IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 24, 2020, accepted June 28, 2020, date of publication July 1, 2020, date of current version July 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3006278

Dynamic Multiple-Replica Provable Data Possession in Cloud Storage System

YILIN YUAN, JIANBIAO ZHANG, (Member, IEEE), AND WANSHAN XU
Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
Beijing Key Laboratory of Trusted Computing, Beijing 100124, China
National Engineering Laboratory of Key Technologies for Information Security Level Protection, Beijing 100124, China

Corresponding author: Jianbiao Zhang (zjb@bjut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61971014.

ABSTRACT In cloud storage scenarios, data security has received considerably more attention than before. To ensure the reliability and availability of outsourced data and improve disaster resilience and data recovery ability, important data files possessed by users must be stored on multiple cloud service providers (CSPs). However, we know that CSP is invariably not reliable. In this situation, to verify the integrity of replica files stored by users on multiple CSPs simultaneously, a new dynamic multiple-replica provable data possession (DMR-PDP) scheme is proposed. In addition, due to the importance of the tag set, we utilize vector dot products instead of using the modular power calculation in the traditional PDP scheme, which greatly reduces the calculation time and storage space usage. Moreover, a novel dynamic data structure, the divided address-version mapping table (DAVMT), is presented and used to solve the problem of data dynamic operation. A practical experiment validates the effectiveness of our proposed scheme in the end.

INDEX TERMS Cloud storage, data security, provable data possession, dynamic operation.

I. INTRODUCTION

Cloud computing [1], [2], as the new generation of distributed computing, parallel computing and grid computing, has achieved rapid development and wide application since it was proposed. The advantages of cloud computing, such as scalability, on- demand service, and high- reliability flexibility, have attracted a large number of users. However, the increasing security issue has become the primary obstacle that restricts the development of cloud computing.

As we mentioned before, cloud storage [3], [4] is an extension of the concept and application of cloud computing and has been developed continuously as a new type of network storage technology since it was put forward. Cloud storage is essentially a cloud computing system with data storage and management as the core. Therefore, ensuring the security of outsourced data is also the primary problem restricting the development of cloud storage systems. The security of data storage consists of three aspects: confidentiality, integrity and availability (CIA). Confidentiality guarantees that unauthorized users cannot access the data, integrity guarantees unauthorized users cannot modify the data illegally, and availability guarantees legally authorized users can access the data in a timely, accurate and uninterrupted manner. In our study, the integrity of data storage security is our main focus, and the details are discussed as follows.

In the cloud storage scenario, when a user uploads the local data to the cloud, the control over the outsourced data may be totally lost; thus, data integrity becomes a problem. To verify the integrity of the outsourced data, the PDP scheme [5] was proposed in 2007. In the PDP scheme, the DO (data owner) calculates a set of homomorphic tags for the outsourced data, uploads them together with the encrypted file to the CSP, and deletes the local file on the premise of keeping the secret key. When the DO needs to verify the integrity of the data stored on the cloud, he/she will send a challenge to the CSP, the CSP responds to the challenge, and the DO verifies the response. Different from the traditional integrity verification scheme, in the PDP scheme, the DO uses a sampling method with probability in the integrity verification phase. However, as the PDP scheme in [5] is only applicable to static data and cannot realize the data's dynamic operation (such as update, append), some dynamic PDP schemes are proposed [6]–[11].

To verify the integrity of replica files stored by users on multiple CSPs at the same time, MR-PDP [12] is proposed, which proves that integrity verification on multiple replica

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Huang.

files simultaneously has a much higher efficiency compared with multiple integrity verification on a single replica. In the MR-PDP scheme, the DO calculates the tag set for the source file. This scheme provides a method to recover the data of the corrupted replica. If the CSP discards some data for some reasons (the CSP discards some uncommon files to save space or file loss due to server downtime), the DO can recover the discarded data with the help of the rest of the replicas, but this scheme cannot achieve the data's dynamic operation and does not support public verification. In [13], the authors proposed the MB-PMDDP scheme for the dynamic operation of replica files, in which the tag generation method is completely different from [12]. With the help of the MVT table, the MB-PMDDP scheme realizes the data's dynamic operation. However, because the tag set is closely related to the replicas, if some replicas are discarded or damaged, it inevitably leads to failure in the integrity verification phase, and the damaged replicas cannot be located. In [14], combining chaotic mapping with an AVL tree, this scheme realizes the data's dynamic operation. With the help of the AVL tree, this scheme greatly improves the performance of the data block search and reduces the difficulty and overhead of the data's dynamic operation. However, the tag generation method is similar to [13].

