Research Article

Efficient Dynamic Replicated Data Possession Checking in Distributed Cloud Storage Systems

Jinxia Wei, Jianyi Liu, Ru Zhang, and Xinxin Niu

1 Information Security Center and National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, Beijing 100876, China
2 China Information Technology Security Evaluation Center, Beijing 100085, China

Correspondence should be addressed to Jinxia Wei; weijinxiaymx@bupt.edu.cn

Received 27 August 2015; Accepted 6 December 2015

Academic Editor: Zongming Fei

Copyright © 2016 Jinxia Wei et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

More and more organizations outsource their data to remote cloud servers (RCSs). Data owners rent the cloud service providers (CSPs) infrastructure to store their unlimited resources by paying fees metered in month or gigabyte. For increasing the availability and scalability of data, the data owners store their data replicas on multiple servers across multiple data centers. Data owners should ensure that the CSPs actually store all their copies according to the service contract. In this paper, we present an efficient dynamic multicopies possession checking scheme that has the following properties: (1) the data owner uses fully homomorphic encryption (FHE) algorithm to generate multiple copies; (2) the scheme supports data block dynamic operation; (3) the scheme supports third-party auditor’s public validation. Finally, security analysis and experimental results show that our scheme can resist forgery, replacement, and replay attack and perform better than some other related scheme published recently.

1. Introduction

Storing abundant file resources in the cloud has become the current trend [1]. An increasing number of users send their data to the cloud service providers (CSPs) for storage. Storing data on the remote cloud server enables organizations to relieve the burden brought by server updating and other computing issues. Also, authorized users can conveniently use the stored data from different geographic locations. When data has been stored on a remote CS that may be unreliable, the data owners cannot control their privacy data directly. This lack of control for privacy data would exhibit data confidentiality and privacy protection problem. For the CSP, some errors may occur, such as software or hardware failures and operational errors of system administrator [2–5]. For maintaining the CSP’s reputation, he/she usually tries to hide data loss incidents. Therefore, he/she may not be trustworthy. In order to deal with data confidentiality, the sensitive data is encrypted before it is outsourced to the CSP. For integrity protection, many researchers have proposed provable data possession (PDP) schemes to verify the integrity of data stored on the CSP.

For intensifying responsibility of the CSP, it is essential for the data owner to demand that the CSP provide evidence that the data owners’ data are not discarded or damaged [6]. PDP [7] is a method to check integrity of the data stored on the CSP. To keep the security of data, the data owner encrypts the data file and calculates tag for encrypted data file. Then he/she stores the encrypted data blocks and the corresponding tags on the CSP and deletes them from the local computers. When the data owner (or verifier) wants to check the integrity of data file, he/she sends a challenge vector to the CSP. The CSP makes a response for challenge and sends the response to the data owner (or verifier).

How to efficiently guarantee that the CSP faithfully stores the data owner’s data file is the main goal of remote PDP scheme. Nowadays, there exist two types of verification scheme: provable data possession (PDP) and Proof of Retrieval (POR). In addition, to validate the correctness of data, the POR scheme can also check out the corrupted data block and then recover the original data. In order to check the correctness of data stored on the CSP, Atienza et al. [7, 8] proposed two provably secure PDP schemes. In the subsequent year, different variations of PDP schemes
were proposed, such as [9–12] which are based on different cryptographic hard problem assumptions. In 2007, a model of POR was constructed in [13], Juels and Kaliski used error-correcting code to present a sentinel-based POR protocol. Various POR schemes that check static data file integrity can be found in [14–19].

In fact, it is not realistic to check static data file integrity, because the stored data block is usually modified by the data owner. Therefore, we consider that one of the basic requirements of storing data on the remote CSP is to support dynamic update operation of data block. The updated operation mainly refers to modification, insertion, or deletion of the stored data.

Another design goal is to support public validation. It is not appropriate to allow any side of the data owner or the CSP to verify the integrity of data stored on the CSP, because neither of them could be ensured to get unbiased verification results [20]. The third-party verifier is a suitable choice for the storage verification. A third-party verifier has the ability to do meaningful work and it is trusted by the data owners and CSPs. Under the influence of the distributed storage system, the data storage verification becomes more significant. Many schemes have been proposed: Proof of Retrieval (POR) protocols [21], remote integrity checking protocols [22, 23], and provable data possession protocols [24]. However, these protocols only apply to the scene that the data owner verifies the integrity of the data file stored on the CSP. In cloud computing, the data file storage verification is provided by a third-party rather than the data owners. Therefore, we propose a PDP scheme that supports public verifiability for data stored on the CSP and dynamic operation.

(1) Related Work. Ateniese et al. [7] first proposed a PDP paradigm and built two PDP schemes. However, they have not considered the multicopies storage system and dynamic data storage operation. To solve dynamic data operation problem, Zeng [10] and Wang et al. [20] proposed a dynamic version which can support limited data block operation but cannot support block insertions. In 2009, Erway et al. [25] modified the PDP scheme in [7] to achieve the dynamic updates of data files stored on the CSP applying rank-based authenticated skip lists. However, the computation and communication efficiency of this scheme achieves good only for a single data block. Wang et al. [20] verified the integrity of file copies using MHT. Although their scheme supports dynamic operation, their data files stored on the CSP are not encrypted and work for a single data block. In 2011, Hao et al. [26] proposed a PDP scheme which supports not only the dynamic operation for data file but also public audit. Public audit is an interaction protocol which allows any entity, not necessarily the data owner, to verify the correctness of data stored on the CSP, since their scheme also does not consider stored data file encryption and the scheme cannot be modified to support multicopies data file stored. Barsoum and Hasan [27] proposed a PDP scheme that encrypts a file \( F \) by using an encryption algorithm and generates different data file replicas. The encryption algorithm has a strong diffusion property, for example, AES. However, the data updating efficiency of their scheme is not good, for all the copies stored on the CSP will be encrypted again and updated on the CSP. To solve this problem, we use the FHE algorithm to encrypt the data file and generate multicopies stored on the CSP. The file copies need not be encrypted again when data updates. Therefore, the verification efficiency of our PDP scheme has been greatly improved.

