The user uses the message locked encryption to encrypt the file and uploads it to the cloud. After that AA employs the key encapsulation mechanism to encapsulate the key.

2) Public auditing phase: We introduce it mainly in four parts. They are BlockSigGen, Challenge, GenProof and VerifyProof, respectively.
    - **Keygen(1^λ)**: The user inputs a security parameter λ and outputs key pair (k, g^k). Here, k is the private key, g^k is the public key.
    - **BlockSigGen($k, H(ct_{d(i)}), u$)**: The user inputs k, the hash value of encrypted file block $H(ct_{d(i)})$, a random number u, and outputs the signature of the block is ϕ_i. The set of signatures for each encrypted block is represented as $\theta = \{\phi_i\}$, $i \in \{1, 2, \cdots, n\}$.
    - **Challenge**: TPA replaces the user to detect the integrity of data in the cloud. TPA takes several blocks of file randomly and sends them to the CSP.

- **GenProof(ct_F, θ, C)**: When CSP received the challenge information C from TPA, CSP inputs encrypted File ct_F, all signature set θ, challenge information C to produces the proof P_f. Finally, CSP sends the proof to TPA.
- **VerifyProof(C, P_f, g^k)**: TPA inputs the challenge Information C, proof P_f, public key g^k, and verifies the results. If it holds, it indicates that the data stored in the cloud is complete. Otherwise, we conclude that the data has been tampered or even lost.

4.2 Threat Model

We consider some threats about the users' data in the following.

- The user authorizes the task of auditing to TPA to ensure the integrity of data in the cloud. Although TPA is credible, he will be curious about the data stored in the cloud.
- CSP may delete data that is occasionally accessed by the user, or CSP may damage the data due to their own processing error without informing the user.
- The previous administrator in the cloud could break into and corrupt the integrity of data. And if the certificate of legitimate user is lost, the other users will use the certificate to destroy the data or even delete the data.

5 Our Detailed Construction

In this section, we construct a secure and efficient cloud data deduplication scheme supporting dynamic data public auditing. Our scheme mainly includes four entities which has been showed in Figure 3. Firstly, AA defines the set of public attributes [15] and sends the user a key related to the attribute. The key which is used to re-encrypt the convergence key to implement the key encapsulation. Then, the user sends a tag to CSP to decide whether the cloud stores the file or not.

5.1 Data Deduplication Phase

This phase intends primarily to detect whether the cloud has the same data by using the equality test over deduplication decision Tree.

- The user has the file F_* and the tag of F_* is $\tau_* = (g^{r_*}, g^{r_* h(F_*)})$. After that, the user requires CSP to return the tag $\tau_i = (g^{r_i}, g^{r_i h(F_i)})$ of current node. The user then detects whether there exists the same file by checking $g^{r_i \cdot h(F_*)} \overset{?}{=} g^{r_i \cdot h(F_i)}$, where r_i and r_* are selected randomly. If the equation is verified to be equal, it means that there is a duplicate one stored in the cloud, it is unnecessary for the user to upload the file.
- Otherwise, the user sends b ($b = B(g^{r_i h(F_*)}) \in \{0, 1\}$) to CSP, if $b = 0$, CSP moves the node to the left, otherwise it moves the node to the right. The algorithm terminates when CSP has traversed the entire tree and can not find the same data. Therefore, the user is authorized to upload the ciphertext to the cloud. Specifically, the user computes $k_F = Hash(pp, F)$, $ct_F = Enc_{pp}(k_F, F)$, and gives ct_F to CSP, where $Enc(\cdot)$ is the symmetric encryption algorithm, pp is the public parameters. The convergent keys k_F is encrypted with the key associated with the attribute. The user can get the convergence keys to decrypt the ciphertext only if he has the attribute.

5.2 Public Auditing Phase

TPA replaces the user to check the integrity of data in the cloud and returns the results to the user. Detailed details are as follows.

