Verification Algorithm for the Duplicate Verification Data with Multiple Verifiers and Multiple Verification Challenges

Guangwei Xu[1], Miaolin Lai[1], Xiangyang Feng[1], Qiubo Huang[1], Xin Luo[1], Li Li[2*], and Shan Li[1]
[1] School of Computer Science and Technology, Donghua University
[e-mail: gwxu@dhu.edu.cn, lerrrs@163.com, fengxy@dhu.edu.cn, huangturbo@dhu.edu.cn, xluo@dhu.edu.cn, dhulishan@163.com]
[2] College of Architecture and Urban Planning, Tongji University
Shanghai, 201620, China
[e-mail: lilicaup@tongji.edu.cn]
*Corresponding author: Li Li

Received June 24, 2019; revised September 12, 2019; revised July 16, 2020; accepted January 28, 2021; published February 28, 2021

Abstract

The cloud storage provides flexible data storage services for data owners to remotely outsource their data, and reduces data storage operations and management costs for data owners. These outsourced data bring data security concerns to the data owner due to malicious deletion or corruption by the cloud service provider. Data integrity verification is an important way to check outsourced data integrity. However, the existing data verification schemes only consider the case that a verifier launches multiple data verification challenges, and neglect the verification overhead of multiple data verification challenges launched by multiple verifiers at a similar time. In this case, the duplicate data in multiple challenges are verified repeatedly so that verification resources are consumed in vain. We propose a duplicate data verification algorithm based on multiple verifiers and multiple challenges to reduce the verification overhead. The algorithm dynamically schedules the multiple verifiers' challenges based on verification time and the frequent itemsets of duplicate verification data in challenge sets by applying FP-Growth algorithm, and computes the batch proofs of frequent itemsets. Then the challenges are split into two parts, i.e., duplicate data and unique data according to the results of data extraction. Finally, the proofs of duplicate data and unique data are computed and combined to generate a complete proof of every original challenge. Theoretical analysis and experiment evaluation show that the algorithm reduces the verification cost and ensures the correctness of the data integrity verification by flexible batch data verification.

Keywords: Cloud Storage, Data Integrity, Multiple Challenges, Frequent Itemsets

The work was sponsored by the Natural Science Foundation of Shanghai (Nos.19ZR1402000 and 17ZR1400200), the Education and Scientific Research Project of Shanghai (C160076), the National Natural Science Foundation of China (Nos.61772018 and 61772128), and the Seed Funds of the Key Laboratory of Ecology and Energy-saving Study of Dense Habitat of Ministry of Education (Tongji University).

http://doi.org/10.3837/tiis.2021.02.010 **ISSN : 1976-7277**

1. Introduction

With the rapid development of the cloud computing, cloud storage as a new generation of computing infrastructure has received more and more attention. It is a distributed computing model for sharing virtualized computing resources, such as storage, CPU, applications and services. Cloud computing brings many benefits to users, and users' demand for cloud storage services is also increasing. Some recent reports indicate that more than 79% of organizations need to use data outsourcing, the demand for cloud storage services is increasing [1-3]. Cloud computing is attractive as a cost-effective and high-performance model. However, the trustworthiness and reliability of cloud infrastructure has increasingly aroused people's concern, the outsourced data or the cloud infrastructure often encounters security issues [4-6]. Cloud computing appears as a black-box to the users, which is beneficial to cloud service providers (CSP) for management and security purposes. Unfortunately, the black-box nature introduces the lack of transparency for CSP's activities, which results in distrust and lack of accountability of clouds [7]. In result, data owners worry about whether outsourced data is intact or corrupted on remote storage servers because they are deprived of direct control over the data [8-10]. In order to solve these problems, many solutions on the data integrity verification have been proposed to verify the integrity of remote data storage [3-16].

Some of existing data integrity verification schemes introduce third-party verifier (TPA) to execute the fair and professional verification of data stored in cloud storage space [13-20]. However, it's very hard to seek a completely credible third-party verifier in practice. Although the data owner who publishes the data sharing service in the cloud has the responsibilities and obligations to ensure the integrity of shared data for the data users, the fairness of the verification executed by the data owner is questioned since the verification results provided by third party verifier is more convincing. Facing such a difficult situation, Mauro et al. [12] proposed an idea to verify the data integrity in the data sharing service. In the context of data sharing service scenario, the cloud service provider provides a way for the data owner, e.g., most companies or individuals, to store their data into the cloud, such as computer software, data documents, and so on. Users authorized by the data owner can download these data over the Internet. Certainly, before a user downloads the data over the Internet, he is very concerned about whether the data has been corrupted or tampered, otherwise, he will waste the transmission cost in vain. Thus, the integrity of the data is very important for the user so that he will perform the data integrity verification in person before downloading the data [12]. In this case, the users are both the users of data and the verifier of data since each users is a third party independent of the data owner and the cloud service provider. However, since many users often download the same data in cloud data sharing service at a similar time, e.g., hot data, it causes high data verification overhead for the cloud service provider in a short time. Moreover, the existing verification schemes only consider that one or more verification requests are launched by one TPA [13-20]. In order to provide data sharing services for users, the data owner needs to rent the service resources from the cloud and bear the rental cost accordingly. Therefore, when multiple verifiers launch multiple data verification requests for the same data at a similar time, multiple challenges with the same data require CSP to repeatedly calculate the data proofs and tag proofs of the same checked data. The duplicate verification data not only increase the unnecessary verification overhead, but also put a burden on the data owner and the cloud service provider. In particular, data owner cares about computational cost consumed during the verification, since data verification is not free for the verifiers and CSP, and the bills for data proofs and tag proofs computed by CSP during the verification process will be paid by the data owner. To reduce the verification cost,

this paper proposes a verification scheme for the duplicate verification data with multiple verifiers and multiple verification challenges. Our main contributions are summarized as follows:

1) We propose a data verification algorithm for multiple verifiers and multiple verification challenges (MT-DVA) to reduce the unnecessary overhead caused by duplicate data verification. Considering that the role of TPA will be assumed by the data users, TPA refers to the users in the following paper. The algorithm dynamically schedules the challenge set according to the deadline of the challenges requested by TPA. Then it restructures the verification tasks according to the same verification data object.

2) We use the FP-Growth algorithm to extract the frequent itemsets from different challenge data sets. Then, each original challenge data set is split according to frequent itemsets which are extracted before, where the data proof and tag proof of duplicate parts and that of different parts are calculated respectively.

3) We combine the duplicate part and different part according to the original challenge data sets which are requested by different TPAs. The verification proof including data proofs and tag proofs will be formed, and then is used by the verification model to verify the integrity of the data.

The rest of this paper is organized as follows. In Section 2, we briefly summarize the current research on data integrity verification. Section 3 outlines the system model and problem statement. In Section 4, we present a verification algorithm for the duplicate verification data with multiple TPAs and multiple verification challenges. In Section 5, we analyze our scheme. Section 6 evaluates the performance of the proposed algorithm. Finally, Section 7 gives the conclusions of the work and future directions.

2. Related Work

At present, the integrity verification schemes for remote storage data can be divided into two types, i.e., provable data possession (PDP) [13] and proof of retrievability (POR) [17].

1) Data verification scheme

The data integrity is recently focusing on the remote data storage. Since Ateniese et al. [13] and Shacham et al. [17] proposed the provable data possession (PDP) and the proof of retrievability (POR), respectively, many verification schemes based on the two main mechanisms are improved to guide the current data integrity verification. Without doubt, the two types of verification schemes have one thing in common that POR can be implemented by integrating the PDP and the data recovery technology.