The goal of our PDP scheme differs from the goal defined in the earlier researches. Our PDP scheme aims to keep confidentiality, integrity, and public verifiability of data copies stored on the CSP. Considering the FHE algorithm which will generate a random noise in each process, we encrypt a file \( F \) to get multiple file copies. The novelty of our scheme comes with ensuring data security and providing good efficiency for data updating while preserving the public verifiability. Because the file copies are stored on the distributed cloud server, we assume that all copies will not be damaged at the same time.

(2) Design Goal. The proposed scheme in our paper should hold the following properties simultaneously. (1) Correctness: the third-party auditor must accept all valid information proved by the cloud; (2) Public Auditing: the public auditor cannot get any information about the stored data block; (3) security goals: the proposed scheme can resist forgery attack, replacement attack, replay attack, and channel attack. The corrupted data block in corrupted copies can be checked out in this scheme. Thus, the data owner just needs to regenerate the new copy of corrupted data block instead of regenerating the whole new file copy. When the data owner needs to regenerate the valid form of corrupted data block copy, he/she first obtains the original data block from other uncorrupted data copies and then reencrypts the original data block to generate a new data block copy.

The rest of the paper is organized as follows. The preliminaries are introduced in Section 2, followed by the proposed scheme description in Section 3. Section 4 analyzes the security of our scheme. Section 5 presents the implementation and experimental result. Conclusions are given in Section 6.

2. Preliminary

Some preliminary knowledge used in our paper is introduced in this section. Let \( F \) be a file outsourced to the CSP. It is divided into a sequence of \( n \) blocks; that is, \( F = \{b_1, \ldots, b_n\} \), where \( b_j \in \mathcal{F} \) for some large prime \( q \). Denote \( \tilde{F}_i \) as the \( i \)th file copy. Thus \( \tilde{F}_i = \{\tilde{b}_i_1, \ldots, \tilde{b}_i_n\} \), where \( \tilde{b}_i_j \) represents the \( j \)th file block of the \( i \)th copy.

**Bilinear Map/Pairing.** Denote \( \mathcal{G}_1, \mathcal{G}_2, \) and \( \mathcal{G}_T \) as three cycle multiplicative groups; they have the same prime order \( q \); that is, \( |\mathcal{G}_1| = |\mathcal{G}_2| = |\mathcal{G}_T| = q \). Let \( e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T \) be a bilinear map [2], which holds the following properties:

(a) bilinearity: \( \forall g_1 \in \mathcal{G}_1, g_2 \in \mathcal{G}_2, \) and \( a, b \in \mathbb{F}_q^* \):

\[
e (g_1^a, g_2^b) = e (g_1, g_2)^{ab} ;
\]

(b) nondegeneracy: \( \exists g_3 \in \mathcal{G}_1, g_4 \in \mathcal{G}_2 \), such that \( e(g_3, g_4) \neq 1_{\mathcal{G}_T} \).
property that satisfies plaintext operation corresponding to
entfilereplicas. On the other hand, we use the homomorphic
original file.

On one hand, each homomorphic encryption generates a
data file copies. However, the encryption time complexity of their scheme is $O((\log N)^3)$; it is a
particul large number when the size of $N$ is 1024 bits. Therefore, we choose a fully homomorphic [28]
smaller time complexity than the scheme in [29] to
generate the data block copies.

The proposed MR-PDP scheme includes nine functions
KeyGen, CopiesGen, TagGen, Challenge, Response, Verify,
Request, Execution, and IndicieRetri.

(1) KeyGen. The data owner inputs a security parameter $\lambda$ and
generates a public/secret key pair $(pk, sk)$ for encryption or
decryption. Also, he/she chooses a random integer $\delta \in \mathbb{Z}_q$ as
secret key for signature and generates a public key $Y = g^\delta$
for verification; $g$ is the generator of $\mathbb{G}_2$.

(2) CopiesGen. For a file $F$, the data owner creates $\tau$
differentiable copies $F = \{F_1, \ldots, F_{\tau}\}$ using FHE algorithm.
For copy $F_j$, $1 \leq i \leq \tau$, it can be denoted as $F_j = \{b_{i1}, \ldots, b_{i\gamma}\}_{1 \leq i \leq \gamma}$, where $b_{ij}$ is the encryption of data block $b_j$.
Firstly, homomorphic encryption generates a random noise
that is a variable. Therefore, encrypting one original data $s$
times with a public key will generate $s$ different data replicas.
That is, even if $b_j = b_{j'}$, we still get $\tilde{b}_{ij} \neq \tilde{b}_{ij'}$, $1 \leq j, j' \leq n$, $j \neq j'$. Thus, the data owner can obtain different tags for
the same data blocks. Secondly, we use the homomorphic
property that satisfies plaintext operation corresponding to
ciphertext operation to realize dynamic operation for data
replicas stored on the CSPs, including data block insertion,
modification, and deletion. Finally, the authorized users only
need to hold a single decryption secret key $sk$. Upon obtaining
the file copies from the CSP, the authorized users decrypt the
file copies and get the original file plaintexts.

(3) TagGen. It is necessary to suppose that the data owner
generates the tags sequentially in accordance with the index $j$.
That is, the data owner generates a tag for a data block $\tilde{b}_{j2}$
after $\tilde{b}_{j1}$. For data block $\tilde{b}_{ij}$, $1 \leq i \leq \tau$, $1 \leq j \leq n$, the data
owner does the following:

(a) choosing a random element $\omega \in \mathbb{G}_1$;
(b) computing $T_{ij} = (h(F_N ||) \cdot \omega^{\delta}) \in \mathbb{G}_1$, where $F_N$ is
the name of file $F$;
(c) outputting $T_{ij}$;
(d) computing the tag $Tag$ of $(\tilde{b}_{ij}, T_{ij})$ using the secret key
δ and sending $Tag$ and $(\tilde{b}_{ij}, T_{ij})$ to the CSP.

