Verifying Outsourced Replicated Data in Cloud Computing Storage Systems

Ayad F. Barsoum[1] and M. Anwar Hasan [2]

[1]Computer Science Department, St.Mary's University, San Antonio, Texas, USA

[2]Department of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada.

abarsoum@stmarytx.edu, +1 (210) 436-3315

Abstract

Storage-as-a-Service offered by cloud service providers (CSPs) enables customers to store and retrieve almost unlimited amount of data by paying fees metered in GB/month. For an increased level of scalability, availability and durability, some customers may want their data to be replicated on multiple servers across multiple data centers. In this paper, we propose a pairing-based provable multi-copy data possession (PB-PMDP) scheme, which provides an evidence that all outsourced copies are actually stored and remain intact. Moreover, it allows authorized users (i.e., those who have the right to access the owner's file) to seamlessly access the file copies stored by the CSP, and supports public verifiability. The proposed scheme is proved to be secure against colluding servers. We illustrate the performance of the PB-PMDP scheme through theoretical analysis, which is then validated by experimental results on a commercial cloud platform. The verification time of the proposed scheme is practically independent of the number of file copies. Additionally, we discuss how to identify corrupted copies by slightly modifying the proposed PB-PMDP scheme.

Index Terms

Cloud computing, outsourcing data storage, data replication, cryptographic protocols, data integrity, verification techniques.

1 Introduction

Cloud computing is a distributed computational model over a large pool of shared-virtualized computing resources (e.g., storage, processing power, memory, applications, services, and network bandwidth). Cloud computing represents a vision of providing computing services as public utilities like water and electricity. It can potentially offer a number of key advantages: cost effectiveness, low management overhead, immediate access to a wide range of applications, flexibility to scale up/down information technology (IT) capacity, and mobility where customers can access information wherever they are, rather than having to remain at their desks.

Outsourcing data to a remote cloud service provider (CSP) allows organizations to store more data on the CSP than on private computer systems. Such outsourcing of data storage enables organizations to concentrate on innovations and get relief from the burden of constant server updates and other computing issues [1]. Moreover, many authorized users can access the remotely stored data from different geographic locations making it more convenient for them. A relatively recent survey indicates that IT outsourcing has grown by a staggering 79% as organizations seek to focus more on their core competencies and reduce costs [2].

However, the fact that data owners no longer physically possess their sensitive data raises new challenges to the tasks of data confidentiality and integrity in cloud computing systems. The confidentiality issue can be handled by encrypting sensitive data before outsourcing to remote servers. As for data integrity, the owners need to have a strong evidence that cloud servers still possess their data and it is not being tampered with or partially deleted over time, especially because the internal operation details of the CSP may not be known to customers.

Traditional cryptographic primitives for data integrity and availability based on hashing and signature schemes are not applicable to outsourced data without having a local copy. It is impractical for the owners to download all stored data to validate its integrity; this would require expensive I/O operations and immense communication overheads across the network. Therefore, efficient techniques are needed to verify the integrity of outsourced data with reduced communication, computation, and storage overheads. Consequently, many researchers have focused on the problem of provable data possession (PDP), and proposed different schemes to audit the data on remote storage sites.

PDP is a technique for validating data integrity over remote servers. In a typical PDP model, the data owner generates some metadata/information for a data file to be used later for verification purposes through a *challenge-response* protocol with the remote/cloud server. The owner sends the file to be stored on a remote server which may be untrusted, and deletes the local copy of the file. As a proof that the server is still possessing the data file in its original form, it needs to correctly compute a response to a challenge vector sent from a verifier – who can be the original data owner or a trusted entity that shares some information with the owner. Researchers have proposed different variations of PDP schemes under different cryptographic assumptions; for example, see [3]–[11]. PDP schemes presented in [3]–[11] focus on a *single copy* of the file and provide no proof that the CSP stores multiple copies of the owner's file.

Curtmola et al. [12] were the first to present a multiple-replica PDP (MR-PDP) scheme that creates multiple copies of an owner's file and audit them. The MR-PDP scheme increases data availability; a corrupted data copy can be reconstructed using duplicated copies on other servers. The interaction between authorized users (those who have the right to access the owner's file) and the CSP was not considered in [12]. The MR-PDP scheme supports only private verifiability, i.e., only the data owner can check data possession. Public verifiability is a key feature in remote data checking schemes to avoid disputes that may arise between the data owner and the CSP. Delegating the auditing process (without revealing secret keys) to a trusted third party for verifying the data integrity can resolve such disputes.

Main contributions. Our contributions can be summarized as follows:

- We propose a pairing-based provable multi-copy data possession (PB-PMDP) scheme. This scheme provides an adequate guarantee that the CSP stores all copies that are agreed upon in the service contract, and these copies are intact. The authorized users can seamlessly access the copies received from the CSP. The PB-PMDP scheme supports public verifiability.
- We justify the performance of the proposed PB-PMDP scheme through theoretical analysis, experimental results on Amazon cloud platform, and comparison with the MR-PDP model [12]. Moreover, we discuss a slight modification of the proposed PB-PMDP scheme to identify corrupted copies.
- We show the security of our scheme against colluding servers.

Paper organization. The remainder of the paper is organized as follow. Section 2 contains related work. Section 3 presents a basic provable multi-copy data possession, and a review of the MR-PDP scheme [12]. Our system and assumptions are presented in Section 4. Section 5 highlights our security model. The proposed scheme is elaborated in Section 6. The performance analysis is shown in Section 7. Section 8 presents the implementation and experimental results on Amazon cloud platform. How to identify the corrupted copies is discussed in Section 9. Concluding remarks are given in Section 10.

2 Related Concepts

2.1 Single-Copy PDP Schemes

The key purpose of PDP schemes is to validate the integrity of data outsourced to remote servers. In PDP schemes, a verifier is allowed to efficiently, periodically, and securely validate that a remote server – which supposedly stores the owner's potentially very large amount of data – is actually storing the data in its original form. A simple solution to tackle the problem of data integrity over remote servers is by fragmenting the data file F into blocks $\{b_1, b_2, \ldots, b_m\}$, computing a message authentication code (MAC) σ_j for each block b_j: $\sigma_j = \text{MAC}_{sk}(j||b_j)_{1\leq j\leq m}$, sending both the data file F and the MACs $\{\sigma_j\}_{1\leq j\leq m}$ to the remote/cloud server, deleting the local copy of the file, and storing only the secret key sk. During the verification process, a verifier requests for a set of randomly selected blocks and their corresponding MACs, re-computes the MAC of each retrieved block using sk, and compares the re-computed MACs with the received values from the remote server [4]. This approach suffers from a severe drawback; the communication complexity is linear with the queried data size, which is impractical especially when the available bandwidth is limited.