Erway et al. [14] proposed a PDP scheme of a rank-based skip list to support the data dynamic operation. Shacham and Waters [15] used the BLS short message signature mechanism [16] to construct a verification tag based on homomorphic encryption technology, which proved to be safe in a strong threat environment. Yu et al. [10] proposed a new construction of identity-based RDIC protocol by making use of key-homomorphic cryptographic primitive to reduce the system complexity and the cost for establishing and managing the public key authentication framework in PKI-based RDIC schemes. Yang et al. [20] proposed a public auditing protocol for shared cloud data to support identity privacy and identity traceability.

The integrity verification schemes above will cost a lot of computation overhead. Although the verification proof is generated by the cloud server, the resource cost on the cloud server is not free. The more resource the verification uses, the more expensive bill the data owner will

receive. If a large number of users download the shared data on the CSP, the verification cost is also expensive. In particular, when multiple users challenging the same data at a similar time, the cloud server cannot but calculate the proof again and again, even though there exists same part in different challenges. However, these algorithms only consider that one TPA challenges the integrity of the data on the cloud server.

2) Execution of the data verification

In the cloud data integrity verification, [13] used the method of randomly extracting data blocks, and CSP calculated the proof once a challenge arrived. To increase efficiency and achieve the scalability of public auditing, TPA should handle multiple audit requests from different users in a fast and cost-effective way, which supports batch auditing [21-23]. Hao Yan et al. [24] mentioned in the scheme that TPA selects two random numbers k_1 and k_2 in data verification and sends them to the cloud server. Then CSP generates a challenge set C according to k_1, k_2. In [25], since the index number of the data block at the random position of each spot check is fixed in advance, and the hash value is calculated by cascading the data blocks, the fast positioning and repair of the data cannot be supported.

The above schemes only deal with one or more verification requests launched by one TPA. In contrast to the above solutions, in this paper, we design a data verification algorithm for multiple verifiers and multiple verification requests (i.e., multi-batch data). While the same data are extracted by different verifiers in a certain period of time, the same part of verified data is only calculated once.

3. System Model and Problem Statement

3.1 System Model

In the data storage service, a traditional verification model [14] is generally composed of the data owner (DO), the cloud service provider (CSP), and the third party verifier (TPA) who refers to the user in this paper. As shown in **Fig. 1**, DO rents CSP's resources to publish his data as a service for users. Users must verify the data integrity before downloading or using the shared data. Otherwise, Once the downloaded data have been corrupted, users will waste their precious network resources in vain. Moreover, as CSP's computation resources are not free, the data verification cost is borne by the data owner. The process of data integrity verification is mainly composed of five polynomial algorithms as follows.

1) $KeyGen(\lambda) \rightarrow (sk, pk)$. The data owner enters the security parameter λ, and then executes a probabilistic key generation algorithm to generate one public key pk and one private key sk.
2) $TagGen(F, sk) \rightarrow T$. The data owner divides the file *F* into *n* data blocks. Then he uses the private key sk to calculate the corresponding data tag t_i for each data block. Finally, he uploads the file $F$and one tag set $T = \{t_i\}_{i\in[1,n]}$ to CSP.
3) $Chall(F) \rightarrow C$. TPA randomly selects the index numbers of $c(c \leq n)$ verified data blocks and puts them into set. Based on this, TPA generates a corresponding random number v_i in $\mathbb{Z}_p$ for each data block, and then forms a challenge $C = \{(i, v_i)_{i\in[1,n]}\}$. TPA finally outputs the challenge C to CSP.
4) $ProofGen(F, T, C) \rightarrow P$. The CSP responds to the challenge and uses c extracted data blocks and corresponding data tags stored in his own storage space, and the challenge C to calculate the verification proof P. Finally, P is returned to the TPA.

5) $VerifyData(C, P, pk, T) \rightarrow 0/1$. The TPA uses the received verification proof P, the public key pk, the challenge C to determine the integrity of the challenged data blocks, and outputs whether these data blocks are intact or not.

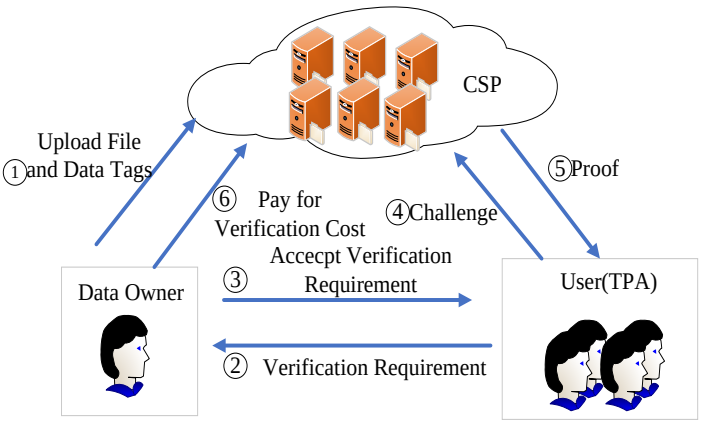
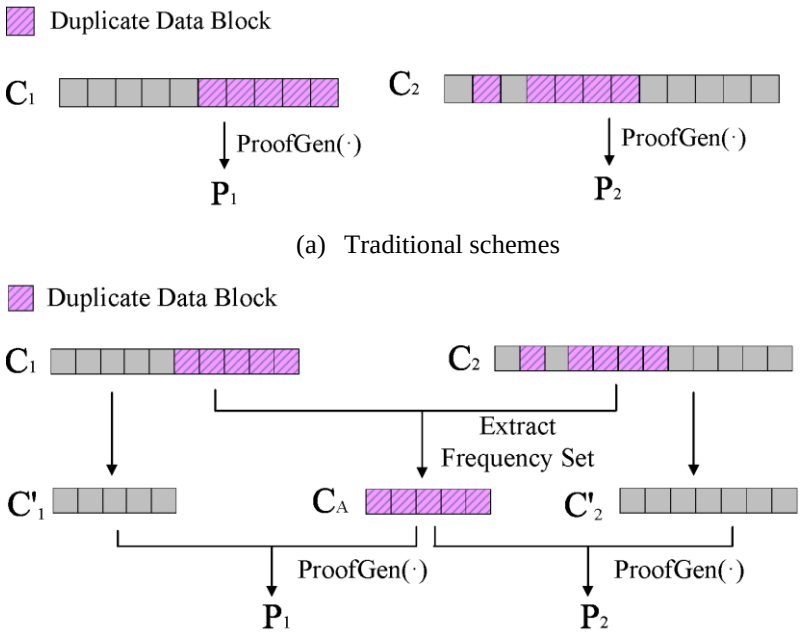


Fig. 1. Data Integrity Verification Model

3.2 Motivation

In traditional verification scheme, after CSP receives the challenges C_1 and C_2 from TPA_1 and TPA_2, he will execute $ProofGen(F, T, C_1)$ and $ProofGen(F, T, C_2)$ in order. As shown in **Fig. 2(a)**, the proofs of 10 data blocks and 12 data blocks are generate for the challenge C_1 and the challenge C_2 respectively. Assume that the time of generating the proof for one block is t. The time of executing $ProofGen(F, T, C_1)$ and $ProofGen(F, T, C_2)$ is $10\,t$ and $12\,t$ respectively. The total time of the proof generation for the challenges C_1 and C_2 is $22t$. In **Fig. 2(a)**, there are the same verification data in the challenges C_1 and C_2 (shaded data blocks in **Fig. 2**). The part of same data blocks C_A *is* first extracted, and their proof is calculated separately. Then the part of the different data blocks C_1', namely the rest part that C_1 removes C_A, is also calculated separately. The part C_2' which is the rest part of C_2 removing C_A is also calculated separately. After all these are done, C_A and C_1' are combined together to form the original challenge C_1. Also C_A and C_2' are combined into C_2. Finally, the proof P_1 corresponding to C_1 and the proof P_2 corresponding to C_2 are gotten respectively. In this way, the total calculation time is shorten to only $17t$, which is $5t$ less than the traditional verification schemes. When the number of TPAs is large or the same verification data are large, CSP needs to repeatedly calculate the duplicate data in their challenges to generate the data proofs and tag proofs, leading to wasting the computational resources.