Upon receiving the $Tag$ and $(\tilde{b}_{ij}, T_{ij})$, the CSP verifies whether
the block-tag pair $(\tilde{b}_{ij}, T_{ij})$ is correct or not. If the block-
tag pair $(\tilde{b}_{ij}, T_{ij})$ is correct, the CSP stores them. Otherwise,
he/she rejects them.

(4) Challenge. The data owner assigns the third auditor to
complete validation tasks. For challenging the CSP and
verifying the possession and integrity of all copies, the

3. Dynamic Multireplicas Provable Remote
Data Possession (DPRDP) Scheme

In this work, the cloud storage model considered in this paper
consists of four entities as shown in Figure 1: (1) a data owner
who can be an individual or an organization possesses privacy
data information, (2) the CSPs provide paid storage space
for storing the data owner’s data files, (3) the third-party
auditor is responsible for integrity verification of file copies
stored on the CSP through challenge-response protocol, and
(4) authorized users share the decryption key with the data
owner and have the right to access the data copies from the
CSP. In our paper, we mainly discuss the first three entities:
the data owner, the CSP, and the third-party verifier.

(c) computability: $\forall g_3 \in \mathbb{G}_1, g_6 \in \mathbb{G}_2$; there exists an
efficient algorithm to compute $e(g_3, g_6)$.

Computational Diffie-Hellman (CDH) Problem. $\mathbb{G}_1$ is a cyclic
multiplicative group on ECC generated by $g_1$. Given $g_3^a$ and
$g_6^b$ with $a, b \in \mathbb{Z}_q$, compute $g_3^{ab}$.

Our protocol involves three cryptographic functions: $h(\cdot)$ is a map-to-point hash function, $\psi(\cdot)$ is a pseudorandom
function, and $\theta(\cdot)$ is a pseudorandom permutation:
$$h(\cdot): \{0, 1\}^* \rightarrow \mathbb{G}_1,$$
$$\psi(\cdot): \mathbb{G}_q^* \times \{1, 2, \ldots, n\} \rightarrow \mathbb{G}_q^*,$$
$$\theta(\cdot): \mathbb{G}_q^* \times \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\}.$$

Fully Homomorphic Encryption (FHE) [28]. The FHE scheme
includes the following three algorithms:

KeyGen: input a security parameter $\lambda$ and output three parameters $(\rho, \eta, \gamma)$, where $\rho = \lambda, \eta = \lambda \cdot \lceil \log n \rceil, \gamma = 5 \cdot \lambda \cdot \lceil \log n \rceil / 2$. Let $p$ be an $\eta$-bit odd integer, $p \leftarrow (2Z + 1) \cap (2^{\gamma+1}, 2^{\gamma})$. Sign $p$ as the secret key $sk$. Compute $q_0 \leftarrow (2Z + 1) \cap (1, 2^\rho / p), t_0 = q_0 \cdot p$. Sign $t_0$ as the public key $pk$.

Enc: input a plaintext $m \in \{0, 1\}$ and the public key $t_0$; choose a random integer $q'$ from $[1, 2^{\rho} / p)$ and $r$ from $(-2^\rho, 2^\rho)$ and then compute ciphertext $C$:
$$C = E(m, pk) = (m + 2 \cdot r + q' \cdot p) \mod t_0.$$  

Dec: input a ciphertext $C$ and the secret key $sk$ and output the plaintext $m$:
$$m = D(C, sk) = (C \mod p) \mod 2.$$  

On one hand, each homomorphic encryption generates a
random noise, which is a variable. Therefore, encrypting one
original file $s$ times with one public key will generate $s$
different file replicas. On the other hand, we use the homomorphic
property that satisfies plaintext operation corresponding to
ciphertext operation to realize dynamic operation for data
replicas stored on the CSPs.
The third-party auditor sends $c$ (number of data blocks to be challenged) and two challenge keys $k_1, k_2$ at each challenge stage: a pseudorandom permutation key $k_1 \in \mathcal{X}^*_q$ and a pseudorandom function key $k_2 \in \mathcal{X}^*_q$. Both the CSP and the third-party auditor use $\theta$ keyed with $k_1$ and $\psi$ keyed with $k_2$ to generate a set $S = \{(j^t, r_j^t)\}$ of $c$ pairs of random indices and their random values, where $r_j^t = \psi_{k_2}(t)_1^{\#S_t}$ and $j^t = \theta_{k_1}(t)_1^{\#S_t}$. The random index $j^t$ indicates the physical positions of $c$ data blocks to be challenged. Let the challenge be a triple $(c, k_1, k_2)$.

(5) **Response.** The CSP receives the triple $(c, k_1, k_2)$ and computes random indices $j^{t'}$ and random values $r_j^{t'}$. After obtaining the values of $j^{t'}$ and $r_j^{t'}$, the CSP does the following:

(a) computing $T^{t'} = \prod_{(j^t, r_j^t) \in S} T_{j^t}^{r_j^t}$, where $T_{j^t}^{r_j^t} = \prod_{i=1}^{\tau} T_i^{r_j^t} \in G$;

(b) computing $\xi = \sum_{(j^t, r_j^t) \in S} r_j^t \cdot \tilde{b}_{j^t}^t \in \mathcal{Z}_p$, where $\xi = \{\xi_i\}_{1 \leq i \leq \tau}$;

(c) sending $(T, \xi)$ to the verifier.

(6) **Verify.** After receiving the proof $(T, \xi)$ from the CSP, the third-party auditor checks the following verification equation:

$$ e(T, g) = e \left( \prod_{(j^t, r_j^t) \in S} \prod_{i=1}^{\tau} h(F_N || i)^{r_j^t} \cdot \omega_{\sum_{i=1}^{\tau} \xi_i}, Y \right). $$

If the above equation holds, the third-party auditor outputs 1, otherwise 0.