For efficient validation of outsourced data integrity, a number of PDP protocols have been proposed in the recent past [3]–[11], which focus on a *single copy* of the file and provide no proof that the CSP stores multiple copies of the owner's file (more details on various PDP schemes can be found in our technical report [13]). Curtmola et al. [12] were the first to address the integrity of multiple data copies outsourced to remote servers. The MR-PDP scheme of [12] is based on the single-copy PDP model of [3].

Ateniese et al. [3] introduced an RSA-based PDP model in which the data owner fragments the file F into blocks $\{b_1, b_2, \ldots, b_m\}$, and generates a tag for each block to be used later for verification purposes. The file and the tags are sent to be stored on remote servers. The scheme presented in [3] provides *probabilistic* guarantee of data possession, where the verifier checks for a random subset of file blocks with each challenge (spot checking).

In their work, Ateniese et al. [3] differentiate between the concept of public verifiability and private verifiability. In public verifiability, anyone who knows the owner's public key can challenge the remote server and verify that the server is still possessing the owner's files. On the other hand, private verifiability allows only the original owner to perform the auditing task. Two main schemes are presented in [3]: sampling PDP (S-PDP) and efficient PDP (E-PDP) schemes. The E-PDP model provides a weaker guarantee of data possession; it guarantees possession of the *sum* of file blocks being challenged and not necessarily possession of each one of these blocks.

2.2 Proof of Retrievability and Data Redundancy

Proof of retrievability (POR) is a complementary approach to PDP, and is stronger than PDP in the sense that the verifier can reconstruct the entire data file from the responses that are reliably transmitted from the server. This is due to encoding of the data file, for example using *erasure codes*, before outsourcing to remote servers. Various POR schemes can be found in the literature, for example [14]–[19].

Data redundancy can be achieved using replication or coding schemes, where the former is the simplest way that can be adopted by many storage systems. For a data file with size $|F|$ bits, the storage cost for n copies over cloud servers is $n|F|$ bits. In erasure codes, the file is divided into m blocks and encoded into ℓ blocks, where $\ell > m$ [20]. The encoded blocks are stored at ℓ different servers (one code block per server to prevent simultaneous failure of all blocks), and thus the storage cost is $\frac{|F|}{m}\ell$ bits. The original file can be reconstructed from any m out of the ℓ servers.

In the context of this work, we are considering economically-motivated CSPs that may attempt to use less storage than required by the service contract through deletion of a few copies of the file. The CSPs have almost no financial benefit by deleting only a small portion of a copy of the file. Redundancy using erasure codes has less storage cost; however, duplicating data file across multiple servers achieves scalability in the sense that if the number of users grows, then with more copies of data the user access time can be kept below a certain threshold. Such scalability is a fundamental customer requirement in cloud computing systems. A file that is duplicated and stored strategically on multiple servers – located at various geographic locations – can help reduce access time and communication cost for users. On the other hand, in responding to a data access request for coding-based systems, the CSP has to access at least m servers to reconstruct the original data file, and thus increased time overhead (network latency and computation time to decode data blocks) occurs at the CSP side.

More importantly, in case of data corruption, erasure codes require the precise identification of failed/corrupted blocks. Without the ability to identify which blocks have been corrupted, there is potentially a factorial combination of blocks to try to reconstruct the original data file; that is $\binom{n}{\ell}$. For replication-based systems, a server's copy can be reconstructed even from a complete damage using duplicated copies on other servers. As a result of the aforementioned reasons, in our work we do not apply erasure codes to the data file before outsourcing.

3 Provable Multi-Copy Data Possession Schemes

In this section, we consider the case of provable possession for multiple data copies, for which we start with a basic provable multi-copy data possession scheme followed by a review of the MR-PDP scheme of [12].

3.1 Basic Provable Multi-Copy Data Possession Scheme

Suppose that a CSP offers to store n copies of an owner's file on different servers for pre-specified fees according to the used storage space. Thus, the data owner needs a strong evidence to ensure that the CSP is actually storing no less than n copies, all these copies are complete and correct, and the owner is not paying for a service that he does not get. A straightforward solution to this problem is to use a single-copy PDP scheme to separately challenge and verify the integrity of each copy on each server. This is not a workable solution, since the CSP can convince the data owner that n copies of the file are stored, while there is only one copy.

Whenever a request for a PDP scheme execution is made to one of the n severs, it is forwarded to the server which actually possesses the stored copy. The core of this cheating is that the n copies are identical making it trivial for the CSP to deceive the owner. Therefore, a step towards the solution is to leave the control of the file copying operation in the owner's hand to create unique *differentiable* copies.

In the basic provable multi-copy data possession scheme, the data owner creates n distinct copies by encrypting the file under n different keys. Hence, the CSP cannot use one copy to answer the challenges for another. This natural solution enables the verifier to separately challenge each copy on the remote servers, and ensure that the CSP is possessing not less than n copies.

Although the above basic scheme is a workable solution, it is impractical and has the following drawbacks:

- *Data access and key management* are serious problems with the basic scheme. Since the file is encrypted under n different keys, the owner has to keep these keys secret from the CSP, and share the n keys with each authorized user for each data file. Moreover, when an authorized user interacts with the CSP to retrieve the data file, it is not necessarily to receive the same copy each time. According to the load balancing mechanism used by the CSP to organize the work of the servers, the authorized user's request is directed to the server with the lowest congestion. Consequently, each copy should contain some indicator about its encryption key to enable the authorized user to properly decrypt and access the received copy.
- The computation and communication complexities of the verification task are linear with the number of copies.

3.2 Multiple-Replica Provable Data Possession Scheme

The MR-PDP scheme of [12] is based on the PDP model of [3]. In [12] distinct copies of a data files are created by first *encrypting* the file using one key, then *masking* the encrypted version (n times) with different randomness generated from a pseudo-random function.

Initially, a file F is fragmented into blocks $\{b_j\}_{1\leq j\leq m}$. The owner encrypts F using a key K to obtain an encrypted version $\widetilde{F} = \{\tilde{b}_j\}_{1\leq j\leq m}$, where $\tilde{b}_j = E_K(b_j)$. The owner generates n distinct copies $\{\widehat{F}_i\}_{1\leq i\leq n}$, where $\widehat{F}_i = \{\hat{b}_{ij}\}_{1\leq j\leq m}$, $\hat{b}_{ij} = \tilde{b}_j + r_{ij}$ (added as large integers in $\mathbb{Z}$), and $r_{ij} = f_x(i||j)$. f_x is a pseudo-random function keyed with a secret key x. Fig. 1 gives a summary of the MR-PDP scheme.