Therefore, the authors consider extracting the same verification data from different challenges and compute them separately so that the duplicate data are only calculated once, as shown in **Fig. 2(b)**. After extracting the duplicate data, the CSP usually calculates the proof of 22 data blocks, and now only calculates the proof for 17 data blocks. Therefore, if the duplicate data in multiple challenges are dealt with properly under certain conditions, the verification cost can be reduced obviously.



(a) Traditional schemes

(b) Splitting the challenged data by the same and different data blocks

Fig. 2. Proof Generation

3.3 Problem Statement

In the system model and motivation described above, when multiple TPAs challenge the integrity of multiple verification data at a similar time, the verification needs to confront and overcome the following problems.

1) The duplicate verification data which are in the multiple verification challenges results in unnecessary verification overhead.
2) The correctness of the proof of different challenges must be ensured after these challenges are split into different parts and each proof is generated for each part separately.

In this paper, our goal is to propose an efficient data verification algorithm to solve the above problems, and reduce the verification cost.

4. Verification Algorithm for the Duplicate Verification Data with Multiple Verifiers and Multiple Verification Challenges

Since each challenge has a release time, one challenge data set is equivalent to the data set in the task. Let each challenge be each task. Therefore, referring to the definition of the task [26-30], we define each challenge C_j ($j \in [1, m]$) as follows.

Definition 1. Challenge C_j. After a TPA generating challenge data set $\Omega_j = \{(j, v_j)_{j\in[1,m]}\}$, he constructs the challenge $C_j = \{r_j, d_j, \Omega_j\}$, where r_j is the release time of C_j, and d_j is the deadline of C_j.

Definition 2. Challenge Set. When one or multiple users launch multiple challenges, a challenge set $C = \{C_1, C_2, \ldots, C_m\}$ is constructed, where m is the total number of the challenges.

As described above, CSP may receive a large number of challenges in a certain period of time, and simultaneously there are parts of the same challenge data among these challenges. In this paper, we first dynamically classify the challenge set according to the deadline of challenges to decide whether there exist the duplicate data in multiple challenges. Then we extract the frequent itemsets from the classified challenge set to construct a new set, and calculate these set separately. Finally, we combine the new set into the original challenges, and generate the final proof P_j and send it to the corresponding TPA. Our algorithm separately computes the duplicate challenge data requested by different TPAs to reduce the verification overhead.

4.1 Possibility of Duplication Data Occurring in Multiple Challenges

In data integrity verification, when multiple TPAs send multiple challenges to CSP, we dynamically schedule the challenges in a certain period of time, and then extract the duplicate data in these challenges.

Suppose there are n data blocks stored by the CSP. During the time interval Δt, the CSP receives m challenges, and the number of indexes in the challenge C_j is num_{C_j}. If $num_{C_j} \cdot m > n$, it must have $C_x \cap C_y \neq \emptyset$, where $x, y \in [1, m]$. That is, the same data blocks are verified twice. If $num_{C_j} \cdot m > n$, let event A denote the challenges existing the duplicate indexes. The probability of the occurrence of the event A is

$$P(A) = 1 - P(\bar{A}) = 1 - \frac{\prod_{j=1}^{m} C_{n-\sum_{j=1}^{m} num_{C_j}}^{num_{C_j}}}{\prod_{j=1}^{m} C_n^{num_{C_j}}} = 1 - \frac{A_{c-num_{C_1}}^{n-\sum_{j=1}^{m} num_{C_j}}}{A_n^{num_{C_2}} \times \cdots \times A_m^{num_{C_m}}}. \quad (1)$$

If $num_{C_j} \cdot m \ll n$, we can increase m by extending the time interval Δt. Thus, the probability $P(A)$ of the event A can be increased.

Definition 3. Computation time of challenge. Suppose a challenge has a_1 data block indexes and its calculation time of proof generation is $C_p(a_1)$. In addition, another challenge with a_2 data block indexes has the calculation time of proof generation is $C_p(a_2)$. Therefore, if the challenge C_j has b data block indexes, the calculation time of proof generation for the challenge C_p is

$$C_p(C_j) = b \times \beta_p + \epsilon_p, \quad (2)$$

where $\beta_p = \frac{C_p(a_1) - C_p(a_2)}{a_1 - a_2}$, $\epsilon_p = \frac{a_1 C_p(a_2) - a_2 C_p(a_1)}{a_1 - a_2}$.

Definition 4. The latest execution time of challenge. Suppose that the challenge C_j has b data block indexes, then the latest execution time of the challenge C_j is

$$S_j' = d_j - C_p(C_j). \quad (3)$$

Assume that TPA launches a challenge C_j at time t_j, CSP calculates the latest execution time S_j' for C_j. If $S_j' = t$, CSP generates the proof of challenge C_j immediately, where t is the current time. If $S_j' > t$, C_j will be added to challenge set C and wait for more challenges to

arrive between time t_j and t. For each challenge C_j we will repeat this process. If the number of challenges satisfies $m_C > m'$ before t, we will sort each challenge C_j in C according to the latest execution time S_j', where m_C is the number of challenges in C and m' is its threshold. Then we further deal with the data set in each challenge C_j. Otherwise, if $m_C < m'$, the proof of C_j is directly calculated according to S_j' without other operations.

4.2 Extraction of the Duplicate Verification Data in Multiple Challenges

Assume there exists the duplicate verification data in the challenge set C. Then, the frequent itemsets of the duplicate verification data are extracted, and are sorted according to the order of the original challenge C_j.

1) Frequent itemset computation

Given the challenge set $C = \{C_1, C_2, \ldots, C_m\}$ that meets the requirements in Section 4.1. To extract duplicate data block indexes, we use the FP-Growth algorithm [18] to mine frequent itemsets in the challenge set.

Set a minimum support ξ, and build an FP tree for all index values in Ω_j, where Ω_j is the verified data set in the challenge C_j.Assume that there exists the same challenge index h between two different data sets Ω_1 and Ω_2. The random number corresponding to the index in Ω_1 cannot be equal to that in Ω_2. Therefore, when constructing the FP tree according to the index in the challenge, it is necessary to store the random number v_i of the index which is duplicate in more than one challenge, where v_i is the corresponding random number of the index in set Ω_1. In the process of recursively mining the new frequent itemsets, if a new frequent item is already in the frequent itemsets, it would be deleted. Otherwise, the new frequent item is added to the frequent itemsets. We finally get d frequent itemsets $\overline{\Omega}_k'$ ($k \in [1, d]$), and form d new challenges for $\overline{\Omega}_k'$.

2) Sorting frequent itemsets

As we sort each challenge C_j in the challenge set C, we also sort d frequent itemsets $\overline{\Omega}_k'$ similarly according to the latest execution time of their corresponding challenges. For example, given Ω_1, Ω_2, and Ω_3, and there exit two frequent itemsets $\overline{\Omega}_1$ and $\overline{\Omega}_2$ among Ω_1, Ω_2, and Ω_3. Supposed that $\overline{\Omega}_1 \subseteq \Omega_1$, $\overline{\Omega}_1 \subseteq \Omega_3$, $\overline{\Omega}_2 \subseteq \Omega_2$, and $S_1' < S_2' < S_3'$. As $S_1' < S_2'$, the deadline of Ω_1 is closer. When calculating the proof of each item in the frequent itemsets, we first calculate $\overline{\Omega}_1$ and then $\overline{\Omega}_2$.