---

**Figure 1:** Cloud storage system model.

**Figure 2:** The challenge-response protocol in the DPRDP scheme.
The data owner

- Chooses the updated data block \( b'_j \)
- Encrypts \( \Delta b_j \) and obtains \( \tilde{b}'_j = E(\Delta b_j) \), where \( \Delta b_j = b'_j - \tilde{b}_j \)
- Generates a new tag \( T'_{ij} \) for \( \tilde{b}'_j, T'_{ij} = (h(F_N \parallel i) \cdot \omega^\pi)^\theta \)
- Chooses two keys \( k_1 \) and \( k_2 \) to verify the correctness of the updated data file copies

The CSP

- \( (F_N, \text{modify, } j, E(\Delta b_j), T'_{ij}, k_1, k_2) \)
- Implements homomorphic addition/multiplication (\(*\) operation)
- \( E(b'_j) = E(\tilde{b}_j) \ast E(\Delta b_j) \)
- Replaces the old tag with \( T'_{ij} \)

\[ \text{Verifies } e(T, g) = e(\prod_{i'=1}^\tau \mathcal{H}^{k_{ij}} F_N, \omega^{\sum_{i' = 1}^\tau i_i}, Y) \]

Figure 3: Data block modification operation procedure in the DPRDP scheme.

(7) Request. When the data owner wants to update the data block stored on the CSP, he/she would generate an update request including file name, update command, update index, and encrypted difference and update tag. The data owner sends the update request \((F_N, \text{Request}, j, \text{FHE}(\Delta b_j), T'_{ij})\) to the CSP, where Request denotes update request that may be modification, deletion, or insertion. \( j \) is the index of the updated data block, \( \text{FHE}(\Delta b_j) \) is the encryption of difference between the original data block and the updated data block, and \( T'_{ij} \) is the updated tag. The data block modification includes data block addition and multiplication.

(8) Execution. The CSP receives the update request and executes the update operation. He/she inputs the copies of file \( F \), the tag \( T \), and update request and outputs the copies of an updated file \( F' \) and its new tag \( T' \). Upon completing the block data update operation, the third-party auditor executes the verification protocol to confirm that the CSP has implemented the update operation correctly. We give a detailed data block modification procedure in Figure 3.

The data block insertion and deletion operations are similar with the data block modification. In order to complete the data block insertion, the data owner inserts a new data block before position \( j \) in a copy file. Thus, the number of data blocks in copies will be changed from \( n \) to \( n + 1 \). The data owner chooses an inserted data block \( b'_j \) and encrypts it using FHE algorithm. Also, the data owner calculates a new tag \( T'_{ij} \) for the inserted data block ciphertext \( \tilde{b}'_{ij} \) of \( b'_j \) and then sends \((F_N, \text{insert } j, E(\tilde{b}'_{ij}), T'_{ij})\) to the CSP. Upon receiving these elements, the CSP inserts the new data block ciphertext \( \tilde{b}'_{ij} \) before position \( j \) in copy file \( F''_i \), \( 1 \leq i \leq \tau \). Hence, the inserted file copy can be represented as \( F''_i = (\tilde{b}_{i_1}, \ldots, \tilde{b}_{i_j-1}, \tilde{b}'_{ij}, \tilde{b}_{ij+1}, \ldots, \tilde{b}_{i_{\tau}}) \) and the new data block-tag set can be denoted as \( (\prod_{i=1}^\tau T_{ij}, \ldots, \prod_{i=1}^\tau T_{ij}, \prod_{i=1}^{n+1} T_{ij}, \ldots, \prod_{i=1}^n T_{ij}) \). Finally, the CSP makes a response with the auditor to prove that he/she correctly performs the insertion operation. Data block deletion is opposite to the insertion operation. Upon deleting a data block, the number of file data blocks will be changed from \( n \) to \( n-1 \). If the data owner wants to delete the \( j \)-th block from each copy, he/she sends a delete command \((F_N, \text{Delete, } j, \Theta, \Theta)\) to the CSP, where \( \Theta \) denotes empty element. After receiving this command, the CSP deletes the data block at the \( j \)-th position of all file copies and its tag.

(9) IndiceRetri. The indices of corrupted data blocks can be identified by using this retrieval algorithm. The response \((T, \xi)\) generated by the CSP will be correct and will pass the third-party auditor’s verification only if the data blocks in all copies are consistent and intact. Hence, there exist one or more corrupted data blocks; the whole verification procedure fails. When file copy is corrupted, the data owner should recover the file copy timely for ensuring that the authorized users in different locations can normally use the file copy. If the corrupted copy only includes one corrupted data block, updating the whole file copy may lead to huge computational and communication cost. Therefore, it is necessary to identify the indices of corrupted data block in corrupted file copy. We show a detailed retrieval algorithm in Algorithm 1.

The retrieve algorithm inputs four elements: \( T, \xi, k_1, \) and \( k_2 \). If CSP’s response does not pass the verification, the third-party auditor uses Algorithm 1 to identify the index of corrupted data block. The third-party auditor first identifies which cloud server contains the corrupted data copies. Then he/she checks out the damaged data copies from the cloud server which contains the damaged data copies. The verier calculates \( n_c = \lceil \sqrt{c} \rceil \) and \( n'_c = \lceil \sqrt{c} \rceil \) and divides \( S \) into \( n'_c \) parts, \( S_1, S_2, \ldots, S_{n'_c} \), where \( S_t (1 \leq t \leq n'_{c} - 1) \) includes \( n_c \) elements and \( S_{n'_{c}} \) includes \( c - n_c(n'_{c} - 1) \) elements. Then,
the third-party auditor, respectively, verifies the correctness of data block whose index locates in $S_i$. If the response can pass the third-party auditor’s verification, the data blocks whose indices belong to $S_i$ will be kept correctly. Otherwise, the verifier will continue to check the integrity of smaller range data block (the number of data blocks $\lceil \sqrt{\xi_i^r} \rceil$) until remaining two data blocks or checking all corrupted data blocks. Finally, we can get the indices of corrupted data block in the cloud server. Thus, when the data owner recovers the corrupted copies, he/she only needs to regenerate a correct data block to replace the corrupted data block rather than regenerating a new file copy.