In the above three schemes, the tag generation method all requires complex modular exponentiation, which greatly increases the system's computational cost. In [15], [16], the authors propose their own schemes based on lattice vectors and algebraic signatures. In addition, for different problems, different schemes are put forward [17]–[21]. Therefore, how to design the tag generation method to save system calculation time and storage space usage and how to realize the dynamic operation of data blocks have become problems worthy of consideration.

A. MAIN CONTRIBUTION

In this paper, we propose a new dynamic multiple-replica provable data possession (DMR-PDP) scheme that can verify the integrity of replica files stored by users on multiple CSPs simultaneously. Compared with the previous schemes, our scheme has better computational cost performance and can save system storage space. The main contributions can be highlighted as follows:

(1) We present a dynamic multiple-replica provable data possession scheme, named DMR-PDP, which can verify the replica files' integrity simultaneously stored on multiple CSPs. In this scheme, the method of tag generation is based on the vector dot products instead of the modular power calculation, which greatly reduces the calculation time and the storage space.

(2) To realize the dynamic operation of data blocks, we propose a novel dynamic data structure, the divided address-version mapping table (DAVMT). With the help of DAVMT, the problem of the data block's update operation can be solved.

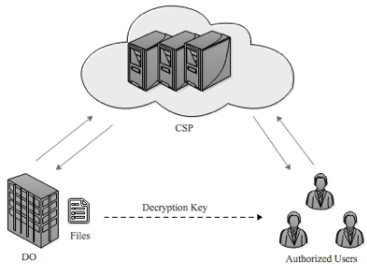

FIGURE 1. The system architecture.

(3) By comparing the experiment with MB-PMDDP, we verify the validity of the proposed scheme.

B. PAPER ORGANIZATION

The rest of the paper is organized as follows. In section II, we propose the system model and system components. The proposed scheme is explained in section III. The security analysis is shown in section IV. The experiment is presented in section V. Section VI is the summary of this paper.

II. PROBLEM STATEMENT

Here, the architecture and components of the system are described in detail.

A. SYSTEM MODEL

The system architecture is shown in Fig. 1 and consists of three entities:

(1) Data Owner (DO): Individuals/institutions/organizations who hold private data and need to store these data in the cloud.

(2) Cloud Service Provider (CSP): Corporation that provides data storage services for DOs.

(3) Authorized Users: Users who are authorized by DOs and have access to private data stored with the CSP.

B. SYSTEM COMPONENTS

The proposed scheme includes 8 algorithms: *KenGen*, *CopyGen*, *TagGen*, Pr *epareUpdate*, *ExecUpdate*, *Challenge*, *GenProof* , *VerifyProof* . The DO runs *KenGen*, *CopyGen*, *TagGen* and Pr *epareUpdate*. The CSP runs *ExecUpdate* and *GenProof* . The DO/Verifier runs *Challenge* and *VerifyProof* .

-*KenGen*: this algorithm is run by the DO to generate the public key (pk) and the secret key (sk).

-*CopyGen*: this algorithm is run by the DO to generate replica files, where the source file is $F = \{c_j\}_{1\le j\le m}$, the encrypted file is $\tilde{F} = \{b_j\}_{1\le j\le m}$, the number of replicas is n and the replicas set is represented as $\tilde{\mathbb{F}} = \{\tilde{F}_i\}_{1\le i\le n}$.

-*TagGen*: this algorithm is run by the DO to create the tag set. Input $\tilde{F}$ and sk, and output $T = \{\sigma_j\}_{1\le j\le m}$.

-Pr *epareUpdate*: this algorithm is run by the DO to update the replica files.

-*ExecUpdate*: this algorithm is run by the CSP. After receiving the update operation instruction from the DO, the

CSP performs the update operation on the specified data block.

-*Challenge*: this algorithm is run by the DO/Verifier to verify the integrity of the replica files.

-*GenProof*: this algorithm is run by the CSP. Input $\tilde{F}$, T and *chal*, and output P.

-*VerifyProof*: this algorithm is run by the DO/Verifier. Input P, pk and *chal*, and output $\{0/1\}$. 1 indicates validation is passed, and 0 indicates a failure.

III. PROPOSED SCHEME

This section includes 5 parts, and the proposed scheme is introduced in detail.

A. NOTATION

-F: the file F is divided into m blocks, $F = \{c_j\}_{1\le j\le m}$. Before generating the replica files, we encrypt F. The encrypted file is $\tilde{F} = \{b_j\}_{1\le j\le m}$ and the replica files are $\tilde{F}_i = \{b_{ij}\}_{1\le i\le n, 1\le j\le m}$.

-E_K: an encryption algorithm.

-$\mathcal{H}(\cdot)$: a hash function.