4.3 Split of Verification Data and Challenge Proof

After extracting the frequent itemsets from the challenges, we split each original challenge into duplicate data set and rest data set according to extracted frequent itemsets. Then we compute the proofs of the two data sets respectively.

4.3.1 Verification Data Split based on Frequent Itemsets

We split the data set Ω_j in each challenge C_j into two sets, i.e., $\Omega_j = \Omega_j' + \overline{\Omega}_k'$, where $\overline{\Omega}_k'$ ($\overline{\Omega}_k' \subseteq \Omega_j$) is the duplicate data set, and Ω_i' is the rest data set. The index number of frequent itemsets of $\overline{\Omega}_k'$ is stored as $Index_{\overline{\Omega}_k'}$ corresponding to $\overline{\Omega}_k'$ through the array. The detailed process of challenge split is shown in **Fig. 3**. First, the CSP forms a challenge queue for all challenges sent by the TPA in the $Chall(\cdot)$. Next, the CSP extracts frequent itemsets in challenge set, and generates new challenges (i.e., C_A and C_B) for each set in frequent itemsets.

Finally, the challenges C_A and C_B are added to the challenge queue, and the challenges C_A and C_B are executed first.

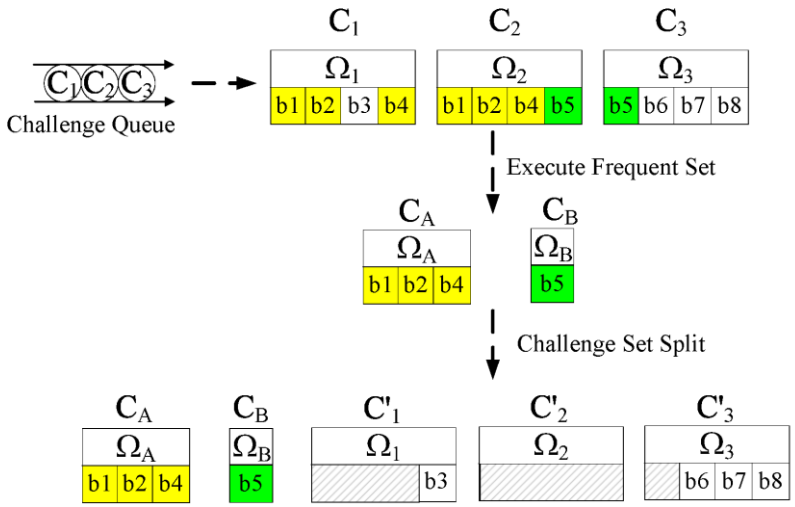


Fig. 3. Challenge Split

4.3.2 Proof Generation

For each set in the frequent itemsets $\overline{\Omega}_k^{'}$, the CSP first calculates the data proof $DP_{\overline{\Omega}_k^{'}}$ by

$$DP_{\overline{\Omega}_k^{'}} = e(u, pk)^{\sum_{k=1}^{d}(B_k \cdot \frac{\sum v_k}{num_{V_k}})}, \tag{4}$$

Where B_k is the data block, and num_{V_k} is the number of elements v_k. Then, for each set $\overline{\Omega}_k^{'}$, CSP computes tag proof $TP_{\overline{\Omega}_k^{'}}$ by

$$TP_{\overline{\Omega}_k^{'}} = \prod_{k\in[1,d]} \sigma_k^{\frac{\sum v_k}{num_{V_k}}}. \tag{5}$$

After CSP completes the proof computation of frequent itemset $\overline{\Omega}_k^{'}$, the data proof and the proof of frequent items are computed. The data proof $DP_{\Omega_j'}$ of $\Omega_j^{'}$ is

$$\mathrm{DP}_{\Omega_j^{'}} = e(u, pk)^{\sum_{j\in\Omega_j^{'}} B_j v_j}, \tag{6}$$

and the tag proof $TP_{\Omega_j^{'}}$ is

$$\mathrm{TP}_{\Omega_j^{'}} = \prod_{j\in\Omega_j^{'}} \sigma_j^{v_j}. \tag{7}$$

4.4 Combination of Verification Data and Verification Proof in Each Challenge

After computing the verification proofs of the duplicate data and the rest data, we recombine the duplicate data and the rest data into the original data set Ω_j. Also, we recombine the verification proofs of the duplicate data and the rest data into the verification proof P_j of the challenge C_j.

4.4.1 Combination of Verification Data

We first combine the challenges together the deadline of which is closed to each other. As shown in **Fig. 4**, when we combine the challenge data set Ω_j, we find the set in frequent itemsets by the index $Index_{\bar{\Omega}_k'}$. In this way, we can combine $\Omega_j^{'}$ and $\bar{\Omega}_k^{'}$ into original challenge data set Ω_j.

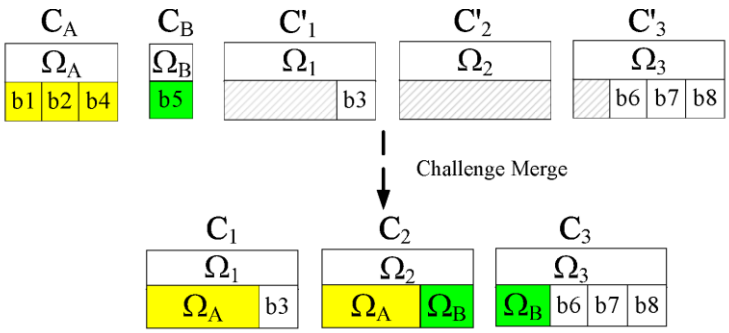


Fig. 4. Challenge Data Combination

4.4.2 Combination of Verification Proof

After combining the verification data, we find the index of frequent itemsets of Ω_j in $Index_{\bar{\Omega}_k'}$ to get the data proof and tag proof of corresponding frequent itemsets. After that, CSP computes the data proof $DP_{\bar{\Omega}_k'}$ of $\bar{\Omega}_k^{'}$ by

$$DP_{\bar{\Omega}_k'} = \prod_{\bar{\Omega}_j \in \bar{\Omega}_k'} DP_{\bar{\Omega}_j}. \tag{8}$$

CSP computes the tag proof $TP_{\bar{\Omega}_k'}$ of $\bar{\Omega}_k^{'}$ by

$$TP_{\bar{\Omega}_k'} = \prod_{\bar{\Omega}_j \in \bar{\Omega}_k'} TP_{\bar{\Omega}_j}. \tag{9}$$

For each Ω_j, CSP computes the data proof DP_{Ω_j} of Ω_i by

$$DP_{\Omega_j} = DP_{\Omega_j'} \cdot DP_{\bar{\Omega}_j'}, \tag{10}$$

and the tag proof TP_{Ω_j} by

$$TP_{\Omega_j} = TP_{\Omega_j'} \cdot TP_{\bar{\Omega}_j'}. \tag{11}$$

Finally, CSP returns the data integrity proof $P_j = (DP_{\Omega_j}, TP_{\Omega_j})$ and the set of random number $V_{\bar{\Omega}_k'}$.