We give an example for Algorithm 1. As shown in Figure 4, Algorithm 1 first gets the corrupted data block copies $\bar{F}_2$ and $\bar{F}_r$ which are marked with blue frame and then obtains the corrupted copies information $\bar{b}_1$ and $\bar{b}_3$. That is, the actually corrupted data blocks are $\bar{b}_{21}$ and $\bar{b}_{r3}$ which are marked with red frame. Therefore, the data owner only needs to regenerate the data block copies $\bar{b}_{21}$ and $\bar{b}_{r3}$ and restores them in the copies $\bar{F}_2$ and $\bar{F}_r$, respectively.

4. Security Analysis

**Theorem 1.** If both the data owner and the CSP are honest to perform the DPRDP scheme, the response of CSP can pass the auditor’s validation:

$$e(T, g) = e\left( \prod_{(j', r) \in S} \prod_{t=1}^{\tau} h(F_N || i)_{j'}^{r'/t}, g \right)$$

$$= e\left( \prod_{(j', r) \in S} \prod_{t=1}^{\tau} h(F_N || i)_{j'}^{r'/t} \cdot \omega_{\xi_{t-1}}^{r'}, Y \right)$$

**Theorem 1 is proved.** The details of security analysis are given in the following.

(1) Scenario. In our protocol, there exist four entities in the security model: the data owner, verifier, CSP, and authorized user. The security analysis of protocol involves the first three participants, so we only consider the communication among the data owner, verifier, and CSP. The data owner encrypts data block and gets the replicas of each data block. Then the data owner generates the tags corresponding to the data.
block replicas and stores the block-tag pair on the CSP. When the verifier wants to validate the integrity of data copies, it sends to the CSP a challenge vector. Upon receiving the challenge vector, the CSP aggregates the tag of challenged data block and the challenged data blocks, respectively, and then sends the aggregation back to the verifier. That is, the verifier sends a challenge vector to the CSP, and then the CSP makes a response for the challenge. Finally, the verifier checks the aggregation information. The interaction among the data owner, CSP, and verifier is shown in Figure 5.

(2) Possible Attack

(a) Internal Attack. The main internal attack considered in our scheme refers to forgery attack, replacement attack, and replay attack. All these attacks initiated by dishonest CSP. The dishonest CSP may forge a single data block tag or replace a corrupted data block-tag pair with another valid block-tag pair. Even, he/she may also try to make a response to the current challenge of verifier with the previous proof, without querying the current data block actually.

(b) External Attack. An adversary may temper the messages transmitting on the channel.

(3) Interaction Protocol Design. In order to clearly present interaction between any two participants, a detailed communication process is given in Figure 6.

(4) Security Analysis

(a) External Attack. An adversary may temper the transmitted messages on channel shown in Figure 4. On the first transmission channel, the data owner sends \((T_{ij}, \tilde{b}_{ij})\) and tag Tag\((T_{ij}, \tilde{b}_{ij})\) to the CSP. Upon receiving these elements, the CSP verifies whether the received block-tag pair \(T_{ij}, \tilde{b}_{ij}\) is correct or not according to Tag\((T_{ij}, \tilde{b}_{ij})\) and the public key \(Y\). If the adversary tampers \(T_{ij}\) or \(\tilde{b}_{ij}\), the CSP would fail to validate and refuse to store the received \((T_{ij}, \tilde{b}_{ij})\). Therefore, the adversary cannot attack successfully in the first channel.

For the adversary, tempering with the transmitted messages on the channels \(\oplus\) and \(\ominus\) can be regarded as the CSP tempering with these messages. In addition to tempering messages, the CSP can also initiate forgery, replacement, and replay attack. That is, the CSP has stronger attack ability than an external adversary. Therefore, for the messages transmitted on the second and the third channel, we just need to consider the internal attack initiated by the CSP.

(b) Internal Attack

Forgery. A dishonest CSP attempts to forge the tag for some data block and wishes that the value of forged tag is the same as the original tag. For a single tag \(T_{ij} = (h(F_N||l) \cdot \omega^{\tilde{b}_{ij}^i})^g\), the CSP may change the input information of hash function \(h()\) and the value of \(\tilde{b}_{ij}\). Because the hash function \(h()\) resists collision, the hash result would change a lot if the CSP changes the input value of \(F_N, i, \) or \(j\). For the random element \(\omega \epsilon \mathbb{G}_1\), the exponentiation result would change significantly if the CSP changes \(\tilde{b}_{ij}\). Therefore, it is hard for the CSP to forge a single tag.

Replacement. Assuming that the CSP uses a valid block-tag pair \((T_{iw}, \tilde{b}_{iw})\epsilon G\) to replace some block-tag pair \((T_{ij}, \tilde{b}_{ij})\epsilon G\). We get a conclusion that if the CSP replaces \((T_{ij}, \tilde{b}_{ij})\epsilon G\) using another valid block-tag pair \((T_{iw}, \tilde{b}_{iw})\), the proof \((T', \xi')\) passes the third-party auditor’s verification with negligible probability.