In the MR-PDP scheme, if an authorized user interacts with the CSP to access an owner's file, the CSP retrieves one of the available copies. Upon receiving this copy, the authorized user

Setup

- $N = \acute{p}\acute{q}$ is the RSA modulus ($\acute{p}$ & $\acute{q}$ are prime numbers)
- $\acute{g}$ is a generator of QR_N (QR_N is the set of quadratic residues modulo N)
- Public key $pk = (N, \acute{g}, e)$, secret key $sk = (d, v, x)$, $v, x \in_R \mathbb{Z}_N$, and $ed \equiv 1 \bmod (\acute{p}-1)(\acute{q}-1)$
- π_k is a pseudo-random permutation keyed with a key k, f_x is a pseudo-random function keyed with the secret key x, and H is a hash function ($H : \{0,1\}^* \rightarrow QR_N$)
- File $F = \{b_j\}_{1 \le j \le m}$, and E_K is an encryption algorithm under a key K

Data Owner

- Encrypts the data file F under the key K to obtain an encrypted version $\widetilde{F} = \{\tilde{b}_j\}_{1 \le j \le m}$, where $\tilde{b}_j = E_K(b_j)$
- Uses the encrypted version $\widetilde{F}$ to create a set of tags $\{T_j\}_{1 \le j \le m}$ for all copies: $T_j = (H(v||j) \cdot \acute{g}^{\tilde{b}_j})^d \bmod N$
- Generates n distinct copies $\{\widehat{F}_i\}_{1 \le i \le n}$, $\widehat{F}_i = \{\hat{b}_{ij}\}_{1 \le j \le m}$ utilizing random masking:

```
for $i$ = 1 to $n$ do
    for $j$ = 1 to $m$ do
        1. Computes a random value $r_{ij} = f_x(i||j)$
        2. Computes the replica's block $\hat{b}_{ij} = \tilde{b}_j + r_{ij}$ (added as large integers in $\mathbb{Z}$)
```

- Sends the copy $\widehat{F}_i$ to a server S_i, $i :\ 1 \rightarrow\ n$

Checking possession of a replica $\widehat{F}_z$

Owner — Remote Server S_z

1. Picks a key k for the function π, c (# of blocks to be challenged), and $\acute{g}_s = \acute{g}^s \bmod N$ ($s \in_R \mathbb{Z}_N$)

$$\xrightarrow{c,\ k,\ \acute{g}_s}$$

2. Computes a set A of random indices: $A = \{j\} = \pi_k(l)_{1 \le l \le c}$
3. Computes $T = \prod_{j \in A} T_j \bmod N$
4. Computes $\rho = \acute{g}_s^{\sum_{j \in A} \hat{b}_{zj}} \bmod N$

$$\xleftarrow{T,\ \rho}$$

5. Computes $A = \{j\} = \pi_k(l)_{1 \le l \le c}$
6. Checks $\left(\frac{T^e}{\prod_{j \in A} H(v||j)} \cdot \acute{g}^{r_{chal}}\right)^s \overset{?}{=} \rho$, where $r_{chal} = \sum_{j \in A} r_{zj}$

Fig. 1: The MR-PDP scheme by Curtmola et al. [12].

has to know the copy index to properly *unmask* it before decryption. Due to the opaqueness of the internal operations of the CSP, the authorized users cannot recognize which copy has been received. If i (the copy index) is attached with each copy forming the structure $(i||\widehat{F}_i)$, corrupting or swapping copy indices hinder the correct unmasking process. Thus, the authorized users are unable to access the data file.

For verification purposes, portion of the set $\{r_{ij}\}$ is needed to be generated ($r_{chal} = \sum_{j \in A} r_{zj}$ in Fig. 1). These random values cannot be publicly known, otherwise the CSP can derive the encrypted version $\widetilde{F}$, and store only one copy. Hence, only private verifiability is supported.

4 Our System and Assumptions

System components. The cloud computing storage model considered in this work consists of three main components as illustrated in Fig. 2: (i) a data owner that can be an individual or an organization originally possessing sensitive data to be stored in the cloud; (ii) a CSP who manages cloud servers and provides paid storage space on its infrastructure to store the owner's files; and (iii) authorized users – a set of owner's clients who have the right to access the remote data.

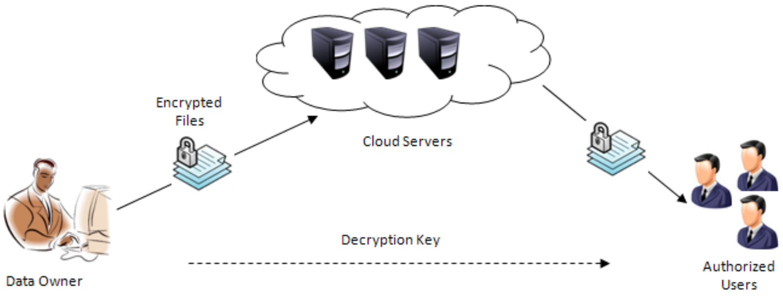


Fig. 2: Cloud computing data storage system model.

The storage model used in this work can be adopted by many practical applications. For example, e-Health applications can be envisioned by this model where the patients' database that contains large and sensitive information can be stored on cloud servers. In these types of applications, the e-Health organization can be considered as the data owner, and the physicians as the authorized users who have the right to access the patients' medical history. Many other practical applications like financial, scientific, and educational applications can be viewed in similar settings.

In this work, we focus on *sensitive* archived and *warehoused* data, which is essential in many applications such as digital libraries and astronomical/medical/scientific/legal repositories.

Such data are subject to infrequent change, so we treat them as static.

Outsourcing and accessing. The data owner has a file F consisting of m blocks and the CSP offers to store n copies $\{\widetilde{F}_1, \widetilde{F}_2, \ldots, \widetilde{F}_n\}$ of the owner's file on different servers – to prevent simultaneous failure of all copies – in exchange for pre-specified fees metered in GB/month. The number of copies depends on the nature of data; more copies are needed for critical data that cannot easily be reproduced, and to achieve a higher level of scalability. This critical data should be replicated on multiple servers across multiple data centers. On the other hand, non-critical, reproducible data are stored at reduced levels of redundancy. The CSP pricing model is related to the number of data copies.

For data confidentiality, the owner encrypts his data before outsourcing to the CSP. An authorized user of the outsourced data sends a data-access request to the CSP and receives a file copy in an encrypted form that can be decrypted using a secret key shared with the owner. According to the load balancing mechanism used by the CSP to organize the work of the servers, the data-access request is directed to the server with the lowest congestion, and thus the authorized user is not aware of which copy has been received.

We assume that the interaction between the owner and the authorized users to authenticate their identities and share the secret key has already been completed, and it is not considered in this work. Throughout this paper, the terms cloud server and cloud service provider are used interchangeably.

Threat model. The completeness and correctness of customers' data in the cloud may be at risk due to the following reasons. First, the CSP – whose goal is likely to make a profit and maintain a reputation – has an incentive to hide data loss (due to hardware failure, management errors, various attacks) or reclaim storage by discarding data that has not been or is rarely accessed. Second, a dishonest CSP may store fewer copies than what has been agreed upon in the service contact with the data owner, and try to convince the owner that all copies are correctly stored intact. Third, the cloud infrastructures are subject to a wide range of internal and external security threats. Incidences of security breaches of cloud services surface from time to time [21], [22]. In short, although outsourcing data to the cloud is attractive from the view point of cost and complexity of long-term large-scale data storage, it does not offer sufficient guarantee on data integrity. This problem, if not properly handled, may hinder the successful deployment and wide acceptance of the cloud paradigm. The goal of the proposed scheme is to detect (with *high probability*) the CSP misbehavior by validating the number and integrity of file copies.