4.4.3 Proof Check

After the TPA receives the data integrity proof P_j and the random number set $V_{\bar{\Omega}_k'}$ sent by the CSP, TPA verifies the proof by

$$DP_{\Omega_j} \cdot e(\prod_{j\in\Omega_j'} H(B_{id})^{v_j} \cdot \prod_{k\in\bar{\Omega}_k'} H(B_{id})^{\frac{\sum v_k}{num_{V_k}}}, pk) = e(TP_{\Omega_j}, g), \quad (12)$$

where B_{id} is the identity of the data block B_i. If the above formula holds, the outsourced data in CSP's storage space are intact; otherwise, the data are corrupted.

4.5 Verification Algorithm for the Duplicate Verification Data with Multiple Verifiers and Multiple Challenges

The proposed algorithm not only verifies the outsourced data integrity, but also reduces the unnecessary verification overhead caused by the duplicate data. In this paper, we improve $ChallGen(F)$, $ProofGen(F, \Phi, C_j)$, and $Verify(\Omega_j, P_j)$.

1) $ChallGen(F) \rightarrow \Omega_j$. TPA selects c data blocks in the data file F to launch a challenge, and generates $\Omega_j = \{(j, v_j)\}$. Then TPA outputs the challenge $C_j = \{r_j, d_j, \Omega_j\}$ to CSP.
2) $ProofGen(F, \Phi, C_j) \rightarrow (P_j)$. CSP computes the latest execution time S_j' of the challenge C_j according to formula (3). If the number of the challenges satisfies $m_C > m'$ before t, we sort the items in challenge set C according to S_j', computing C_i with a closer S_j' first. CSP extracts the challenge data set Ω_j in C_j, performs the extraction of the duplicate verification data, splits and combines the verification data to get the verification proof P_j. CSP returns P_j to TPA.
3) $Verify(\Omega_j, P_j) \rightarrow (true/false)$. After TPA receives P_j, he verifies the proof by checking (12). If the data are intact, it outputs true, otherwise it outputs false.

Algorithm 1 shows the detailed process of the proof generation.

Algorithm 1 $ProofGen(\cdot)$

Input: Challenge C_j, file F, file tags T
Output: Proof P_j

1. if $S_j' > t$ then
2. $C \leftarrow C_j$;
3. else
4. generate P_j ;
5. return P_j;
6. end if
7. compute $\triangle t$;
8. Sort(Ω);

9. if $m_C > m'$ then
10. $\Omega \rightarrow \bar{\Omega}$;
11. Sort($\bar{\Omega}$);
12. if $\bar{\Omega} \neq \emptyset$ then
13. for j=1 to m do
14. $DP_{\bar{\Omega}_j} = e(u, pk)^{\sum_{j\in\bar{\Omega}_j}(B_j \cdot \frac{\sum_{v_j\in V_j} v_j}{num_{V_j}})}$;
15. $TP_{\bar{\Omega}_j} = \prod_{j\in\bar{\Omega}_j} \sigma_j^{\frac{\sum_{v_j\in V_j} v_j}{num_{V_j}}}$;
16. endfor
17. endif
18. endif
19. for Ω_j in Ω do
20. $\Omega_j = \bar{\Omega}_k^{'} + \Omega_j^{'}$;
21. $DP_{\bar{\Omega}_j'} = \prod_{\bar{\Omega}_j\in\bar{\Omega}_j'} DP_{\bar{\Omega}_j}$ and $TP_{\bar{\Omega}_j'} = \prod_{\bar{\Omega}_i\in\bar{\Omega}_j'} TP_{\bar{\Omega}_j}$;
22. $DP_{\Omega_j} = DP_{\Omega_j'} \cdot DP_{\bar{\Omega}_j'}$, and $TP_{\Omega_j} = TP_{\Omega_j'} \cdot TP_{\bar{\Omega}_j'}$;
23. $P_j = \{DP_{\Omega_j}, TP_{\Omega_j}\}$;
24. endfor
25. return P_j;

In Algorithm 1, lines 1-6 is to generate challenge set C. The CSP calculates S_j' according to (3), and adds the challenge C_j to the challenge set C while $S_j' > t$; if $S_j' = t$, the proof P_j of the challenge C_j is directly calculated. In lines 7-9, for challenge set C, if the number of challenges satisfies $m_c > m'$ before t, the CSP extracts $\bar{\Omega}$ according to Section 4.2. After that, the CSP sorts the elements in $\bar{\Omega}$. In lines 10-19, if there exist frequent itemsets, the CSP will split and combine the challenges according to Section 4.3 and Section 4.4 to get P_j. Finally, the proof P_j is sent to the TPA.

5. Algorithm Analysis

Definition 5. Computational Diffie-Hellman (CDH) problem. If g, g^a, and g^b are known, it is computationally infeasible to calculate g^{ab} with unknown $a, b \in Z_p$.

5.1 Correctness of Algorithm

The data proof without challenge split $DP_{\Omega_j} = e(u, pk)^{\sum_{j\in\Omega_j} m_j v_j}$ and the tag proof $TP_{\Omega_j} = \prod_{j\in\Omega_j} \sigma_j^{v_j}$ can be checked by

$$DP_{\Omega_j} \cdot e\left(\prod_{j\in\Omega_j} H(B_{id})^{v_j}, pk\right) = e\left(TP_{\Omega_j}, g\right). \quad (13)$$

The data proof of a challenge has been split and combined by

$$DP_{\Omega_j} = DP_{\Omega_j'} \cdot DP_{\bar{\Omega}_j'} = e(u,pk)^{\sum_{j\in\Omega_j'} B_j v_j} \cdot \prod_{\bar{\Omega}_j\in\bar{\Omega}_j'} e(u,pk)^{\sum_{j\in\bar{\Omega}_j}(m_j\cdot\frac{\sum_{v_j\in V_j} v_j}{num_{V_j}})}$$
$$= e(u,pk)^{\sum_{j\in\Omega_j} B_j v_j + \sum_{\bar{\Omega}_j\in\bar{\Omega}_j'}\sum_{j\in\bar{\Omega}_j}(B_j\cdot\frac{(\sum_{u_j\in V_j} u_j) - v_j num_{V_j}}{num_{V_j}})}$$
$$= e(u,pk)^{\sum_{j\in\Omega_j} B_j v_j} \cdot e(u,pk)^{\sum_{\bar{\Omega}_j\in\bar{\Omega}_j'}\sum_{j\in\bar{\Omega}_j}(B_j\cdot\frac{(\sum_{u_j\in V_j} u_j) - v_j num_{V_j}}{num_{V_j}})}.$$

Since the operations are exponential operations, the data proof of the challenge data set Ω_j can be obtained by multiplying the duplicate data set and the rest data set. The data proof of a challenge that has been split can be converted from the data proof that has not been split. The tag proof after the challenge split is computed by

$$\mathrm{TP}_{\Omega_j} = \mathrm{TP}_{\Omega_j'} \cdot \mathrm{TP}_{\bar{\Omega}_j'} = \prod_{j\in\Omega_j'} \sigma_j^{v_j} \cdot \prod_{\bar{\Omega}_j\in\bar{\Omega}_j'}\prod_{j\in\bar{\Omega}_j} \sigma_j^{\frac{\sum_{v_j\in V_j} v_j}{num_{V_j}}}$$
$$= \prod_{j\in\Omega_j} \sigma_j^{v_j} \cdot \prod_{\bar{\Omega}_j\in\bar{\Omega}_j'}\prod_{j\in\bar{\Omega}_j} \sigma_j^{\frac{(\sum_{u_j\in V_j} u_j) - v_j num_{V_j}}{num_{V_j}}}.$$