- **KeyGen**: The user first randomly selects the private key $k \in Z_p$ and the public key $\eta = g^k$. After that, the user divides the encrypted file into n blocks.
- **BlockSigGen**: Generally speaking, the user generates $\mathcal{BLS}$ signature for each encrypted block $ct_{d[i]}$ of the file is $\phi_i = (H(ct_{d[i]}) \cdot u^{ct_{d[i]}}))^k$, where $ct_{d[i]} = Enc_{pp}(k_{d[i]}, d[i])$, $i \in \{1, 2, \cdots, n\}$ and a random element $u \in G$. Here $H(ct_{d[i]})$ indicates the hash value of the encrypted file block. The user generates a $\mathcal{MHT}$ and a root nodes H_R with $H(ct_{d[i]})$ as a leaf node for all $i \in \{1, n\}$ (H_R is the root node of the encrypted data block). The signature for the root node is expressed as $\rho = H(R)^k$. After that, the user sends $\{ct_F, \theta, \rho\}$ to CSP for storage.
- **Challenge**: In order to ensure that their files are intact, the users upload the files to the cloud and delegate the task of verifying the integrity of data to TPA. Then TPA randomly extracts a portion of the file from CSP, and CSP responds to the extracted data. TPA randomly selects Q and b_i, and sends $C \leftarrow (i, b_i)_{i \in Q}$ to CSP, where $Q = \{r_1, r_2, \cdots, r_k\} \in [1, n]$ and $i \in Q$.
- **GenProof**: When receives the challenge message C, CSP starts to search ith node in the hash tree ($i \in C$). We determine whether the search node exists in the hash tree by validating the root node. For example, we need to search for file block $d[4]$, CSP can use all the auxiliary information on the path of the search node. The auxiliary information(AI) is expressed as: $[(H_C, 2, L), (H(d[3]), 1, L), (H_B, 4, R)]$, where L represents the position of node on the left, R is represented as the position of the node on the right. SIB specifies the number of sibling nodes on the left. When the detection of the node passes, CSP continues to search for the other nodes on the hash tree. Otherwise CSP terminates the auditing and

computes:

$$\phi \leftarrow \prod_{i=r_1}^{r_k} \phi_i^{b_i} \in G.$$

and

$$\mu \leftarrow \prod_{i=r_1}^{r_k} b_i D[i] \in G.$$

After that, P_f is sent to TAP by CSP, $P_f = \{\mu, \phi, (H(ct_{d_i}), AI(i)_{i\in Q}, \rho, SIB)$. Here $AI(i)$ is the auxiliary information of i on the path from the node $H(ct_{d[i]})$ to the root node.

- **VerifyProof**: TPA establishes the root node of the tree and verifies the root node by detecting whether the following equation holds or not:

$$e(H(R), g^k) = e(\rho, g). \qquad (1)$$

Here, $H(R)$ is the hash value of the root of the encrypted data block. Finally, TPA sends the verification results to the user. On authentication verification successes, TPA proceeds the auditing process and verifies whether the following equation holds or not:

$$e(\prod_{i=r_1}^{r_k} H(ct_{d[i]})^{b_i} \cdot u^{\mu}, g^k) = e(\phi, g) \qquad (2)$$

That is to say,

$$\begin{aligned} & e(\prod_{i=r_1}^{r_k} H(ct_{d[i]})^{b_i} \cdot u^{\mu}, g^k) \\ =& e(\prod_{i=r_1}^{r_k} H(ct_{d[i]})^{b_i} \cdot u^{b_i ct_{d[i]}}, g^k) \\ =& e(\prod_{i=r_1}^{r_k} (H(ct_{d[i]}) \cdot u^{ct_{d[i]}})^{b_i}, g^k) \\ =& e(\prod_{i=r_1}^{r_k} ((H(ct_{d[i]}) \cdot u^{ct_{d[i]}})^k)^{b_i}, g^k) \\ =& e(\prod_{i=r_1}^{r_k} \phi_i^{b_i}, g) \end{aligned}$$

If the validation process passes, the file that the user stores in the cloud is complete. Otherwise the user's file is compromised.