-$\pi_{key}(\cdot)$: a pseudo-random permutation (PRP) with *key*: $key \times \{0,1\}^{\log_2(m)} \to \{0,1\}^{\log_2(m)}$.

-$\psi_{key}(\cdot)$: a pseudo-random function (PRF) with *key*: $key \times \{0,1\}^* \to Z_P$.

- Bilinear Map/Pairing: G, G_T are multiplicative cyclic groups with the order p, g is the generator of G. A bilinear pairing $\hat{e} : G \times G \to G_T$ satisfies the following properties:

(1) Bilinearity: $\forall u, v \in G, a, b \in Z_P, \hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$;

(2) Non-degeberacy: $\hat{e}(g_1, g_2) \neq 1$;

(3) Computable: there is an efficient algorithm to calculate $\hat{e}$.

B. DIVIDED ADDRESS-VERSION MAPPING TABLE

In this paper, we proposed a novel divided address-version mapping table (DAVMT), which is a dynamic data structure, to implement update operations. With DAVMT, the DO can achieve the update operation of data blocks.

DAVMT contains two columns: the logic index (LI_j) is the logical index of the data blocks, and the block version (Ver_j) is the current version of the data block. When the data block is updated, the corresponding Ver_j adds one, and the initial Ver_j of all data blocks is 1. According to the total number of data blocks, the DAVMT can be divided into several child- tables. Note that DAVMTs are stored on the DO, and are irrelevant to the number of replicas. For example, assuming that the file has 18 data blocks, we can divide 3 child tables (DAVMTs) to realize the dynamic update operation, which is shown in Fig. 2.

C. SPECIFIC ALGORITHMS

KenGen : $\hat{e} : G \times G \to G_T$ is a bilinear pairing map, and g is a generator of G. The DO selects a group of random numbers $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_s), \alpha_k = \alpha^k, k \in [1, s]$, where s is the sector number of a block. Define $g_k = g^{\alpha^k} = g^{\alpha_k}, k \in [1, s]$. K_s, K_1 and s_o are the *key* of PRP, K_s is used in the replica generation phase, K_1 is used in the challenge phase, and s_o is used to generate the block tag. K_2 is the *key* of PRF and is used in the challenge phase. The DO runs *KenGen* to generate the public key $pk = (g_1, g_2, \ldots, g_s, K_1, K_2)$, and the corresponding secret key $sk = (\vec{\alpha}, K_s, s_o)$. The DO keeps sk secret and publishes pk.

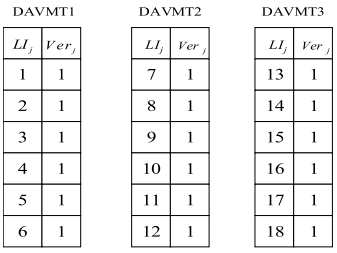

FIGURE 2. An example of 3 DAVMTs.

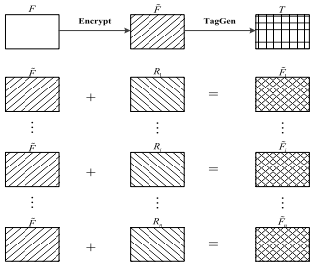

FIGURE 3. The process of generating the replica files.

CopyGen : The DO runs this algorithm to generate replica files. For a file, $F = \{c_j\}_{1\le j\le m}$ consists of m data blocks (every block consists of s sector, $c_j = \{c_{jk}\}_{1\le k\le s}$), the DO encrypts the file and the encrypted file is $\tilde{F} = \{b_j\}_{1\le j\le m} (b_j = E_K(c_j||j))$ and uses the $\tilde{F}$ to generate the replica files $\tilde{F}_i = \{u_{ij}\}_{1\le i\le n, 1\le j\le m}$, where the minimum unit of every replica is $u_{ijk} (u_{ijk} = b_{ijk} + r_{ijk}, r_{ijk} = \psi_{K_s}(i||j||k)_{1\le k\le s})$. From Fig. 3, we can learn how the replica files are generated.

TagGen : The DO runs this algorithm to generate the tag set. Note that the tag set is generated by $\tilde{F}$ and is independent of the number of replicas. We use vector dot products to generate block tags, which can save time for tag generation compared with modular exponentiation. First, DO carries out the PRP operation for $\vec{\alpha}$, and the result is $\vec{\beta}_j = PRP_{s_o}(j, \vec{\alpha})$. Then, the DO computes

$$\hat{\sigma}_j = \vec{\beta}_j \cdot \vec{b}_j = \beta_{j_1} \cdot b_{j_1} + \cdots + \beta_{j_s} \cdot b_{j_s} = \sum_{k=1}^{s} \beta_{j_k} \cdot b_{j_k}$$

and the block tag for b_j is $\sigma_j = \mathcal{H}(LL_j \parallel Ver_j \parallel j) \cdot \hat{\sigma}_j$; thus, the tag set is $T = \sum_{j=1}^{m} \sigma_j$. Finally, DO uploads all replicas $\tilde{F}_i$ and the tag set T to the CSP, keeps sk secret and deletes the local file.