We can judge the correctness of the proposed algorithm by verifying the correctness of the verification (12). The proof of the verification is given by

$$left = DP_{\Omega_j} \cdot e\left(\prod_{j\in\Omega_j'} H(B_{id})^{v_j} \cdot \prod_{j\in\bar{\Omega}_j'} H(B_{id})^{\frac{\sum_{v_j\in V_j} v_j}{num_{\bar{\Omega}_j'}}}, pk\right)$$
$$= \prod_{j\in\bar{\Omega}_j'} e(u,pk)^{\sum_{j\in\bar{\Omega}_j}\left(B_j\cdot\frac{\sum_{v_j\in V_j} v_j}{num_{\bar{\Omega}_j}}\right)} \cdot e(u,pk)^{\sum_{j\in\Omega_j'} B_j v_j}$$
$$\cdot \mathrm{e}\left(\prod_{j\in\Omega_j'} H(B_{id})^{v_j} \cdot \prod_{j\in\bar{\Omega}_j'} H(B_{id})^{\frac{\sum_{v_j\in V_j} v_j}{num_{\bar{\Omega}_j}}}, pk\right)$$
$$= e\left(u^{\sum_{j\in\bar{\Omega}_j'}\left(\sum_{j\in\bar{\Omega}_j}\left(B_j\cdot\frac{\sum_{v_j\in V_j} v_j}{num_{\bar{\Omega}_j}}\right)\right)} \cdot u^{\sum_{j\in\Omega_j'} B_j v_j} \cdot \prod_{i\in\Omega_i'} H(B_{id})^{v_i} \cdot \prod_{j\in\bar{\Omega}_j'} H(B_{id})^{\frac{\sum_{v_j\in V_j} v_j}{num_{\bar{\Omega}_j}}}, pk\right)$$
$$= e\left(u^{\sum_{j\in\bar{\Omega}_j'}\left(\sum_{j\in\bar{\Omega}_j}\left(B_j\cdot\frac{\sum_{v_j\in V_j} v_j}{num_{\bar{\Omega}_j}}\right)\right)} \cdot \prod_{j\in\bar{\Omega}_j'} H(B_{id})^{\frac{\sum_{v_j\in V_j} v_j}{num_{\bar{\Omega}_j}}} \cdot \prod_{j\in\Omega_j} H(B_{id})^{v_j} \cdot u^{\sum_{j\in\Omega_j} v_j B_j}, g^a\right)$$
$$= e(\mathrm{TP}_{\Omega_j'} \cdot \mathrm{TP}_{\bar{\Omega}_j'}, g) = \mathrm{e}(TP_{\Omega_j}, g) = right$$

According to the proof of the verification formula, homomorphism technology can be used to verify the integrity of data. In the verification, our algorithm uses the property of the bilinear pairing to generate an encrypted proof with the challenge stamp. Even though the TPA cannot decrypt it, he can verify the correctness of the proof without decrypting it. Based on Definition 5, it is computationally infeasible to forge DP_{Ω_j} and TP_{Ω_j}. Since the CSP cannot decrypt sk from pk in our algorithm, he also cannot forge or replace $\{DP_{\Omega_j}, TP_{\Omega_j}\}$ effectively and get the true results. In addition, when challenging the integrity of data, the TPA randomly generates the value v_j for each challenge. The CSP cannot predict it or store the corresponding data proofs and tag proofs after calculating these data. Due to the randomness of the value of the challenge, the repetition rate of challenges in different verification is very small so that the verification can effectively resist the replay attack on the CSP. Only if the received challenge and the corresponding data blocks are used while the CSP generates the proofs, the verification can be passed. We can draw conclusion from the previous analysis that the proposed algorithm is correct.

Based on the analysis above, the security of the data owner's outsourced data can be guaranteed, and the proposed algorithm can effectively resist the forge attack, replace attack, and replay attack.

5.2 Feasibility Analysis of Challenge Split

The purpose of dynamically scheduling the challenge data set in the verification is to ensure that all challenges can be successfully completed before the deadline. Thus, the whole process of challenge scheduling should be optimized in terms of the deadline of each challenge and CSP's dynamic resource allocation. Our algorithm ensures that every challenge can be successfully finished from three aspects.

First, the enforceability of the challenge is evaluated. When the new challenge C_j is released, the CSP calculates $S_j^{'}$ by applying Equation (3). If $S_j^{'} > 0$, the challenge C_j will be added to the challenge set C for further operation. Otherwise, it will be executed directly and calculated to get the proof P_j.

Second, the cost of the challenge split and computation overhead of combination of verification data and verification proof are evaluated. If the challenge C_j has c data blocks to be calculated, the calculation cost of proof for the traditional verification algorithms is equal to $C_p(c)$ by (1), and if there are m challenges now, the calculation cost of the proof is $m \cdot C_p(c)$. In this paper, when the duplicate rate of the challenge data set is high, the duplicate data among the challenges will not be calculated again. Suppose that the time of the frequent itemset extraction is $T_f(c)$. So the verification time cost of the algorithm is $m \cdot C_p(c-d) + T_f(c)$. This will continue to be discussed in Section 5.3.

Finally, whether the challenge can be completed on time after the split is evaluated. The completion time of the challenge C_j depends on its execution time and deadline. Since the deadline is fixed, we can apply (3) to dynamically schedule the start time of the challenge. Let the current time be t, we have sorted each challenge C_j in set C by S_j' from small to large. Also, when $t = S_j'$, the challenge C_j must start to generate the proof to ensure that challenge can be completed before the deadline.

5.3 Computational Complexity

The computational cost of the algorithm depends on the cost of proof generation, the cost of frequent itemset extraction, and the cost of the challenge split and combination. If there are m

challenges with c data blocks in each challenge now, the cost of proof generation for the traditional verification algorithms is $m \cdot C_p(c)$. However, for our scheme, the calculation time of (1) and (2) is negligible because they are simply algebraic operations. On the other hand, the verification cost of our algorithm mainly depends on the execution time of (3)-(10). Assume that each of the m challenges has d duplicate indexes of data blocks, and the extraction time of frequent itemsets is $T_f(c)$. The total computation cost is $m \cdot C_p(c-d) + T_f(c)$. The cost of frequent itemset extraction mainly depends on the operation time of FP-Growth. We will find in simulation that when the duplicate rate is less than 0.3%, the execution time of frequent itemset extraction is higher than the traditional schemes. As the larger d is, the smaller $C_p(c-d)$ is. Therefore, it is more suitable to use MT-DVA while the number of duplicate indexes in the challenge set is large.

6. Simulation

In order to further evaluate the performance of the proposed algorithm, we rent the computing and storage resource of Alibaba cloud as the cloud service provider, and utilize a laptop which is equipped with Intel Core i5-4210M 2.60GHz dual-core processors and 4GB RAM to work as the data owner. For the cloud storage data of 40GB, let the length of each data block be 320kB. The length of each element in group G is 1,024 bits, and the length of each element in Z_p is 160 bits. The algorithm is implemented based on the JPBC library and using Java.

For ease of description and fairness, our algorithm is abbreviated as MT-DVA and compared with [13] (called PDP) and [30] (called DHT-PA). Each experiment is repeated 20 times in the same environment and then averaged. The performance of these algorithms is compared in terms of the time of challenge set preprocessing, verification time, storage cost, and challenge loss rate.