6 Data Dynamic Procedures

Data dynamic operation [9] is very important in data auditing, which includes data modification procedure ($\mathcal{DMP}$), data insertion procedure ($\mathcal{DIP}$), data append procedure ($\mathcal{DAP}$) and lastly data deletion procedure($\mathcal{DDP}$). Data dynamic procedures which allow the user to update the selected file when a file block changes.

6.1 Data Modification Procedure

When the user wants to modify the ith file block, the user generates a new signature for this data block as $\phi' = (H(ct_{d[i']}) \cdot u^{ct_{d[i']}})^k$, and produces a new file tag as $\tau' = (g^r, g^{rh(F')})$, and then the user sends $(M, i, ct(d[i']), H(ct_{d[i']}),', \tau')$ to CSP, where M is denoted as $\mathcal{DMP}$. Cloud storage server replaces $ct(d[i])$ with $ct(d[i'])$, replaces ϕ with ϕ', replaces τ with τ', replaces $H(ct_{d[i']})$ with $H(ct_{d[i']})$, and then generates a new hash of the root node $H(R')$ by reconstructing $\mathcal{MHT}$ with ith modified file block. Then the user authenticates the root by using $(H(ct_{d[i']}), AI(i))_i, \rho)$, produces $\rho' = H(R')^k$ with secret key k and sends it to the cloud for storage. In this way, the user completes the modification operation of the data block.

6.2 Data Insertion Procedure

When the user wants to insert the file block, the user first generates a signature for the inserted file block as $\phi^* = (H(ct_{d[i^*]}) \cdot u^{ct_{d[i^*]}})^k$ and computes the tag as $\tau^* = (g^r, g^{rh(F^*)})$. Assume that the insertion file block is $(I, i, ct(d[i^*]), \phi_i, \tau^*)$ and sends them to the CSP, where I is denoted as $\mathcal{DIP}$. When a new node is inserted, the hash value of the node at that location becomes $(H(ct(d[i])) \parallel H(ct(d[i^*])))$ and the index value of it will also change, too. Specific detail is shown in [9]. The user establishes $\mathcal{MHT}$ to authenticate the root node using $AI(I), H(ct(d[i]))$ in Equation (2), if the certification holds, it means that CSP performs the insert operation honestly.

6.3 Data Deletion Procedure

$\mathcal{DDP}$ is similar to $\mathcal{DIP}$. Assuming that we want to delete a data, then the hash value of the node is transferred, the corresponding index of the node changes, too.

6.4 Data Append Procedure

$\mathcal{DAP}$ is similar to the data insertion.

7 Security Analysis

In this part, we will analyze the security of our scheme. The Theorem 1 is reduced to that of Garg and Bawa's scheme [9], which has been proved to be secure in random oracle model. We show that our scheme is Path-PRV-CDA2 secure in Theorem 2. Finally, we give an analysis to show that the auditing is sound in Theorem 3.

Theorem 1. *As Garg and Bawa's scheme [9] is security, then an adversary $\mathcal{C}$ has an negligible advantage ϵ to generate a forgery for a data block $ct_{d[i']}$:*

$$\epsilon' \geq (1/q_S + 1)e)\epsilon$$

Table 1: Functionalities and features comparison of deduplication and auditing

Schemes	Ciphertext deduplication	No additional key server	Auditing	Block dynamic operations
[1]	√	√	×	×
[9]	–	–	√	√
[13]	√	√	×	×
[15]	√	×	√	×
[23]	–	–	√	√
Ours	√	√	√	√

and in polynomial time:

$$t_c \geq t + t_{sm}(q_H + 2q_S)$$

where t_{sm} is time for one scalar multiplication in G and e is base of natural logarithm.

Proof. Assuming the attacker $\mathcal{A}$ has already known (g, g^a, g^b), our purpose is to solve the CDH problem by using a subroutine $\mathcal{B}$, $\mathcal{B}$ is maintained by $\mathcal{A}$.