Underlying algorithms. The proposed scheme consists of five polynomial time algorithms:

KeyGen, CopyGen, TagGen, Prove, and Verify.

– $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$. This algorithm is run by the data owner. It takes as input a security parameter 1^κ, and returns a public key pk (publicly known) and a private key sk (kept secret by the owner).

– $\widetilde{\mathbb{F}} \leftarrow \mathsf{CopyGen}(CN_i, F)_{1 \leq i \leq n}$. This algorithm is run by the data owner. It takes as input a copy number CN_i and a file F, and generates n copies $\widetilde{\mathbb{F}} = \{\widetilde{F}_i\}_{1 \leq i \leq n}$. The owner sends the copies $\widetilde{\mathbb{F}}$ to the CSP to be stored on cloud servers.

– $\Phi \leftarrow \mathsf{TagGen}(sk, \widetilde{\mathbb{F}})$. This algorithm is run by the data owner. It takes as input the private key sk and the file copies $\widetilde{\mathbb{F}}$, and outputs tags/authenticators set Φ, which is an ordered collection of tags for the data blocks. The owner sends Φ to the CSP to be stored along with the copies $\widetilde{\mathbb{F}}$.

– $\mathbb{P} \leftarrow \mathsf{Prove}(\widetilde{\mathbb{F}}, \Phi, chal)$. This algorithm is run by the CSP. It takes as input the file copies $\widetilde{\mathbb{F}}$, the tags set Φ, and a challenge $chal$ (sent from a verifier). It returns a proof $\mathbb{P}$, which guarantees that the CSP is actually storing n copies and all these copies are intact.

– $\{1, 0\} \leftarrow \mathsf{Verify}(pk, \mathbb{P})$. This algorithm is run by a verifier (original owner or any other trusted auditor). It takes as input the public key pk, and the proof $\mathbb{P}$ returned from the CSP. The output is 1 if the integrity of all file copies is correctly verified or 0 otherwise.

5 Security Model

Generally speaking, a remote data checking scheme is considered to be secure if (i) a polynomial-time algorithm that can cheat the verifier and pass the auditing procedure with non-negligible probability does not exist; and (ii) there exists a polynomial-time extractor that can repeatedly execute the challenge response protocol until it extracts the original data file.

Following [14], we would like the remote data checking scheme to be *correct* and *sound*. These two requirements are defined as follows:

- Correctness requires that the verifier accepts valid server responses.
- Soundness requires that any cheating server that passes the verification process is actually storing the owner's data intact.

The security of the proposed scheme can be stated using a "game" that captures the data possession property [3], [14]. The data possession game between an adversary $\mathcal{A}$ (acts as a malicious CSP) and a challenger $\mathcal{C}$ (acts as a verifier) consists of the following:

- Setup. $\mathcal{C}$ runs the KeyGen algorithm to generate a key pair (pk, sk), and sends pk to $\mathcal{A}$.
- Interact. $\mathcal{A}$ interacts with $\mathcal{C}$ to get the file copies and the verification tags set Φ. $\mathcal{A}$ adaptively selects a file F and sends it to $\mathcal{C}$. $\mathcal{C}$ runs the two algorithms CopyGen and

TagGen to create n distinct copies $\widetilde{\mathbb{F}}$ along with the tags set Φ, and returns both $\widetilde{\mathbb{F}}$ and Φ to $\mathcal{A}$.

Moreover, $\mathcal{A}$ can request challenges $\{chal_i\}_{1\leq i\leq L}$ for some parameter $L \geq 1$ of his choice, and return proofs $\{\mathbb{P}_i\}_{1\leq i\leq L}$ to $\mathcal{C}$. $\mathcal{C}$ runs the Verify algorithm and provides the verification results to $\mathcal{A}$. The INTERACT step between $\mathcal{A}$ and $\mathcal{C}$ can be repeated polynomially-many times.

- CHALLENGE. $\mathcal{A}$ decides on a file F previously used during the INTERACT step, requests a challenge $chal$ from $\mathcal{C}$, and generates a proof $\mathbb{P} \leftarrow \mathsf{Prove}(\widetilde{\mathbb{F}}', \Phi, chal)$, where $\widetilde{\mathbb{F}}'$ is $\widetilde{\mathbb{F}}$ except that at least one of its file copies (or a portion of it) is missing or tampered with. Upon receiving the proof $\mathbb{P}$, $\mathcal{C}$ runs the Verify algorithm and if $\mathsf{Verify}(pk, \mathbb{P})$ returns 1, then $\mathcal{A}$ has won the game. The CHALLENGE step can be repeated polynomially-many times for the purpose of data extraction.

The proposed scheme is secure if the probability that any polynomial-time adversary $\mathcal{A}$ wins the game is negligible. In other words, if a polynomial-time adversary $\mathcal{A}$ can win the game with non-negligible probability, then there exists a polynomial-time extractor that can repeatedly execute the CHALLENGE step until it extracts the blocks of data copies.

File swapping attack. In this type of attacks, the remote server tries to prove the possession of the data using blocks from different files. A remote data checking scheme must be secure against such an attack.

6 PROPOSED PB-PMDP SCHEME

6.1 Overview and Rationale

Generating unique differentiable copies of the data file is the core to design a provable multi-copy data possession scheme. Identical data copies enable the CSP to simply deceive the owner by storing only one copy and pretending that it stores multiple copies. Using a simple yet *efficient* way, the proposed scheme generates distinct copies utilizing the *diffusion* property of any secure encryption scheme. The diffusion property ensures that the output bits of the ciphertext depend on the input bits of the plaintext in a very complex way, i.e., there will be an unpredictable complete change in the ciphertext, if there is a single bit change in the plaintext [23]. The interaction between the authorized users and the CSP is considered through this methodology of generating distinct copies, where the former can decrypt and access a file copy received from the CSP. In the proposed scheme, the authorized users need only to keep a single secret key – shared with the data owner – to decrypt the file copy, and it is not necessarily to recognize the index of the received copy.