6.1 Computation Cost of Challenge Set Split and Combination

For the proposed scheme, the CSP dynamically schedules the challenges according to their deadlines, and then extracts frequent itemsets, splits the challenge data, and combines the split verification data and proofs. All these operations cause external computational cost. The extraction of the frequent itemsets in the challenge data set is main time-consuming operation. So, we will evaluate the time of challenge set preprocessing, which includes dynamically scheduling challenges by their deadlines, extracting frequent itemsets, and splitting the challenge data. Let duplicate rate in the challenge be $\beta = \frac{duplicate\ index\ in\ a\ challenge}{total\ index\ in\ a\ challenge}$. We assume that $\beta = 20\%$ in each challenge. **Fig. 5** shows the execution time of the external steps of the proposed algorithm, when the number of TPAs is 10 and 20 respectively. As shown in **Fig. 5**, with the increasing number of data blocks in the challenge, the running time also increases. However, even if each challenge contains 50,000 indexes, the runtime which is 4326ms is much less than the execution time of a challenge.

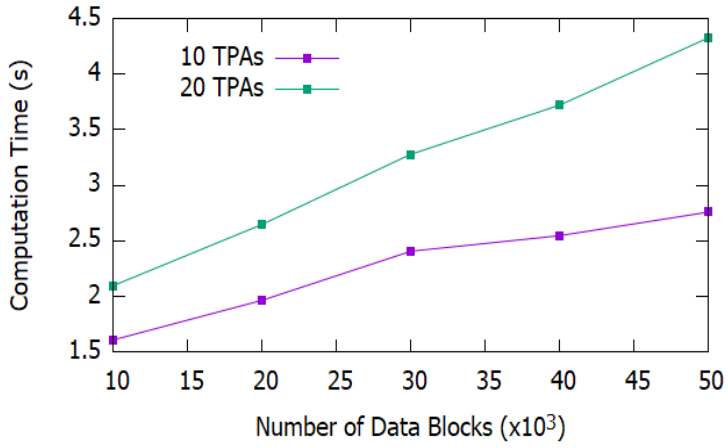


Fig. 5. The Time of Challenges Preprocessing

6.2 Verification Efficiency

Fig. 6 shows the average verification time for one of 20 TPAs. When the duplicate rate is 0.3%, the verification time of MT-DVA is higher than DHT-PA at the beginning. However, when the number of verification data blocks reaches 20,000, the average verification time of MT-DVA is 829.388s, and the average verification time of DHT-PA is 809.783s and PDP is 815.621s. With the increasing of the number of verification data blocks, the verification time of MT-DVA is much less than DHT-PA. In **Fig. 6**, the curves of the PDP and DHT-PA almost coincide. The reason is that DHT-PA only performs the verification challenge sent by each TPA sequentially. DHT-PA is sometimes several tens of seconds faster than PDP, but it is not clearly visible in the figure. Moreover, when the challenge repetition rate is higher, the average verification time of MT-DVA for per TPA is shorter. In this paper, since we will batch generate the proofs of the duplicate data in challenge data set, it reduces the time of CSP repeatedly computing the proofs.

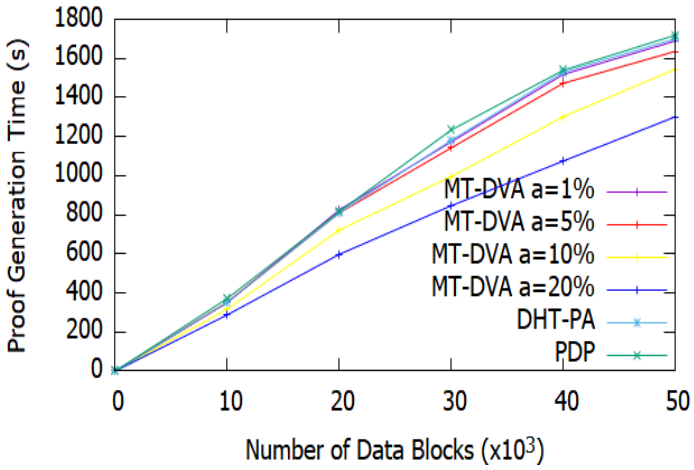


Fig. 6. The Time of Verification

6.3 Storage Overhead

During verification data splitting, if the same index exists in the two challenge data sets, CSP needs to store the random numbers, leading to storage overhead increasing. We set the number of TPA is 20, and the duplicate rate is 20%. As the number of data blocks in the challenge data set increases, shown in **Fig. 7**, both the number of duplicate data blocks and the storage overhead also increase. In **Fig. 7**, the curve of PDP coincides with DHT-PA, because PDP and DHT-PA only need to store *DP* and *TP*, the storage size of which is almost the same. Although the storage overhead of MT-DVA is larger than PDP and DHT-PA, the storage overhead of the challenges, each of which contains 50,000 data blocks, is only 4680 KB. So, it is acceptable.

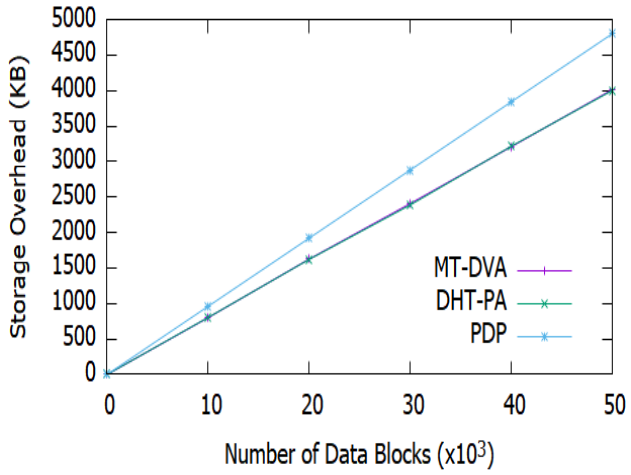


Fig. 7. Storage Cost of Challenge

6.4 Challenge Loss Rate

In the data verification process, the failure of the challenges should be avoided as much as possible. The main factor that causes the challenge failure is to miss the challenge deadline. The authors define the challenge loss rate as the ratio of the number of challenges missing deadline to the total number of challenges. The challenge loss rate is shown in **Fig. 8**. Obviously, DHT-PA and PDP both have a greater challenge loss rate. The main reason is that both DHT-PA and PDP do not schedule the challenges before the verification, so it is more likely to miss the deadline. For the proposed algorithm, we first execute the challenge of earlier deadline. At the same time, a challenge with a later deadline may have the same set of frequent items as the executed challenge before, so reducing the time of proof generation, and avoiding missing the deadline.

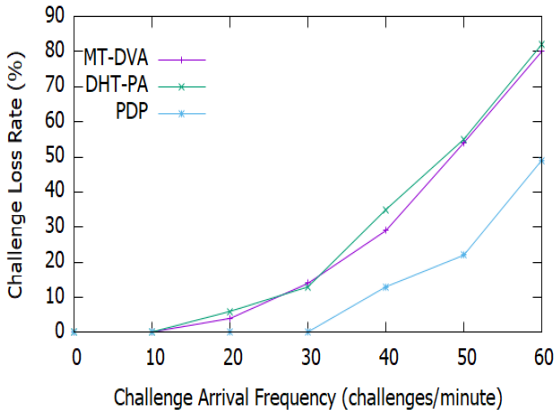


Fig. 8. Challenge Loss Rate

In summary, the proposed algorithm can batch compute the duplicate data in the challenges sent by different TPAs. Although the storage overhead is increased by a small amount, the computation load of CSP is greatly reduced while the duplicate rate is high or the number of data block is large.

7. Conclusion

Data integrity verification provides an important way to verify the integrity of outsourced data. In order to reduce the unnecessary overhead caused by duplicate data verification, this paper provides an optimized data verification algorithm for multiple verifiers and multiple verification challenges. The algorithm extracts frequent itemsets to split the challenges and sorts the challenges according to their deadline. This algorithm reduces the computation overhead of proof generation. In the future, we will further optimize the algorithm and reduce the cost by adjusting Δt.