Pr*epareUpdate* : the DO runs Pr*epareUpdate* algorithm to update data blocks. The detailed steps are in D. Update operations.

ExecUpdate : After receiving the update instruction of the data block, the CSP runs the *ExecUpdate* algorithm. The detailed steps are in D. Update operations.

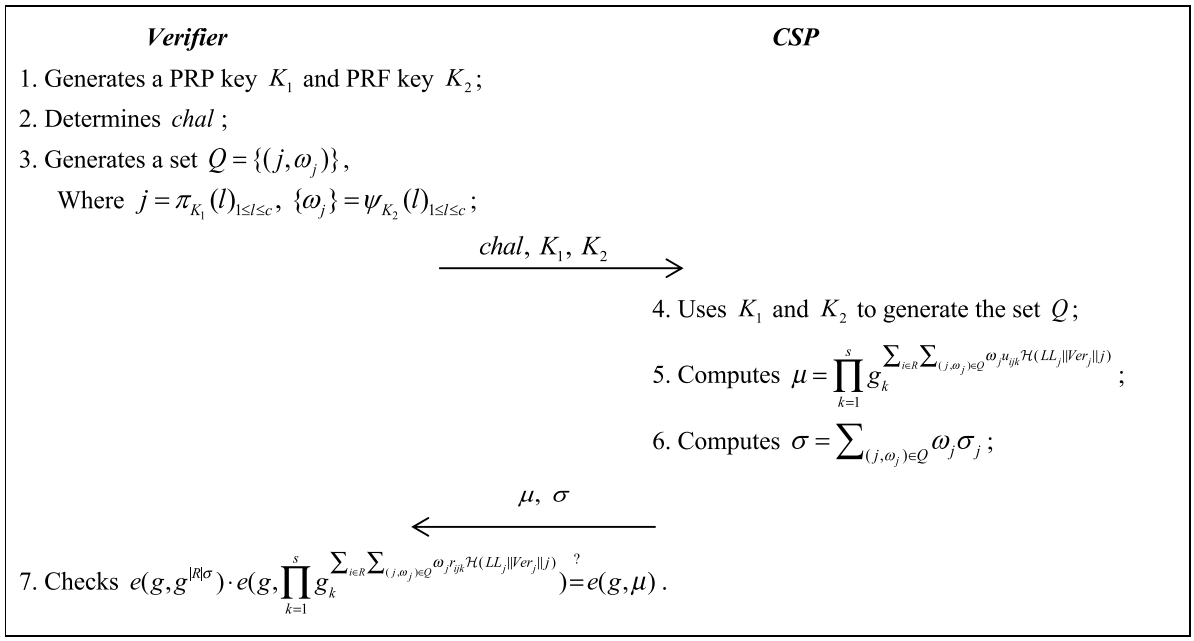

FIGURE 4. The challenge-response process between Do/Verifier and CSP.

Challenge : The DO/Verifier runs the *Challenge* algorithm to verify the integrity of the replica files. The DMR-PDP scheme provides two data integrity verification methods. The first method verifies the integrity of all replica files; the second method verifies the integrity of the data blocks specified in the replica files and is discussed in the proposed scheme. The DO/Verifier sends $chal = (R, c, K_1, K_2)$ to the CSP, where R is the challenged replicas set and c is the number of challenged blocks. K_1, K_2 are two fresh keys selected by the DO/Verifier in every challenge phase. K_1 is a key of $PRP(\pi)$ to generate random indices, and K_2 is a key of $PRF(\psi)$ to generate random values, where $j = \pi_{K_1}(l)_{1\le l\le c}, \{\omega_j\} = \psi_{K_2}(l)_{1\le l\le c}$. Both the DO and the CSP use K_1, K_2 to generate the challenge query set $Q = \{(j, \omega_j)\}$.

GenProof : The CSP runs the *GenProof* algorithm to prove the data blocks' integrity. After receiving *chal* from the DO/Verifier, the CSP uses K_1, K_2 to generate the challenge query set $Q = \{(j, \omega_j)\}$. Then, CSP generates the proof

$$\mu = \prod_{k=1}^{s} g_k^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j u_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)}$$

and $\sigma = \sum_{(i,\omega_j)\in Q}\omega_j\sigma_j$ to the DO/Verifier. The challenge-response process between the DO/Verifier and the CSP is given in Fig. 4.