6.2 Notations

- F is a data file to be outsourced, and is composed of a sequence of m blocks, i.e., $F = \{b_1, b_2, \ldots, b_m\}$.
- $\pi_{key}(\cdot)$ is a pseudo-random permutation (PRP): $key \times \{0,1\}^{\log_2(m)} \rightarrow \{0,1\}^{\log_2(m)}$.
- $\psi_{key}(\cdot)$ is a pseudo-random function (PRF): $key \times \{0,1\}^* \rightarrow \mathbb{Z}_p$ (p is a prime number).
- **Bilinear Map/Pairing**. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of prime order p. Let $\bar{g}$ and g be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. A bilinear pairing is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with the following properties [24]:
  1) *Bilinear*: $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}\ \forall\ u \in \mathbb{G}_1, v \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p(\mathrm{p}isaprimenumber)$
  2) *Non-degenerate*: $\hat{e}(\bar{g}, g) \neq 1$
  3) *Computable*: there exists an efficient algorithm for computing $\hat{e}$.
- $\mathcal{H}(\cdot)$ is a map-to-point hash function : $\{0,1\}^* \rightarrow \mathbb{G}_1$.
- E_K is an encryption algorithm with strong *diffusion* property and a key K, e.g., AES.

Remark 1. Homomorphic linear authenticators (HLAs) [14], [18], [25] are basic building blocks in the proposed scheme. Informally, the HLA is a tag computed by the owner for each data block b_j that enables a verifier to validate the data possession on remote servers by sending a challenge vector $chal$ of c elements: $chal = \{r_1, r_2, \ldots, r_c\}$. As a response, the servers can homomorphically construct a tag authenticating the value $\sum_{j=1}^{c} r_j \cdot b_j$. The response is validated by a verifier, and accepted only if the servers honestly compute the response using the owner's file blocks. The proposed scheme utilizes the BLS (Boneh-Lynn-Shacham) HLAs [14].

6.3 PB-PMDP Procedural Steps

■ ***Key Generation***. As earlier, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map and g is a generator of $\mathbb{G}_2$. The data owner runs the KeyGen algorithm to generate a private key $x \in \mathbb{Z}_p$ and a public key $y = g^x \in \mathbb{G}_2$ along with s elements $(u_1, u_2, \ldots, u_s) \in_R \mathbb{G}_1$.

■ ***Generation of Distinct Copies***. The data owner runs the CopyGen algorithm to create n differentiable copies $\widetilde{\mathbb{F}} = \{\widetilde{F}_i\}_{1 \leq i \leq n}$. The copy $\widetilde{F}_i$ is generated by concatenating a copy number i with the file F, then encrypting using E_K, i.e., $\widetilde{F}_i = E_K(i||F)$. $\widetilde{F}_i$ is divided into blocks $\{\tilde{b}_{ij}\}_{1 \leq j \leq m}$, and the block $\tilde{b}_{ij}$ is further fragmented into s sectors $\{\tilde{b}_{ij1}, \tilde{b}_{ij2}, \ldots, \tilde{b}_{ijs}\}$, i.e., the copy $\widetilde{F}_i = \{\tilde{b}_{ijk}\}_{\substack{1 \leq j \leq m \\ 1 \leq k \leq s}}$, where each sector $\tilde{b}_{ijk} \in \mathbb{Z}_p$ for some large prime p.

The authorized users need to keep only a single secret key K. Later, when an authorized user receives a file copy from the CSP, he decrypts the copy and removes the index from the copy header to reconstruct the plain form of the received file copy.

- ***Generation of Tags***. Given the distinct file copies $\widetilde{\mathbb{F}} = \{\widetilde{F}_i\}$, where $\widetilde{F}_i = \{\tilde{b}_{ijk}\}$, the data owner runs the TagGen algorithm to generate a tag σ_{ij} for each block $\tilde{b}_{ij}$ as $\sigma_{ij} = (\mathcal{H}(ID_F||j).\prod_{k=1}^{s} u_k^{\tilde{b}_{ijk}})^x \in \mathbb{G}_1$ $(i : 1 \to n,\ j : 1 \to m,\ k : 1 \to s)$. In the tag computation, $ID_F = Filename||n||m||u_1||\ldots||u_s$ is a unique fingerprint for each file F comprising the file name, the number of copies for this file, the number of blocks per copy, and the random values $\{u_k\}_{1\leq k\leq s}$. Embedding the ID_F into the block tag σ_{ij} prevents the CSP from cheating by using blocks from different files (*file swapping attack*).

  In order to reduce storage overhead on cloud servers and lower communication cost, the data owner generates an aggregated tag σ_j for the blocks at the same indices in each copy $\widetilde{F}_i$ as $\sigma_j = \prod_{i=1}^{n} \sigma_{ij} \in \mathbb{G}_1$. Hence, instead of storing mn tags, the proposed PB-PMDP scheme requires the CSP to store only m tags for the files copies $\widetilde{\mathbb{F}}$. Let us denote the set of aggregated tags as $\Phi = \{\sigma_j\}_{1\leq j\leq m}$. The data owner sends $\{\widetilde{\mathbb{F}}, \Phi, ID_F\}$ to the CSP, and deletes the copies and the tags from its local storage.

- ***Challenge***. For challenging the CSP and validating the integrity of all copies, the verifier sends c (# of blocks to be challenged) and two fresh keys at each challenge: a PRP(π) key k_1 and a PRF(ψ) key k_2. *Both* the verifier and the CSP use π keyed with k_1 and the ψ keyed with k_2 to generate a set $Q = \{(j, r_j)\}$ of c pairs of random indices and random values, where $\{j\} = \pi_{k_1}(l)_{1\leq l\leq c}$ and $\{r_j\} = \psi_{k_2}(l)_{1\leq l\leq c}$.

- ***Response***. The CSP runs the Prove algorithm to generate a set $Q = \{(j, r_j)\}$ of random indices and values, and provide an evidence that the CSP is still correctly possessing the n copies. The CSP responds with a proof $\mathbb{P} = \{\sigma, \mu\}$, where

$$\sigma = \prod_{(j,r_j)\in Q} \sigma_j^{r_j} \in \mathbb{G}_1,\ \mu = \{\mu_{ik}\}_{\substack{1\leq i\leq n, \\ 1\leq k\leq s}} \text{ and } \mu_{ik} = \sum_{(j,r_j)\in Q} r_j \cdot \tilde{b}_{ijk} \in \mathbb{Z}_p.$$

- ***Verify Response***. Upon receiving the proof $\mathbb{P} = \{\sigma, \mu\}$ from the CSP, the verifier runs the Verify algorithm to check the following verification equation:

$$\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}([\prod_{(j,r_j)\in Q} \mathcal{H}(ID_F||j)^{r_j}]^n \cdot \prod_{k=1}^{s} u_k^{\sum_{i=1}^{n}\mu_{ik}}, y). \tag{1}$$