Parameter Query: The adversary $\mathcal{C}$ makes queries to $\mathcal{B}$ about the relevant system parameters , and $\mathcal{B}$ responds it in the following, $\mathcal{B}$ chooses $r \in Z_p$ as its own private key, generates $g^r \in G$ as the public key, and sends (r, g^r, G) to the adversary, where r keeps secret.

Hash Query: The adversary $\mathcal{C}$ can make a Hash Query, $\mathcal{B}$ responds it as below. The adversary $\mathcal{C}$ also makes a list L_H: $\{ct_{d[i]}, w_i, k_i, c_l\}$, which is initially empty.

- If there has been a query on $ct_{d[i]}$ in the L_H, $\mathcal{B}$ responds with $H(ct_{d[i]}) = w_i$.
- Otherwise, $\mathcal{B}$ flips a coin $c \in \{0, 1\}$, when $c_l = 0$, the probability of it is δ. And when $c_l = 1$, the probability of it is $1 - \delta$.
- $\mathcal{B}$ chooses a element $k_i \in Z_p$ randomly, computes $w_i = (g^b)^{(1-c_l)}\phi(g)^{k_i}$ and puts $\{ct_{d[i]}, w_i, k_i, c_l\}$ to list L_H.

Sign Query: The adversary makes a signature inquiry about the data block $ct_{d[i]}$, and $\mathcal{B}$ replies to him as follows:

- If $c_l = 0$, $\mathcal{B}$ halts.
- Otherwise, if $c_l = 1$, then $\mathcal{B}$ responds with $\sigma_i = (\phi(g^a)^{k_i})(\phi(g)^{k_i r})$. Here $w_i = \phi(g)^{k_i}$, therefore $\sigma_i = w_i^{a+r}$.
- Assume that the adversary $\mathcal{C}$ has created a forged signature σ_k^* for the data block $ct_{d[k]}$, Then if $c_l = 1$, $\mathcal{B}$ halts. when $c_l = 0$, B replies to the adversary with $H(ct_{d[k]}) = g^b\phi(g)^k$ and σ_k^*.

  Then we can derive that:

$$g^{ab} = \sigma_k^*/(g^{rb}\phi(g^{ak})\phi(rk)).$$

Only $\mathcal{B}$ can calculate g^{ab} in the polynomial time.

Next, we show the probability that the CDH problem can be solved by $\mathcal{B}$. $\mathcal{B}$ wants to succeed mainly based on the following three cases:

- E_1: Any sign of an adversary $\mathcal{C}$ does not make $\mathcal{B}$ stop the game;
- E_2: The adversary $\mathcal{C}$ generates a genuine σ_k on data block $ct(d[k])$;
- E_3: The adversary $\mathcal{C}$ produces a forged signature on data block $ct(d[k])$ and $\mathcal{B}$ does not abort.

The probability of B succeed is

$$P[E_1 \cap E_2 \cap E_3] = P[E_1] \cdot P[E_2 \parallel E_1]P[E_3 \parallel E_1 \cap E_2]$$

From Garg and Bawas scheme in [9], we can calculate the probability of B succeed is ϵ'

$$\epsilon' \geq (1/q_S + 1)e)\epsilon.$$

Since our scheme is based on the same difficult problem as Garg and Bawa's scheme, And the advantage of the attacker to break our scheme is the same as the attacker break the Garg and Bawa's scheme. So, our scheme is also secure in CDH problem. □

Theorem 2. *Our μR-MLE2 scheme is Path-PRV-CDA2 [1] secure.*

Proof. Given two sequences $v = (v_1, \cdots, v_n)$ and $v' = (v_1', \cdots, v_n')$, we consider the information that the adversary $\mathcal{C}$ can get through the two sequences. And if the tree path of two files is same, we recognize that they are the same one. We consider it in two cases.