According the verification equation, the following equations are obtained:

\[
e(T', g) = e \left( \prod_{(j', r_j') \epsilon S, j' \neq l} \left( \prod_{i=1}^{r} h(F_N||l)^{r_j'} \cdot \omega^{\sum_{i'=1}^{r} r_{j'}\tilde{b}_{i'}} \cdot \left( \prod_{i=1}^{r} h(F_N||l)^{r_j'} \right) \cdot \omega^{\sum_{i'=1}^{r} r_{j'}\tilde{b}_{i'}} \right) \right) \cdot Y
\]

\[
e(T, g) = e \left( \prod_{(j', r_j') \epsilon S, i=1}^{r} h(F_N||l)^{r_j'} \cdot \omega^{\sum_{i'=1}^{r} \xi_i} \cdot Y \right)
\]

We set

\[
\prod_{i=1}^{r} h(F_N||l)^{r_j'} \cdot \omega^{\sum_{i'=1}^{r} r_{j'}\tilde{b}_{i'}} = g^{x'},
\]

\[
\prod_{i=1}^{r} h(F_N||l)^{r_j'} \cdot \omega^{\sum_{i'=1}^{r} r_{j'}\tilde{b}_{i'}} = g^{y'},
\]

where \(x', y' \epsilon \mathbb{F}_q^*, x' \neq y'\).

If \((T', \xi')\) can pass the verifier’s verification, we have \(g^{x'} = g^{y'}\). Since \(q\) is a large prime integer, the probability of \(g^{x'} = g^{y'}\) is negligible.

Replay. The essence of replay attack is to use the previous proof to replace the current challenge’s response. That is, the CSP does not actually access the latest data status. To some extent, the replay attack can be regarded as the multiple replacement attack. Therefore, we can use the same analysis method as replacement attack to analyze replay attack. If the challenged latest block-tag pairs are different with the previous block-tag pairs, the previous response will not pass the verifier verification.
The data owner
Chooses $\delta$ as the secret key and
$Y = g^\delta$ as the public key
For $F = \{b_1, \ldots, b_n\}$, he/she computes $\tilde{b}_i = \text{Enc}(b_i)$
For hash function $h : \{0, 1\}^* \rightarrow G_1$,
he/she computes $T_{ij} = (h(F_N \parallel i) \cdot \omega_{\tilde{b}_i})^\delta$

Upon receiving $\text{Sig}(T_{ij}, \tilde{b}_i)$ and $\text{Sig}(T_{ij}, \tilde{b}_i)$
he/she verifies whether sig is correct or not;
if the sign is correct, then he/she stores $(T_{ij}, \tilde{b}_i)$;
otherwise, he/she refuses it

The verifier
Chooses $c$ as the number of challenged block
For $k_1, j' = \theta_{k_1}(t)_{1\ldots\text{SEC}}$
For $k_2, r_j' = \psi_{k_2}(t)_{1\ldots\text{SEC}}$
Signs challenge vector $(c, k_1, k_2)$

Verifies $e(T, g) = e(\prod_{j', r_j' \in S}(\prod_{i=1}^{\tau} T_{ij})^{r_j} \cdot \tilde{b}_j, Y)$

Figure 6: The communication process among the data owner, CSP, and verifier.

5. Performance Analysis

In terms of our scheme implementation, we have simulated
this MR-PDP protocol in C language. We conduct the
following experiments on the Win7 system. To start, we
perform several experiments on a system with an Intel(R)
Core(TM) 3.10GHZ processor and 4 GB RAM computer.
We set the encryption security parameters $\lambda = 60$
according to [28]. We store copies of a data file of sizes 1 MB, 5 MB, 10 MB,
and 20 MB, and their copies are divided into $2^{18}$ blocks. We
assume that the elliptic curve group we deal with has a 256-bit
group order [30]. The communication cost for each stage (file
size 1 MB) in this protocol is presented in Table 1, where $s$
is number of data files, $B_h$ is the size of hash function, $C$
denotes the size of ciphertext, $B_N$ is the size of the file name, and $\Delta C$
is the size of FHE($\Delta b_j$).

We give a performance comparison between the MR-PDP
scheme proposed in our scheme and the DMR-PDP scheme
proposed in [29].

Figure 7 presents the data setup computation time
comparison between the DMR-PDP scheme and the DPRDP
scheme using three copies for the file sizes 1, 5, 10, and 20 MB,
respectively. Similarly with the scheme in [29], the data setup
process is completed by the data owner only once. As shown
in Figure 7, the DPRDP scheme performs faster than DMR-
PDP scheme.

We use the FHE encryption to generate the data block
copy, but Raghul uses the Paillier encryption to generate the
data block copies in DMR-PDP scheme. The following three
elements are considered as the main influence factors for
the efficiency of data setup. (1) The generation of encryption
parameters: the time for generating parameter of FHE is
much faster than that in Paillier encryption. In addition to
generating random numbers, Paillier encryption also needs
to compute the least common multiple which is a relatively
consuming time process when both $p$ and $q$ are large number.
However, this is not the main factor to affect the efficiency
of data setup. (2) Encryption: the time complexity of FHE
is $O(\gamma)$ and the time complexity of Paillier encryption is $2 \cdot O((\log N)^3)$. Referring to the security parameter in DMR-PDP scheme and DPRDP, we get $O(\gamma) < 2 \cdot O((\log N)^3)$. The encryption time is the main factor of influencing the data setup efficiency. As shown in Figure 8, the computation difference between DMR-PDP scheme and DPRDP scheme is more and more obvious with the increase of file size.

(3) Signature parameter setting: both DMR-PDP scheme and DPRDP scheme are involved in power computation. But the DMR-PDP scheme also needs to complete $N$ times multiplication and addition operations; it will slow down the computation speed. Therefore, the data setup efficiency of the DPRDP scheme is better than that in the DMR-PDP scheme. In particular, when the file size increases, the advantage of the DPRDP scheme is more obvious.

In Figure 8, the signing efficiency comparison of data owner in DPRDP scheme and DMR-PDP scheme is presented. We set the number of replicas as 3, 5, and 10. The solid line denotes the signing time of data owner in DMR-PDP scheme and the dotted line represents the signing time of data owner in DPRDP scheme.