  In equation (1), the term $\sum_{i=1}^{n}\mu_{ik}$ is linear in n, while the term $[\cdot]^n$ costs one more exponentiation for any value of n. If the verification equation passes, the Verify algorithm returns 1, otherwise 0. The correctness of verification equation (1) can be shown as follows:

$$\hat{e}(\sigma, g) \quad = \quad \hat{e}(\prod_{(j,r_j)\in Q} \sigma_j^{r_j}, g)$$

$$
\begin{aligned}
&= \hat{e}(\prod_{(j,r_j)\in Q}[\prod_{i=1}^{n}\sigma_{ij}]^{r_j}, g) \\
&= \hat{e}(\prod_{(j,r_j)\in Q}[\prod_{i=1}^{n}(\mathcal{H}(ID_F||j)\cdot\prod_{k=1}^{s}u_k^{\tilde{b}_{ijk}})^{x}]^{r_j}, g) \\
&= \hat{e}(\prod_{(j,r_j)\in Q}[\prod_{i=1}^{n}\mathcal{H}(ID_F||j)\cdot\prod_{i=1}^{n}\prod_{k=1}^{s}u_k^{\tilde{b}_{ijk}}]^{r_j}, y) \\
&= \hat{e}(\prod_{(j,r_j)\in Q}\prod_{i=1}^{n}\mathcal{H}(ID_F||j)^{r_j}\cdot\prod_{(j,r_j)\in Q}\prod_{i=1}^{n}\prod_{k=1}^{s}u_k^{r_j\cdot\tilde{b}_{ijk}}, y) \\
&= \hat{e}([\prod_{(j,r_j)\in Q}\mathcal{H}(ID_F||j)^{r_j}]^{n}\cdot\prod_{k=1}^{s}u_k^{\sum_{i=1}^{n}\sum_{(j,r_j)\in Q}r_j\cdot\tilde{b}_{ijk}}, y) \\
&= \hat{e}([\prod_{(j,r_j)\in Q}\mathcal{H}(ID_F||j)^{r_j}]^{n}\cdot\prod_{k=1}^{s}u_k^{\sum_{i=1}^{n}\mu_{ik}}, y).
\end{aligned}
$$

Remark 2. The proposed PB-PMDP scheme supports public verifiability where anyone, who knows the owner's public key but is not necessarily the data owner, can send a challenge vector to the CSP and verify the response. Public verifiability can resolve disputes that may arise between the data owner and the CSP regarding data integrity. If such a dispute occurs, a trusted third party auditor (TPA) can determine whether the data integrity is maintained or not. Since the owner's public key is only needed to perform the verification step, the owner is not required to reveal his secret key to the TPA. The PB-PMDP scheme is presented in Fig. 3, and the security analysis is given in Appendix A.

6.4 Reducing the Communication Cost

One can attempt to change the PB-PMDP scheme to reduce the communication cost by a factor of n by permitting the CSP to compute and send $\mu = \{\hat{\mu}_k\}_{1\leq k\leq s}$, where $\hat{\mu}_k = \sum_{i=1}^{n}\mu_{ik}$. However, this modification enables the CSP to simply cheat the verifier as follows:

$$
\hat{\mu}_k = \sum_{i=1}^{n}\mu_{ik} = \sum_{i=1}^{n}\sum_{(j,r_j)\in Q} r_j\cdot\tilde{b}_{ijk} = \sum_{(j,r_j)\in Q} r_j\cdot\sum_{i=1}^{n}\tilde{b}_{ijk}.
$$

Thus, the CSP can just keep the sectors summation $\sum_{i=1}^{n}\tilde{b}_{ijk}$ not the sectors themselves. Moreover, the CSP can corrupt the block sectors and the summation is still valid. Therefore, the proposed scheme requires the CSP to send $\mu = \{\mu_{ik}\}_{\substack{1\leq i\leq n\\ 1\leq k\leq s}}$, and the summation $\sum_{i=1}^{n}\mu_{ik}$ is done on the verifier side.

A slightly modified version of the PB-PMDP scheme can reduce the communication cost by a factor of s during the response phase by allowing the CSP to compute and send $\mu = \{\mu_i\}_{1\leq i\leq n}$

Setup

- File $F = \{b_1, b_2, \ldots, b_m\}$.
- $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map, g is a generator for $\mathbb{G}_2$, $x \in \mathbb{Z}_p$ is a private key, and $y = g^x \in \mathbb{G}_2$ along with $(u_1, u_2, \ldots, u_s) \in_R \mathbb{G}_1$ form a public key.

Data Owner

- Creates distinct file copies $\widetilde{\mathbb{F}} = \{\widetilde{F}_i\}_{1 \leq i \leq n}$, where $\widetilde{F}_i = E_K(i||F)_{1 \leq i \leq n}$. Each copy $\widetilde{F}_i$ is an ordered collection of blocks fragmented into sectors, i.e., $\widetilde{F}_i = \{\tilde{b}_{ijk}\}_{\substack{1 \leq j \leq m \\ 1 \leq k \leq s}}$, where $\tilde{b}_{ijk} \in \mathbb{Z}_p$.
- Calculates the block tag $\sigma_{ij} = (\mathcal{H}(ID_F||j).\prod_{k=1}^{s} u_k^{\tilde{b}_{ijk}})^x \in \mathbb{G}_1$.
- Computes a set of aggregated tags $\Phi = \{\sigma_j\}_{1 \leq j \leq m}$ for the blocks at the same indices in each copy $\widetilde{F}_i$, where $\sigma_j = \prod_{i=1}^{n} \sigma_{ij} \in \mathbb{G}_1$.
- Sends $\{\widetilde{\mathbb{F}}, \Phi, ID_F\}$ to the CSP and deletes the copies and the tags from its local storage.

Challenge Response

Verifier   CSP

1) Picks c (# of blocks to be challenged) and two fresh keys k_1 and k_2

2) Generates a set $Q = \{(j, r_j)\}$, $\{j\} = \pi_{k_1}(l)_{1 \leq l \leq c}$ and $\{r_j\} = \psi_{k_2}(l)_{1 \leq l \leq c}$

$$\xrightarrow{c,\ k_1,\ k_2}$$

3) Generates a set Q as the verifier did

4) Computes $\sigma = \prod_{(j,r_j) \in Q} \sigma_j^{r_j} \in \mathbb{G}_1$

5) Computes $\mu = \{\mu_{ik}\}_{\substack{1 \leq i \leq n, \\ 1 \leq k \leq s}}$

$$\mu_{ik} = \sum_{(j,r_j) \in Q} r_j \cdot \tilde{b}_{ijk} \in \mathbb{Z}_p$$

$$\xleftarrow{\sigma,\ \mu=\{\mu_{ik}\}_{\substack{1 \leq i \leq n \\ 1 \leq k \leq s}}}$$

6) Checks $\hat{e}(\sigma, g) \overset{?}{=} \hat{e}([\prod_{(j,r_j) \in Q} \mathcal{H}(ID_F||j)^{r_j}]^n \cdot \prod_{k=1}^{s} u_k^{\sum_{i=1}^{n} \mu_{ik}}, y)$

Fig. 3: The proposed PB-PMDP scheme.

instead of $\mu = \{\mu_{ik}\}_{\substack{1 \leq i \leq n. \\ 1 \leq k \leq s}}$ In this version, a file copy $\widetilde{F}_i$ is divided into blocks, but the blocks are not fragmented into sectors, i.e., a copy $\widetilde{F}_i = \{\tilde{b}_{ij}\}_{1 \leq j \leq m}$, where $\tilde{b}_{ij} \in \mathbb{Z}_p$. A tag σ_{ij} is generated for each block $\tilde{b}_{ij}$: $\sigma_{ij} = (\mathcal{H}(ID_F||j).u^{\tilde{b}_{ij}})^x \in \mathbb{G}_1$, where u is a generator for $\mathbb{G}_1$.