VerifyProof : After receiving the response from the CSP, the DO/Verifier runs the *VerifyProof* algorithm to check whether the following equation holds:

$$e(g, g^{|R|\sigma})\cdot e(g, \prod_{k=1}^{s} g_k^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j r_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)}) \stackrel{?}{=} e(g, \mu), \tag{1}$$

$|R|$ is the size of the challenged replicas set. If equation (1) holds, the output "1" indicates that the CSP passes the check; Otherwise, the output will be "0". The demonstration process is as follows, $e(g, g^{|R|\sigma}) \cdot e(g, \prod_{k=1}^{s} g_k^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j r_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)})$, as shown at the bottom of the next page.

D. UPDATE OPERATIONS

For updating data blocks, the DO sends an instruction to the CSP, and the CSP performs update operations. We denote the update operations by BU.

♦ For an encrypted file $\tilde{F} = \{b_j\}_{1\le j\le m}$, assume that the DO wants to update a block b_j with b'_j, the DO runs the Pr*epareUpdate* algorithm, and the steps are as follows:

(1) The DO searches the logic index LI_j of the b_j in DAVMTs and updates $Ver_j = Ver_j + 1$;

(2) The DO creates a new block b'_j, where $b'_j = E_K(c'_j \parallel j)$ has s sectors; then, the DO computes u'_{ij} for every $\tilde{F}_i$, where $u'_{ijk} = b'_{ijk} + r_{ijk}$, and $r_{ijk} = \psi_{K_s}(i \parallel j \parallel k)_{1\le k\le s}$ remains unchanged.

(3) The DO computes the tag σ'_j of b'_j as follows:

$$\hat{\sigma'}_j = \vec{\beta'}_j \cdot \vec{b'}_j = \beta'_{j_1} \cdot b'_{j_1} + \cdots + \beta'_{j_s} \cdot b'_{j_s} = \sum_{k=1}^{s} \beta'_{j_k} \cdot b'_{jk},$$

$$\sigma'_j = \mathcal{H}(LI_j \parallel Ver_j \parallel j) \cdot \hat{\sigma'}_j.$$

(4) The DO sends the update instruction $< BU, j, \{u'_{ij}\}_{1\le j\le n, 1\le j\le m}, \hat{\sigma'}_j >$ to the CSP (this instruction means that the DO wants to modify b_j, j indicates the location of b_j, b'_j indicates a new block, and σ'_j is the tag of the new block).

♦ After receiving the instruction, CSP performs the *ExecUpdate* algorithm to update the block.

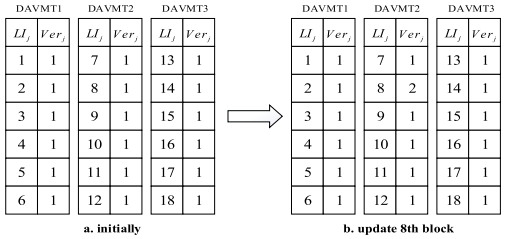

FIGURE 5. An example of a data update operation.

(1) CSP replaces u_{ij} with $u'_{ij}(u_{ij} \rightarrow u'_{ij})$, σ_j with $\sigma'_j(\sigma_j \rightarrow \sigma'_j)$.

CSP updates u_{ij} with u'_{ij} for replica, σ_j with σ'_j. The new replica file is $\tilde{F'}_i = \{u_{i1}, \ldots, u_{i(j-1)}, u'_{ij}, u_{i(j+1)}, \ldots, u_n\}_{1\le i\le n, 1\le j\le m}$.

(2) CSP returns "done" instruction to DO.

An example of update operations with the help of DAVMTs is shown in Fig. 5.

After the update operations are performed, the new replicas set is $\tilde{\mathbb{F}} = \{\tilde{F'}_i\}_{1\le i\le n}$, and the new tag set is $T' = \{\sigma_1, \ldots, \sigma_{j-1}, \sigma'_j, \sigma_{j+1}, \ldots, \sigma_n\}_{1\le j\le m}$.

E. FIND AND RECOVER CORRUPTED REPLICA

In the challenge phase, if the CSP fails to pass the integrity verification, it indicates that there are corrupted blocks in the replica files.

♦ How are corrupted replicas found?

When finding the CSP cannot pass the integrity verification in the challenge phase, the proposed scheme makes it possible to find the corrupted replicas and corrupted blocks. Because our scheme can challenge any number of replicas, we just need to adjust $chal = (R, c, K_1, K_2)$ several times and use the recursive divide-and-conquer method to lock the corrupted replicas and corrupted blocks.

♦ How can corrupted replicas be recovered?