The signing efficiency comparison of data owner in DPRDP scheme and DMR-PDP scheme for file sizes 1MB and 5MB is shown in Figure 8(a). The red line denotes the data owner signing time comparison with file size 1MB for the number of copies 3, 5, and 10. Although the algorithms for generating tags in DPRDP scheme and DMR-PDP scheme are two similar methods, but multiplication and addition must be needed in DMR-PDP scheme, the signing time will be slower than that in DPRDP scheme. The blue line denotes the data owner signing time comparison with file size 5MB for the number of copies 3, 5, and 10. When there exist 3 data copies, the signing time in DMR-PDP scheme is 0.037 seconds slower than that in DPRDP scheme. In Figure 8(b), we show the signing efficiency comparison of data owner in DPRDP scheme and DMR-PDP scheme for file sizes 10MB and 20MB. The green line denotes the data owner signing time comparison with file size 10MB with the number of copies 3, 5, and 10. When the file size is 10MB, the difference of signing time between these two schemes is small. In Figure 8(b), the purple line denotes the data owner signing time comparison with file size 20MB with the number of copies 3, 5, and 10. For 20MB file, the growth rate of signing time in DMR-PDP scheme and DPRDP scheme increases a little when the number of copies changes from 3 to 5.

The tag aggregation efficiency comparison of CSP in DPRDP scheme and DMR-PDP scheme is presented in Figure 9. Let the number of replicas be 3, 5, and 10. The solid line denotes the tag aggregation time of CSP in DMR-PDP scheme and the dotted line represents the tag aggregation time of CSP in DPRDP scheme.

In Figure 9(a), we show the tag aggregation efficiency comparison of CSP in DPRDP scheme and DMR-PDP scheme for file sizes 1MB and 5MB. The red line denotes the CSP aggregation time comparison with file size 1MB for the number of copies 3, 5, and 10. For the algorithms for aggregation tags in DPRDP scheme and DMR-PDP scheme are two similar methods, the difference of aggregation time between these two schemes is small when the file size is 1MB. The blue line denotes the CSP aggregation time comparison with file size 5MB with the number of copies 3, 5, and 10. When there exist 3 data copies, the aggregation time in DMR-PDP scheme is 2.2 seconds slower than that in DPRDP scheme. The aggregation efficiency difference between DMR-PDP scheme and DPRDP scheme narrows as the increase of number of copies. When the number of copies is 10, the aggregation efficiency of the two schemes is very close.

The aggregation efficiency comparison of CSP in DPRDP scheme and DMR-PDP scheme for file sizes 10MB and 20MB is shown in Figure 9(b). The green line denotes the CSP aggregation time comparison with file size 10MB for the number of copies 3, 5, and 10. When the file size is 10MB, the difference of aggregation time between these two schemes is small. With the increase of number of copies, growth rate of aggregation time is getting bigger. The purple line denotes the CSP aggregation time comparison with file size 20MB for the number of copies 3, 5, and 10. Also, the growth rate of aggregation time is getting bigger for both DMR-PDP scheme and DPRDP scheme. For 20MB file, the growth rate of aggregation time in DMR-PDP scheme is faster than that

### Table 1: DPRDP scheme communication cost with file size 1MB.

| Phase          | Cost                           | Data   | From          | To        |
|----------------|--------------------------------|--------|---------------|-----------|
| Storage        | $2^{23} \cdot \log C$ bits     | C      | Data owner    | CSP       |
| Challenge      | $\log N + 2 \log q$ bits       | $(c, k_1, k_2)$ | Data owner/verifier | CSP       |
| Verification   | $\tau \cdot c \cdot (2 \cdot q^2 \cdot B_h \cdot C \cdot \log q + \log C \cdot q)$ bits | $(T, \xi)$ | CSP | Data owner/verifier |
| Update         | $\log B_h + \log \Delta C + 2 \cdot \tau \cdot q \cdot B_h \cdot C \cdot \log q$ bits | $(F_N, \text{Request}_j, \text{FHE}(\Delta b_j), T_j')$ | Data owner | CSP |
| Retrieval      | $c \cdot \tau \cdot (2 \cdot B_h \cdot C \cdot \log q + \log C)$ | $(T_{ij}, \hat{b}_{ij})$ | CSP | Data owner/verifier |

![Figure 7: The Comparison of data setup time between DMR-PDP and DPRDP scheme.](image-url)
in DPRDP scheme when the number of copies changes from 5 to 10. Therefore, when both the number of data copies and the file size are large, the DPRDP scheme is more effective.

6. Conclusion

In this paper, we have presented a dynamic replicated data possession checking scheme for checking the data copies integrity in the CSP. The data owners encrypt their data using FHE algorithm, obtain the data replicas, and then store their encryption data replicas on the CSP. Firstly, the proposed scheme not only supports checking integrity of the data file copies anytime and anywhere but also satisfies public verification without leaking any information to the third-party auditor. Secondly, the authorized users can decrypt data copies received from the CSP using a single decryption secret key. Thirdly, the data owner can implement data block update operation on data replicas. Finally, the retrieval algorithms in this scheme can identify the corrupted data blocks in corrupted copies accurately. The corrupted data block copies can be reconstructed by the data owner, and thus he/she needs not to generate the whole corrupted copy file. Through security analysis, we have shown that our scheme can resist forgery attacks, replacement attack, and replay attack. At the same time, our DPRDP scheme achieves a higher efficiency than DMR-PDP scheme.

We consider that there exists data expansion in the encryption process, so how to reduce the data expansion is our future work. Thus, we can further improve the efficiency of the proposed scheme.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors thank the editor and reviewers for their suggestions to improve the quality of paper. This work was
References

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

[2] A. Jula, E. Sundararajan, and Z. Othman, “Cloud computing service composition: a systematic literature review,” Expert Systems with Applications, vol. 41, no. 8, pp. 3809–3824, 2014.