Tags are aggregated into a set $\Phi = \{\sigma_j\}_{1\leq j\leq m}$, where $\sigma_j = \prod_{i=1}^{n} \sigma_{ij}$. In this scenario, the CSP responds with $\sigma = \prod_{(j,r_j)\in Q} \sigma_j^{r_j} \in \mathbb{G}_1$ and $\mu = \{\mu_i\}_{1\leq i\leq n}$, where $\mu_i = \sum_{(j,r_j)\in Q} r_j \cdot \tilde{b}_{ij} \in \mathbb{Z}_p$. The verification equation (1) will be modified to $\hat{e}(\sigma, g) \overset{?}{=} \hat{e}([\prod_{(j,r_j)\in Q} \mathcal{H}(ID_F||j)^{r_j}]^n \cdot u^{\sum_{i=1}^{n}\mu_i}, y)$.

This reduced communication cost will be at the expense of increased storage overhead on the CSP side, where each block $\tilde{b}_{ij} \in \mathbb{Z}_p$ will be accompanied with a tag $\sigma_{ij} \in \mathbb{G}_1$ of equal length. If the block size is greater than $|p|$ (the bit length of the prime p), the CSP can simply cheat by storing $\tilde{b}_{ij}$ mod p instead of the whole block $\tilde{b}_{ij}$. Therefore, with this slightly modified version, to store n copies each of size $|F|$ bits, the total storage over the CSP will be $(n+1)|F|$ bits (using tag aggregation approach). The storage overhead equals the size of a complete file copy. The more storage space is used over the CSP side, the more fees the customers are charged (pay-as-you-go pricing model).

7 Performance Analysis

In this section, we evaluate the performance of the presented schemes: MR-PDP and PB-PMDP. The file F used in our performance analysis is of size 64MB divided in blocks of 4KB. Without loss of generality, we assume that the desired security level is 80-bit. Thus, we utilize an elliptic curve defined over Galois field $GF(p)$ with $|p|$ = 160 bits (a point on this curve can be represented by 161 bits using compressed representation [26]), and the size of the RSA modulus N is 1024 bits.

Similar to [7], [27], the computation cost for the MR-PDP and PB-PMDP is estimated in terms of used cryptographic operations, which are notated in Table 1. $\mathbb{G}$ indicates a group of points over a suitable elliptic curve in the bilinear pairing, and QR_N is a set of quadratic residues modulo N.

TABLE 1: Notation of cryptographic operations.

Notation	Description	Notation	Description
$\mathcal{H}_{\mathbb{G}}$	Hashing to $\mathbb{G}$	$\mathcal{H}_{QR_N}$	Hashing to QR_N
$\mathcal{E}_{\mathbb{G}}$	Exponentiation in $\mathbb{G}$	$\mathcal{E}_{\mathbb{Z}_N}$	Exponentiation in $\mathbb{Z}_N$
$\mathcal{M}_{\mathbb{G}}$	Multiplication in $\mathbb{G}$	$\mathcal{M}_{\mathbb{Z}}$	Multiplication in $\mathbb{Z}$
$\mathcal{M}_{\mathbb{Z}_p}$	Multiplication in $\mathbb{Z}_p$	$\mathcal{D}_{\mathbb{Z}}$	Division in $\mathbb{Z}$
$\mathcal{A}_{\mathbb{Z}_p}$	Addition in $\mathbb{Z}_p$	$\mathcal{A}_{\mathbb{Z}}$	Addition in $\mathbb{Z}$
$\mathcal{P}$	Bilinear pairing	E_K	Encryption using K
$\mathcal{R}$	Random-number generation		

To perform a fair comparison between the PB-PMDP and the MR-PDP [12], we assume two small modifications to the model presented in [12]. First, we assume that the indices of the blocks being challenged are the same across all copies (*this assumption is an optimization for the verification computations of the MR-PDP*). Second, for the CSP to prove the possession of the blocks (not just only their sum), each block being challenged is multiplied by a random value. The second modification makes the S-PDP version of [3] to be the base of the MR-PDP scheme.

Let n, m, and s denote the number of copies, the number of blocks per copy, and the number of sectors per block, respectively. Let c denote the number of blocks to be challenged, and $|F|$ denote the size of the file copy. Let the keys used with π and ψ be of size 128 bits. Table 2 presents a theoretical analysis for the setup, storage, communication, and computation costs of the two schemes.

TABLE 2: Storage, communication, and computation costs for MR-PDP and PB-PMDP schemes.

		MR-PDP [12]	PB-PMDP
System Setup	Copies Generation	$E_K + nm\,\mathcal{R} + nm\,\mathcal{A}_{\mathbb{Z}}$	$n\,E_K$
	Tags Generation	$2m\,\mathcal{E}_{\mathbb{Z}_N} + m\,\mathcal{M}_{\mathbb{Z}} + m\,\mathcal{H}_{QR_N}$	$(s+1)nm\,\mathcal{E}_{\mathbb{G}} + (ns+n-1)m\,\mathcal{M}_{\mathbb{G}} + nm\,\mathcal{H}_{\mathbb{G}}$
Storage	File Copies	$n\|F\|$	$n\|F\|$
	CSP Overhead	$1024m$ bits	$161m$ bits
Communication Cost	Challenge	$1280 + \log_2(c)$ bits	$256 + \log_2(c)$ bits
	Response	$1024(n+1)$ bits †	$161 + 160ns$ bits
Computation Cost	Proof	$(c+n)\,\mathcal{E}_{\mathbb{Z}_N} + (cn+c-1)\,\mathcal{M}_{\mathbb{Z}} + (c-1)n\,\mathcal{A}_{\mathbb{Z}}$	$c\,\mathcal{E}_{\mathbb{G}} + (c-1)\,\mathcal{M}_{\mathbb{G}} + csn\,\mathcal{M}_{\mathbb{Z}_p} + (c-1)sn\,\mathcal{A}_{\mathbb{Z}_p}$
	Verification	$(2n+c+1)\,\mathcal{E}_{\mathbb{Z}_N} + (cn+c+n-1)\,\mathcal{M}_{\mathbb{Z}}$ $+(c-1)n\,\mathcal{A}_{\mathbb{Z}} + c\,\mathcal{H}_{QR_N} + \mathcal{D}_{\mathbb{Z}}$	$2\mathcal{P} + (c+s+1)\,\mathcal{E}_{\mathbb{G}} + (c+s-1)\,\mathcal{M}_{\mathbb{G}}$ $+(n-1)s\,\mathcal{A}_{\mathbb{Z}_p} + c\,\mathcal{H}_{\mathbb{G}}$

† There is an optimization for this response to be $1024 + 160n$ bits using hashing.