The DO retrieves a correct replica and then recovers the corrupted replica with this correct replica. Assuming that the corrupted replica is q, u_{ijk} is the correct block. The steps are as follows:

(1) The DO computes $b_{jk} = u_{ijk} - r_{ijk}$;

(2) The DO computes $u_{qjk} = b_{jk} + r_{qjk}$;

(3) The DO sends the corrected replica to the CSP.

IV. SECURITY ANALYSIS

In this section, we analyze the correctness and security of our proposed scheme. In this paper, we assumed that the DO is fully trusted, but the CSP is not trusted and may maliciously corrupt data blocks.

A. THE CORRECTNESS ANALYSIS

Theorem 1: The Verifier can correctly verify the validity of outsourced data stored on the CSP.

$$
\begin{aligned}
& e(g, g^{|R|\sigma}) \cdot e(g, \prod_{k=1}^{s} g_k^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j r_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)}) \\
&= e(g, g^{|R|\sigma} \cdot \prod_{k=1}^{s} g_k^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j r_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)}) \\
&= e(g, g^{|R|\sigma} \cdot g^{\sum_{k=1}^{s}\beta_k\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j r_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)}) \\
&= e(g, g^{|R|\sigma+\sum_{k=1}^{s}\beta_k\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j r_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)}) \\
&= e(g, g^{\sum_{i\in R}\sigma+\sum_{k=1}^{s}\beta_k\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j r_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)}) \\
&= e(g, g^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j\sigma_j+\sum_{k=1}^{s}\beta_k\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j r_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)}) \\
&= e(g, g^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j(\sigma_j+\sum_{k=1}^{s}\beta_k r_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)}) \\
&= e(g, g^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j((\mathcal{H}(LI_j\|Ver_j\|j)\cdot\sum_{k=1}^{s}\beta_{jk}\cdot b_{jk}+\sum_{k=1}^{s}\beta_k r_{ijk}\mathcal{H}LI_j\|Ver_j\|j)}) \\
&= e(g, g^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j\mathcal{H}(LI_j\|Ver_j\|j)(\sum_{k=1}^{s}\beta_{jk}b_{jk}+\sum_{k=1}^{s}\beta_k r_{ijk})}) \\
&= e(g, g^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j\cdot\mathcal{H}(LI_j\|Ver_j\|j)\sum_{k=1}^{s}\beta_k u_{ijk}}) \\
&= e(g, g^{\beta_k\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j\cdot\mathcal{H}(LI_j\|Ver_j\|j)\sum_{k=1}^{s}u_{ijk}}) \\
&= e(g, \prod_{k=1}^{s} g_k^{\sum_{i\in R}\sum_{(j,\omega_j)\in Q}\omega_j u_{ijk}\mathcal{H}(LI_j\|Ver_j\|j)}) \\
&= e(g, \mu)
\end{aligned}
$$

Proof: The Verifier checks the proof received from the CSP. Based on the properties of bilinear mapping, if the output of equation (1) is "1", the correctness of our scheme is illustrated.

B. THE SECURITY ANALYSIS

Theorem 2 (Resisting Collusion Attack): The CSP cannot convince the DO to believe that they store all replicas, but they actually store only one replica.

Proof: In our scheme, the DO generates n replicas and stores them on the CSP, but the CSP cannot know the content of the replicas and only executes the storage service. Assuming that the CSP stores only one replica, it cannot pass the validation of the verifier in the verification phase. That is, the CSP cannot convince the DO to believe that they store all replicas, but they actually store only one replica.

Theorem 3 (Resisting Replace Attack): In the challenged phase, if the CSP uses another valid and uncorrupted data block to generate P instead of the challenged blocks, it cannot pass the Verifier's verification in the verification phase.

Proof: In the replica generation phase, the minimum unit for every replica is u_{ijk} and $u_{ijk} = b_{ijk} + r_{ijk}$, where $r_{ijk} = \psi_{K_s}(i \parallel j \parallel k)_{1 \le k \le s}$ is a random value related to index i, j, k. In the challenge phase, assume that the CSP replaces the index j' with challenge block j, r_{ijk} changes into $r_{ij'k}$ and u_{ijk} changes into $u_{ij'k}$ correspondingly. In the proof generation phase, the CSP runs the *GenProof* algorithm to generate $P = (\mu', \sigma')$ and response the Verifier, but it cannot pass the validation of the verifier in equation (1).

Theorem 4 (Detectability): Our verification scheme is $(\frac{m}{n}, 1 - (\frac{n-1}{n})^c)$ detectable if the CSP stores the file with n blocks, including m bad blocks (some corrupted or discarded blocks), and c blocks are challenged.