[3] H. Wang, J. Domingo-Ferrer, Q. Wu, and B. Qin, “Identity-based remote data possession checking in public clouds,” IET Information Security, vol. 8, no. 2, pp. 114–121, 2014.

[4] W. Wang, G. Zeng, and J. Yao, “Cloud-DLS: dynamic trusted scheduling for cloud computing original research article,” Expert Systems with Applications, vol. 39, no. 3, pp. 2321–2319, 2012.

[5] H. Q. Wang, “Identity-based distribute provable data possession in multi-cloud storage,” IEEE Transactions on Services Computing, vol. 8, no. 2, pp. 328–340, 2015.

[6] Y.-K. Lin and P.-C. Chang, “Maintenance reliability estimation for a cloud computing network with nodes failure,” Expert Systems with Applications, vol. 38, no. 11, pp. 14185–14189, 2011.

[7] G. Ateniese, R. Burns, R. Curtmola et al., “Provable data possession at untrusted stores,” in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS ’07), pp. 598–609, ACM, Alexandria, Va, USA, October-November 2007.

[8] G. Ateniese, R. Burns, R. Curtmola et al., “Remote data checking using provable data possession,” ACM Transactions on Information and System Security, vol. 14, pp. 1–34, 2011.

[9] M. A. Shah, R. Swaminathan, and M. Baker, “Privacy-preserving audit and extraction of digital contents,” Report 2008/186, Cryptology ePrint Archive, 2008.

[10] K. Zeng, “Publicly verifiable remote data integrity,” in Information and Communications Security: 10th International Conference, ICICS 2008 Birmingham, UK, October 20–22, 2008 Proceedings, vol. 5308 of Lecture Notes in Computer Science, pp. 419–434, Springer, Berlin, Germany, 2008.

[11] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, “Auditing to keep online storage services honest,” in Proceedings of the 11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS ’07), pp. 1–6, Berkeley, Calif, USA, 2007.

[12] F. Sebé, J. Domingo-Ferrer, A. Martínez-Ballesté, Y. Deswarte, and J.-J. Quisquater, “Efficient remote data possession checking in critical information infrastructures,” IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 8, pp. 1034–1038, 2008.

[13] A. Juels and B. S. Kaliski Jr., “Pors: proofs of retrievability for large files,” in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS ’07), pp. 584–597, Alexandria, Va, USA, October 2007.

[14] H. Shacham and B. Waters, “Compact proofs of retrievability,” in Advances in Cryptology—ASIACRYPT 2008, vol. 5350 of Lecture Notes in Computer Science, pp. 90–107, Springer, Berlin, Germany, 2008.

[15] K. D. Bowers, A. Juels, and A. Oprea, “HAIL: a high-availability and integrity layer for cloud storage,” in Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS ’09), pp. 187–198, ACM, Chicago, Ill, USA, November 2009.

[16] K. D. Bowers, A. Juels, and A. Oprea, “Proofs of retrievability: theory and implementation,” in Proceedings of the ACM Cloud Computing Security Workshop (CCSW ’09), pp. 43–54, Chicago, Ill, USA, 2009.

[17] Y. Dodis, S. Vadhan, and D. Wichs, “Proofs of retrievability via hardness amplification,” in Proceedings of the 6th Theory of Cryptography Conference (TCC ’09), pp. 109–127, San Francisco, Calif, USA, March 2009.

[18] R. Curtmola, O. Khan, and R. Burns, “Robust remote data checking,” in Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, pp. 63–68, October 2008.

[19] A. Juels and B. S. Kaliski Jr., “Pors: proofs of retrievability for large files,” in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS ’07), pp. 584–597, ACM, November 2007.

[20] C. Wang, K. Ren, W. Lou, and J. Li, “Toward publicly auditable secure cloud data storage services,” IEEE Network, vol. 24, no. 4, pp. 19–24, 2010.

[21] A. Juels and B. S. Kaliski Jr., “Pors: proofs of retrievability for large files,” in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS ’07), pp. 584–597, ACM, New York, NY, USA, November 2007.

[22] Y. Deswarte, J.-J. Quisquater, and A. Sadane, “Remote integrity checking,” in Proceedings of the 6th Working Conference on Integrity and Internal Control in Information Systems (IICIS ’03), S. J. L. Strous, Ed., pp. 1–11, Lausanne, Switzerland, November 2003.

[23] H. Q. Wang, Q. H. Wu, B. Qin, and J. Domingo-Ferrer, “Identity-based remote data possession checking in public clouds,” IET Information Security, vol. 8, no. 2, pp. 114–121, 2014.

[24] D. L. G. Filho and P. S. L. M. Barreto, “Demonstrating data possession and uncheatable data transfer,” Report 2006/150, Cryptology ePrint Archive, 2006.

[25] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, “Dynamic provable data possession,” in Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS ’09), pp. 213–222, ACM, New York, NY, USA, November 2009.

[26] Z. Hao, S. Zhong, and N. Yu, “A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability,” IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 9, pp. 1432–1437, 2011.

[27] A. F. Barsoum and M. A. Hasan, “On verifying dynamic multiple data copies over cloud servers,” Report 2011/447, Cryptology ePrint Archive, 2011, http://eprint.iacr.org/.

[28] X. Yi, M. G. Kaosar, R. Paulet, and E. Bertino, “Single-database private information retrieval from fully homomorphic encryption,” IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 5, pp. 1125–1134, 2013.

[29] R. Mukundan, S. Madria, and M. Linderman, “Efficient integrity verification of replicated data in cloud using homomorphic encryption,” Distributed and Parallel Databases, vol. 32, no. 4, pp. 567–534, 2014.

[30] A. F. Barsoum and M. A. Hasan, “Provable multicopy dynamic data possession in cloud computing systems,” IEEE Transactions on Information Forensics and Security, vol. 10, no. 3, pp. 485–497, 2015.