One is when both of them have the same tree path, v is in the deduplication list, v' is also in the deduplication list. Then, the adversary $\mathcal{C}$ can not get any information from the two sequences.

The other is that v and v' have the same order relation. v is in the deduplication list, v' is not in the deduplication list. The user gives the path bit to the server. In both cases, the information obtained by the adversary is same, and the number of bits leaked does not exceed the height of the tree. So it is Path-PRV-CDA2 [1] secure. □

Table 2: Number of challenged blocks up to 90% of corrupted blocks

Number of corrupted blocks(x)	Number of challenged blocks(t)
6250	23
12500	12
18750	10
25000	7
31250	5
37500	4
43750	4
50000	3
56250	2

Theorem 3. *Assuming the hash function is collision-resistant, when clients and CSP follow the Modified Merkle tree-based protocol, a dishonest CSP cannot pass through the verification.*

Proof. Suppose that the user needs to verify the ith file block, the malicious CSP transfers part of the user's data and has two ways to cheat the user. One way is to return a forged message. The server may try to find an $ct_{d_j} \neq ct_{d_i}$ satisfying that $H(ct_{d_j}) = H(ct_{d_i})$. However, under the collision-resistant assumption, CSP cannot generate invalid data and the corresponding integrity path that the client can use to compute the correct metadata $H(R)$. The other way is that CSP may ignore the verification without giving the correct $P_f = \{\mu, \phi, (H(ct_{d_i}), AI(i)_{i\in Q}, \rho, SIB\}$, but to return another valid pair of data $P_{f^*} = \{\mu^*, \phi, (H(ct_{d_j}), AI(j)_{j\in Q}, \rho^*, SIB^*\}$ to the user.

In the phase of VerifyProof, TPA found that

$$e(\prod_{j=r_1}^{r_k} H(ct_{d[j]})^{b_j} \cdot u^{\mu^*}, g^k) \neq e(\phi, g)$$

In other words, any malpractice by CSP will be detected by a fully trusted TPA at the VerifyProof stage. Therefore, based on the Equation (2), CSP cannot use another data path pair to pass through the verification successfully. □

8 Performance Analysis

In this section, we will conduct our performance analysis including three aspects, the probability of malpractice at CSP, the computation cost, and communication overhead.

8.1 Probability of Detecting Malpractice at CSP

The solution [9] is to take the sampling method, that is, it divides the file F into n chunks and then randomly extracts t from n data blocks to detect the probability of detecting malpractice at CSP. Suppose that CSP changes x data block in n data blocks, then the probability of randomly sample a corrupted block is $\frac{x}{n}$. If the probability of detecting malpractice at least one block in challenge phase is P, then

$$\begin{aligned} P &= 1-(1-\frac{x}{n})(1-\frac{x}{n-1})\cdots(1-\frac{x}{n-t+1}) \\ &= 1-\prod_{i=0}^{t-1}(1-\frac{x}{n-i}) \end{aligned} \quad (3)$$

In order to know more clearly the relationship among P, x, t, n, we simplify the Equation (3) and finally get

$$1-(1-\frac{x}{n-\frac{t-1}{2}})^t \leq P \leq (1-\frac{x}{n-t+1})^t$$

Suppose the data owner divides the $1GB$ file into $62,500$ blocks, each one of which is $16kb$. We will show the number of blocks(t) required in challenge message to detect the number of blocks corrupted by considering the probability of malpractice detection ($P = 0.9$) in Table 2. Also with the number of corrupted data blocks increasing, the data blocks in the challenge decreasing. When CSP modifies 10% of total outsourced blocks(n), then CSP needs only 23 random blocks in the challenge set to detect a server malpractice with 0.9 probability.

8.2 Computation Cost

As is shown in Table 3. $Hash$ is represented as a hash function, which can map a message from any long bit string to a fixed length. Mul represents a multiplication on group G. Exp is expressed as a exponentiation operation. $Pair$ is represented as a pairing operation. m represents the total number of files in the cloud, each file is divided into n blocks. t is the number of challenged blocks.