Proof: Assume that the CSP stores the file with n blocks, including m bad blocks. The number of challenged blocks is c. The bad blocks can be determined if and only if at least one of the challenged blocks chosen by the verifier matches the bad blocks. We use a discrete random variable X to denote the number of blocks selected by the challenger to match the bad block. We use PX to denote the probability that at least one bad block in the challenge set will be detected. So

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

We can obtain $PX \ge 1 - (\frac{n-1}{n})^c$. Therefore, our scheme is $(\frac{m}{n}, 1 - (\frac{n-1}{n})^c)$ detectable if the CSP stores the file with n blocks including m bad blocks, and c blocks are challenged.

V. EXPERIMENT

All experiments use OpenSSL (1.1.1d) and are conducted on a Windows 10 operating system with a 3.30 GHz Inter(R) Core (TM) i5 processor and 16 GB RAM. We use the type A elliptic curve of the PBC with 160-bit group order, and the security level can be equivalent to the 1024-bit RSA. The size of the outsourced file is 64 MB and the divided blocks are 4 KB.

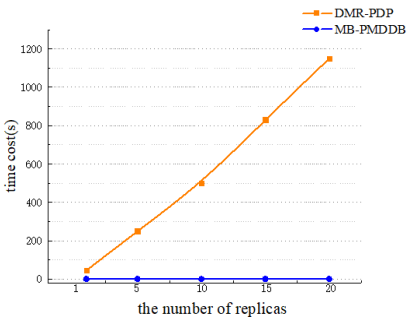

FIGURE 6. Comparison of the replica generation time between DMR-PDP and MB-PMDDB.

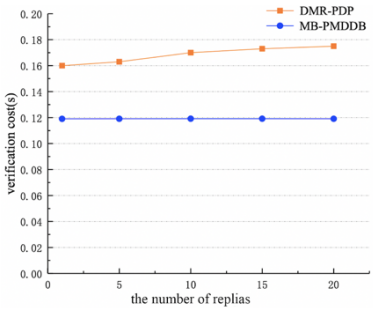

FIGURE 7. Comparison of the verification time between DMR-PDP and MB-PMDDB.

This section demonstrates the feasibility of our DMR-PDP scheme through experiments. For better evaluation, we choose the MB-PMDDB [11] scheme for comparison.

In the replica and tag processing phase, we generate tag set based on encrypted files and is irrelevant to the number of replicas. Fig. 6 shows the difference between DMR-PDP and MB-PMDDB [11] on the cost of tag generation when the number of replicas gradually increases. Due to different tag generation methods, regardless of the number of replicas, our DMR-PDP scheme takes approximately the same time. However, in the MB-PMDDB scheme, the time cost of tag generation increases with the number of replicas, and the relationship between them is linear. Therefore, our DMR-PDP scheme is more efficient than the MB-PMDDB scheme in terms of tag generation.

Fig. 7 displays the cost of verification time of the DMR-PDP and MB-PMDDB schemes when the number of replicas is different. Our scheme is slightly higher than the MB-PMDDB on the time cost, but the increase is tolerable.

Comparing the experiment with the MB-PMDDB scheme, we can verify the validity of the scheme proposed in this paper.

VI. CONCLUSION

At present, data security has become a hot research topic in cloud storage scenarios, and data integrity has become increasing worthy of attention. To verify the integrity of replica files stored by users on multiple CSPs simultaneously,

we proposed a new DMR-PDP scheme in this paper. Furthermore, we provided a new method for generating the tag set by utilizing vector dot products. Then, with the help of the dynamic data structure DAVMT, the problem of the data's dynamic operation was solved. Finally, the effectiveness of our proposed scheme was validated. In future work, we consider proposing a better data structure to achieve full dynamic operations. In addition, how to generate tag set to save calculation time and storage space is still a problem to be considered.

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 Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, Jun. 2009.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 54, no. 4, pp. 50–58, 2010.

[3] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *Proc. Int. Conf. Intell. Comput. Cognit. Informat.*, Kuala Lumpur, Malaysia, Jun. 2010, pp. 380–383.

[4] G. Anthes, "Security in the cloud," *Commun. ACM*, vol. 53, no. 11, pp. 16–18, 2010.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, Oct. 2007, pp. 598–609.

[6] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netw. (SecureComm)*, in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, New York, NY, USA, Sep. 2008, pp. 1–10.

[7] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote data checking using provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 1–34, May 2011.

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

[9] B. Chen and R. Curtmola, "Robust dynamic provable data possession," in *Proc. 32nd Int. Conf. Distrib. Comput. Syst. Workshops*, Macau, China, Jun. 2012, pp. 515–525.