References

[1] M. Sookhak, A. Gani, H. Talebian, A. Akhunzada, S. Ullah Khan, R. Buyya, and A. Y. Zomaya, "Remote data auditing in cloud computing environments: A survey, taxonomy, and open issues," *ACM Computing Surveys*, vol. 47, no. 4, pp. 1-34, July 2015. Article (CrossRef Link)

[2] H. Li, D. Liu, Y. Dai, and T. H. Luan, "Engineering searchable encryption of mobile cloud networks: When QoE meets QoP," *IEEE Wireless Communications*, vol. 22, no. 4, pp. 74-80, Aug. 2015. Article (CrossRef Link)

[3] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, "Efficient Public Verification of Data Integrity for Cloud Storage Systems from Indistinguishability Obfuscation," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 3, pp. 676-688, Mar. 2017. Article (CrossRef Link)

[4] M. Balduzzi, J. Zaddach, D. Balzarotti, E. Kirda, and S. Loureiro, "A security analysis of amazon's elastic compute cloud service," in *Proc. of the 27th International Conference on Dependable Systems and Networks Workshops*, pp. 1427-1434, 2012. Article (CrossRef Link)

[5] G. Xu, Z. Sun, C. Yan, and Y. Gan, "A rapid detection algorithm of corrupted data in cloud storage," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 115-125, Jan. 2018. Article (CrossRef Link)

[6] B. R. Kandukuri, V. R. Paturi, and A. Rakshit, "Cloud security issues," in *Proc. of IEEE International Conference on Services Computing IEEE Computer Society*, pp. 517-520, 2009. Article (CrossRef Link)
[7] S. Zawoad, R. Hasan, and M. K. Islam, "SECProv: Trustworthy and Efficient Provenance Management in the Cloud," in *Proc. of IEEE Conference on Computer Communications*, pp. 1241-1249, Apr. 2018. Article (CrossRef Link)
[8] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. of IEEE Conference on Computer Communications*, pp. 525-533, Mar. 2010. Article (CrossRef Link)
[9] Y. Zhang, C. Xu, S. Yu, H. Li, and X. Zhang, "SCLPV: Secure Certificateless Public Verification for Cloud-Based Cyber-Physical-Social Systems Against Malicious Auditors," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 159-170, Dec. 2015. Article (CrossRef Link)
[10] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-Based Remote Data Integrity Checking with Perfect Data Privacy Preserving for Cloud Storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 767-778, Apr. 2017. Article (CrossRef Link)
[11] Y. Deswarte, J. J. Quisquater, and A. Saïdane, "Remote Integrity Checking - How to Trust Files Stored on Untrusted Servers," in *Proc. of Working conference on Integrity and Internal Control in Information Systems*, pp. 1-11, 2003. Article (CrossRef Link)
[12] M. Cherubini, A. Meylan, B. Chapuis, M. Humbert, I. Bilogrevic, and K. Huguenin, "Towards Usable Checksums: Automating the Integrity Verification of Web Downloads for the Masses," in *Proc. of ACM Conference on Computer and Communications Security*, pp.1256-1271, Oct. 2018. Article (CrossRef Link)
[13] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. of the 14th ACM Conference on Computer and Communications Security*, pp. 598-609, 2007. Article (CrossRef Link)
[14] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. of ACM Conference on Computer and Communications Security*, pp. 213-222, Nov. 2009. Article (CrossRef Link)
[15] H. Shacham and B. Waters, "Compact Proofs of Retrievability," in *Proc. of the 14th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 90-107, Dec. 2008. Article (CrossRef Link)
[16] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297-319, 2004. Article (CrossRef Link)
[17] A. Juels, and B. S. Kaliski Jr., "Pors: proofs of retrievability for large files," in *Proc. of the 14th ACM Conference on Computer and Communications Security*, pp. 584-597, 2007. Article (CrossRef Link)
[18] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," in *Proc. of International Conference on Management of Data*, pp. 1-12, May 2000. Article (CrossRef Link)
[19] 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 System*, vol. 22, no. 5, pp. 847-859, May 2011. Article (CrossRef Link)
[20] G. Yang, J. Yu, and W. Shen, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *Journal of Systems and Software*, vol. 113, pp. 130-139, Mar. 2016. Article (CrossRef Link)
[21] Y. Zhu, H. Hu, G. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multi-cloud storage," *IEEE Transactions on Parallel and Distributed System*, vol. 23, no. 12, pp. 2231-2244, Dec. 2012. Article (CrossRef Link)
[22] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Transactions on Parallel and Distributed System*, vol. 24, no. 9, pp. 1717-1726, Sep. 2013. Article (CrossRef Link)

[23] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An Efficient Public Auditing Protocol With Novel Dynamic Structure for Cloud Data," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2402-2415, Oct. 2017. Article (CrossRef Link)
[24] H. Yan, J. Li, J. Han, and Y. Zhang, "A Novel Efficient Remote Data Possession Checking Protocol in Cloud Storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 78-88, Jan. 2017. Article (CrossRef Link)
[25] 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, Sep. 2011. Article (CrossRef Link)
[26] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the Association for Computing Machinery*, vol. 20, no. 1, pp. 46-61, Jan. 1973. Article (CrossRef Link)
[27] J. M. Rivas, J. J. Gutiérrez, J. C. Palencia, and M. G. Harbour, "Deadline assignment in EDF schedulers for real-time distributed systems," *IEEE Transactions on Parallel and Distributed System*, vol. 26, no. 10, pp. 2671-2684, Oct. 2015. Article (CrossRef Link)
[28] G. Xu, Y. Bai, Q. Pan, Q. Huang, and Y. Yang, "Data verification tasks scheduling based on dynamic resource allocation in mobile big data storage," *Computer Networks*, vol. 126, pp. 246-255, Oct. 2017. Article (CrossRef Link)
[29] S. Kato and N. Yamasaki, "Global EDF-based scheduling with laxity-driven priority promotion," *Journal of Systems Architecture*, vol. 57, no. 5, pp. 498-517, May 2011. Article (CrossRef Link)
[30] H. Tian, Y. Chen, C. C. Chang, H. Jiang, Y. Huang, Y. Chen, and J. Liu, "Dynamic-Hash-Table Based Public Auditing for Secure Cloud Storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 701-714, 2017. Article (CrossRef Link)

Guangwei Xu is a professor in the School of Computer Science and Technology at Donghua University, Shanghai, China. His research interests include remote data storage, data integrity verification, privacy protection in data sharing, searchable encryption, QoS and routing of the sensor network and internet of vehicles.

Miaolin Lai is a master candidate in the School of Computer Science and Technology at Donghua University, Shanghai, China. Her main research interests include the verification of data integrity and searchable encryption.

Xiangyang Feng is an associate professor in the School of Computer Science and Technology at Donghua University, Shanghai, China. His research interests include the data service, and data security.

Qiubo Huang is an associate professor in the School of Computer Science and Technology at Donghua University, Shanghai, China. His research interests include QoS and routing of the wireless and sensor networks, and data security.

Xin Luo is an associate professor in the School of Computer Science and Technology at Donghua University, Shanghai, China. His research interests include image processing, and image data security.

Li Li is an associate professor in the College of Architecture and Urban Planning at Tongji University, Shanghai, China. Her main research interests include the big data processing in architecture and urban planning.

Shan Li is a master candidate in the School of Computer Science and Technology at Donghua University, Shanghai, China. Her main research interests include the verification of data integrity and privacy protection in data sharing.