In Table 3, Jiang *et al.* [13] and Abadi *et al.* [1] used deduplication to reduce the redundant data in the cloud. Although the computational cost of Jiang *et al.* [13] is $2Exp + Mul + Hash + 2(Hash + Exp)\cdot O(\log m)$, which is a litter higher than Abadi *et al.* [1], the time of deduplication reduced greatly from linear $O(m)$ to logarithmic $O(\log m)$. Wang *et al.* [23] and Garg *et al.* [9]

Table 3: Computation cost of deduplication and auditing

Schemes	Deduplication			Auditing		
	$User$	CSP	$Time$	CSP	TPA	$Time$
[1]	$2Exp + Mul$ $+Hash$	$2Pair \cdot O(m)$	$O(m)$	–	–	–
[9]	–	–	–	$tExp$ $+(t-1)Mul$	$2Pair + tExp$ $+tMul$	$O(\log(n))$
[13]	$2Exp + Mul + Hash$ $+2(Hash + Exp) \cdot O(\log m)$	$\emptyset$	$O(\log m)$	–	–	–
[15]	$Hash \cdot O(m)$	$\emptyset$	$O(m)$	$tExp$ $+(t-1)Mul$	$2Pair + tExp$ $+tMul + tHash$	$O(n)$
[23]	–	–	–	$tExp$ $+(t-1)Mul$	$2Pair + (t+2)Exp$ $+(t+1)Mul$	$O(n)$
Ours	$2Exp + Mul + Hash$ $+2(Hash + Exp) \cdot O(\log m)$	$\emptyset$	$O(\log m)$	$tExp$ $+(t-1)Mul$	$2Pair + tExp$ $+tMul$	$O(\log(n))$

Table 4: Communication cost of deduplication and auditing

Phases	Communication cost
Data deduplication	$User \leftrightarrow CSP : l + k$
Delegation of audit task	$User \rightarrow TPA : 2\|G\|$
Challenge phase	$TPA \rightarrow CSP : t\|Z_p\| + 283$
Responding proof	$CSP \rightarrow TPA : \|G\| + \|Z_p\| + (t+2) \times 256$

can detect the integrity of data in the cloud. Although Li *et al.* [15] can achieve both deduplication and audit, it makes the computation cost of auditor's become $2Pair + tExp + tMul + tHash$, which is much higher than Garg *et al.* [9]. And when the file is encrypted, a fully trusted key server is needed. Therefore, we implement both by adopting the method of Jiang *et al.* [13] and Garg *et al.* [9], which can not only reduce the time complexity from linear to logarithmic by using a decision tree, but also make the time of searching node vary from linear to sub-linear by using the Merkle Hash Tree with a relative index.

8.3 Communication Cost

As is showed in Table 4, $|Z_p|$ and $|G|$ are expressed as the size of element in Z_p and G, respectively. t is the number of blocks challenged. l represents as the size of file tag. k represents as the size of decision bit.

Communication costs are mainly divided into four phases. The first phase is to perform the data deduplication test between the user and CSP which includes that CSP sends the tag to user and user returns the decision bits to CSP. The second phase is to authorize the integrity of data to the third parties, therefore the communication costs equals the size of keys, root signature and which is one time for every audit. The third phase is that TPA sends the challenge message to the CSP, which needs to extract t element subset Q randomly and select random element $\{b_i\}_{i \in Q} \in Z_p$ to the CSP, and the fourth phase is the CSP response to the challenge phase. We can obtain that Q can be represented as 283 bits in [9], when the size of file $\leq 1024TB$ and $2 \leq err \leq 80$(err is error a probability).

9 Conclusions

In this paper, we have presented a novel approach to realize a secure and efficient data deduplication scheme supporting dynamic data public auditing. The proposed scheme can not only save the storage cost in the cloud, but also use the auditor to verify the integrity of data in the cloud. It is significant to reduce the time to detect the redundant data and the calculation cost of searching the block of file. In addition, our scheme also supports the dynamic data operations, such as data insertion, data deletion, data modification and data append.