[10] W. Guo, H. Zhang, S. Qin, F. Gao, Z. Jin, W. Li, and Q. Wen, "Outsourced dynamic provable data possession with batch update for secure cloud storage," *Future Gener. Comput. Syst.*, vol. 95, pp. 309–322, Jun. 2019.

[11] F. Wang, L. Xu, H. Wang, and Z. Chen, "Identity-based non-repudiable dynamic provable data possession in cloud storage," *Comput. Electr. Eng.*, vol. 69, pp. 521–533, Jul. 2018.

[12] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-replica provable data possession," in *Proc. 28th Int. Conf. Distrib. Comput. Syst.*, Beijing, China, Jun. 2008, pp. 411–420.

[13] A. F. Barsoum and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 485–497, Mar. 2015.

[14] M. Long, Y. Li, and F. Peng, "Dynamic provable data possession of multiple copies in cloud storage based on full-node of AVL tree," *Int. J. Digit. Crime Forensics*, vol. 11, no. 1, pp. 126–137, Jan. 2019.

[15] L. Li, Y. Yang, and Z. Wu, "FMR-PDP: Flexible multiple-replica provable data possession in cloud storage," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Heraklion, Greece, Jul. 2017, pp. 1115–1121.

[16] H. Hou, J. Yu, and R. Hao, "Provable multiple-replica dynamic data possession for big data storage in cloud computing," *Int. J. Netw. Secur.*, vol. 20, no. 3, pp. 575–584, May 2018.

[17] Z. Deng, S. Chen, X. Tan, D. Song, and F. Wu, "An efficient provable multi-copy data possession scheme with data dynamics," in *Proc. Int. Conf. Secur., Privacy Anonymity Comput., Commun. Storage*, in Lecture Notes in Computer Science, Melbourne, NSW, Australia, Dec. 2018, pp. 392–402.

[18] A. F. Barsoum and M. A. Hasan, "Integrity verification of multiple data copies over untrusted cloud servers," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGrid)*, Ottawa, ON, Canada, May 2012, pp. 829–834.

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

[20] J. Wei, J. Liu, R. Zhang, and X. Niu, "Efficient dynamic replicated data possession checking in distributed cloud storage systems," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 1, Jan. 2016, Art. no. 1894713.

[21] H. Zhang, Z. Cao, X. Dong, and J. Shen, "Proof of multicopy via proof of file position in cloud," *Fundamenta Informaticae*, vol. 157, nos. 1–2, pp. 141–151, Jan. 2018.

[22] H. Wang and Y. Zhang, "On the knowledge soundness of a cooperative provable data possession scheme in multicloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 264–267, Jan. 2014.

[23] S. Peng, F. Zhou, Q. Wang, Z. Xu, and J. Xu, "Identity-based public multi-replica provable data possession," *IEEE Access*, vol. 5, pp. 26990–27001, 2017.

[24] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Trans. Cloud Comput.*, early access, Jul. 16, 2019, doi: 10.1109/TCC.2019.2929045.

[25] J. Xu, A. Yang, J. Zhou, and D. S. Wong, "Lightweight delegatable proofs of storage," in *Proc. 21st Eur. Symp. Res. Comput. Secur.*, in Lecture Notes in Computer Science, Heraklion, Greece, Sep. 2016, pp. 324–343.

[26] J. Huang, J. Zou, and C.-C. Xing, "Competitions among service providers in cloud computing: A new economic model," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 866–877, Jun. 2018.

[27] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS)*, Chicago, IL, USA, Nov. 2009, pp. 213–222.

[28] J. Huang, Q. Duan, S. Guo, Y. Yan, and S. Yu, "Converged network-cloud service composition with end-to-end performance guarantee," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 545–557, Apr. 2018.

YILIN YUAN received the B.S. and M.S. degrees from the College of Computer and Information Engineering, Henan Normal University, Xinxiang, China, in 2015 and 2018, respectively. She is currently pursuing the Ph.D. degree with the College of Computing, Beijing University of Technology, Beijing, China. Her research interests include cloud security and trusted computing.

JIANBIAO ZHANG (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from Northwestern Polytechnic University, Xi'an, China, in 1992, 1995, and 1999, respectively. From 1999 to 2001, he was a Postdoctoral Fellow with Beihang University, Beijing, China. He is currently a Professor and a Ph.D. Supervisor with the Faculty of Information Technology, Beijing University of Technology. He has published over 80 journal/conference papers. His research interests include network and information security, and trusted computing.

WANSHAN XU received the B.S. degree from the Department of Computer Engineering, Taiyuan Institute of Technology, Shanxi, China, in 2011, and the M.S. degree from the College of Computing, Beijing University of Technology, Beijing, China, in 2018, where he is currently pursuing the Ph.D. degree. His research interests include trusted computing and block chain.