Acknowledgments

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

References

[1] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, "Message-locked encryption for lock-dependent messages," in *Advances in Cryptology (CRYPTO'13)*, pp. 374–391, 2013.

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

[3] M. Bellare and S. Keelveedhi, "Interactive message-locked encryption and secure deduplication," in *IACR International Workshop on Public Key Cryptography*, pp. 516–538, 2015.

[4] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: Server-aided encryption for deduplicated storage," *IACR Cryptology ePrint Archive*, vol. 2013, pp. 429, 2013.

[5] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 296–312, 2013.

[6] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Annual International Cryptology Conference*, pp. 213–229, 2001.

[7] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 514–532, 2001.

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

[9] N. Garg and S. Bawa, "Rits-mht: Relative indexed and time stamped merkle hash tree based data auditing protocol for cloud computing," *Journal of Network and Computer Applications*, vol. 84, pp. 1–13, 2017.

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

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

[12] M. S. Hwang, T. H. Sun, and 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. 05, pp. 1750072, 2017.

[13] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, and W. Lou, "Secure and efficient cloud data deduplication with randomized tag," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 3, pp. 532–543, 2017.

[14] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1615–1625, 2014.

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

[16] H. Lima, R. Araujo, R. Viegas, and D. Rosario, "A secure collaborative network protocol," in *Proceedings of the 9th Latin America Networking Conference*, pp. 46–52, 2016.

[17] C. H. Ling, C. C. Lee, C. C. Yang, and M. S. Hwang, "A secure and efficient one-time password authentication scheme for WSN," *International Journal of Network Security*, vol. 19, no. 2, pp. 177–181, 2017.

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

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

[20] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A secure data deduplication scheme for cloud storage," in *International Conference on Financial Cryptography and Data Security*, pp. 99–118, 2014.

[21] E. Stefanov, M. V. Dijk, A. Juels, and A. Oprea, "Iris: A scalable cloud file system with efficient integrity checks," in *Proceedings of the 28th Annual Computer Security Applications Conference*, pp. 229–238, 2012.

[22] S. F. Tzeng, M. S. Hwang, "Digital signature with message recovery and its variants based on elliptic curve discrete logarithm problem", *Computer Standards & Interfaces*, vol. 26, no. 2, pp. 61–71, Mar. 2004.

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

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

[25] J. Xu and E. C. Chang, "Towards efficient proofs of retrievability," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pp. 79–80, 2012.

[26] Z. Yan, M. Wang, Y. Li, and A. V. Vasilakos, "Encrypted data management with deduplication in cloud computing," *IEEE Cloud Computing*, vol. 3, no. 2, pp. 28–35, 2016.

[27] M. Yang and Y. M. Wang, "On the security of three public auditing schemes in cloud computing," *Inter-*

national Journal of Network Security, vol. 17, no. 6, pp. 795–802, 2015.

[28] J. H. Zhang, P. Y. 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.

Biography

Hua Ma received her B.S. and M.S. degrees in Mathematics from Xidian University, China, in 1985 and 1990, respectively. She is a professor of Mathematics and statistics, Her research includes security theory and technology in electronic commerce design and analysis of fast public key cryptography theory and technology of network security.

Xiaoyu Han received her B.S. degree in 2015 from College of Mathematics and Applied Mathematics, Lvliang University. Now, she is a master degree student in Mathematics at Xidian University. Her research focuses on network and information security.

Ting Peng received her B.S.degree in 2014 from College of Mathematics and Information Sciences, Xianyang Normal University. Now, she has graduated from Xidian University. Her research focuses on cryptography and network security.

Linchao Zhang received his B.S. degree in 2015 from College of Mathematics and Applied Mathematics, Hunan Institute of Science and Technology. Now, he is a master degree student in Mathematics at Xidian University. His research focuses on Proxy re-encryption and data security.