7.1 Comments

Sytem Setup. As it can be seen in Table 2, the cost of generating data copies in the proposed PB-PMDP scheme is much less than that of the MR-PDP scheme. On the other hand, Curtmola *et al.* [12] efficiently reduce the computation cost of generating the block tags. This is due to the fact that the tags are generated from the encrypted version of the file before masking with some unique randomness to generate the differentiable copies. In general, the impact of setup computations on the overall system performance may be insignificant; setup is done only once during the life time of the data storage system, which may be for tens of years.

Storage overhead. Storage overhead is the additional space used to store necessary information other than the outsourced file copies $\widetilde{\mathbb{F}}$. Both schemes require $n|F|$ bits to store $\widetilde{\mathbb{F}}$, while the storage overhead for the PB-PMDP scheme is much less than that of the MR-PDP model. The overheads on the CSP are 2MB and 0.31MB for the MR-PDP and PB-PMDP schemes, respectively (about 84% reduction). Reducing the storage overhead on the CSP side is economically a key feature to lower the fees paid by the customers.

Communication cost. The communication cost of the MR-PDP scheme is less than that of PB-PMDP. For 20 copies of F, the communication costs for the MR-PDP and PB-PMDP schemes are about 2.8KB and 80KB, respectively. However, for small s ($\ll n$), the PB-PMDP will have less communication cost.

Computation cost. As observed from Table 2, the cost expression of the proof for the MR-PDP scheme has three terms linear in the number of copies n, while the PB-PMDP scheme has two terms linear in n. Moreover, there are three terms linear in n in the verification cost expression for the MR-PDP scheme, while the PB-PMDP scheme contains only *one* term linear in n in the corresponding expression. These terms affect the total computation time when dealing with a large number of copies in practical applications. We note that since the cost of an addition is negligibly smaller than those of pairing and exponentiation, the verification time in the proposed PB-PMDP scheme is practically not affected by the value of n.

8 Implementation and Experimental Evaluation

8.1 Implementation

We have implemented the MR-PDP and PB-PMDP schemes on Amazon Elastic Compute Cloud (Amazon EC2) [28] and Amazon Simple Storage Service (Amazon S3) [29] cloud platforms. Amazon EC2 is a web service that enables customers to lunch and manage Linux/Unix and Windows server instances (virtual servers) in Amazon's data centers. Customers can automatically scale up and down the number of EC2 instances according to their demands. Moreover, customers can upgrade and downgrade a specific EC2 instance to fit current requirements. Amazon S3 is storage for the Internet. It provides a simple web services interface that can be used to store and retrieve almost unlimited amount of data. Customers are allowed to choose the geographic locations where Amazon S3 will store the data.

Our implementation of the presented schemes consists of three modules: `OModule` (owner module), `CModule` (CSP module), and `VModule` (verifier module). `OModule`, which runs on the owner side, is a library that includes KeyGen, CopyGen, and TagGen algorithms. `CModule` is a library that runs on Amazon EC2 and includes Prove algorithm. `VModule` is a library to be run at the verifier side and includes the Verify algorithm.

In the experiments, we do not consider the system pre-processing time to prepare the different file copies and generate the tags set. Moreover, the time to access the file blocks is not considered in the implementation, as the state-of-the-art hard drive technology allows as much as 1MB to be read in just few nanoseconds [7]. Hence, the total access time is unlikely to have substantial impact on the overall system performance.

Implementation settings. In our implementation we use a ”large” Amazon EC2 instance to run `CModule`. This instance type provides total memory of size 7.5GB and 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each). One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0 - 1.2GHz 2007 Opteron or 2007 Xeon processor [30]. The `OModule` and `VModule` are executed on a desktop computer with Intel(R) Xeon(R) 2GHz processor and 3GB RAM running Windows XP. We outsource copies of a data file of size 64MB to Amazon S3. Algorithms (encryption, pairing, hashing, etc.) are implemented using MIRACL library version 5.4.2. In the experiments, we utilize the MNT curve [31] defined over prime field $GF(p)$ with $|p|$ = 160 bits and embedding degree = 6 (the MNT curve with these parameters is provided by the MIRACL library).

8.2 Experimental Evaluation

Timing measurements. The proposed PB-PMDP scheme is based on pairing and elliptic curve cryptography, while the MR-PDP scheme is based on RSA. To estimate the timing measurements for the cryptographic operations used in the implementations, we run the MIRACL library on the the used desktop computer. Table 3 presents the measured times (in milliseconds), where each reported measurement is an average of thousands of runs.

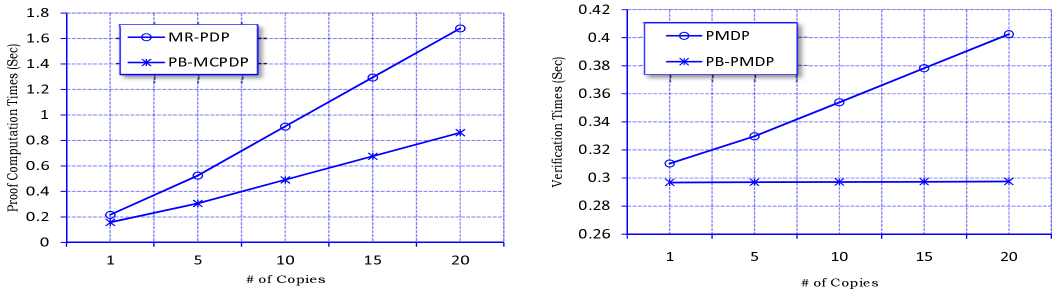
TABLE 3: Timing measurements for the cryptographic operations

Operation	Time (ms)	Operation	Time (ms)
$\mathcal{H}_{\mathbb{G}}$	0.22	$\mathcal{H}_{QR_N}$	0.34
$\mathcal{E}_{\mathbb{G}}$	0.27	$\mathcal{E}_{\mathbb{Z}_N}$	68.92/2.15/0.32
$\mathcal{M}_{\mathbb{G}}$	0.01	$\mathcal{M}_{\mathbb{Z}}$	0.02/0.004/0.0009
$\mathcal{M}_{\mathbb{Z}_p}$	0.00025	$\mathcal{D}_{\mathbb{Z}}$	0.00022
$\mathcal{A}_{\mathbb{Z}_p}$	0.00017	$\mathcal{A}_{\mathbb{Z}}$	0.009/0.00029
$\mathcal{P}$	4.6		

Table 3 shows three measurements for $\mathcal{E}_{\mathbb{Z}_N}$: 68.92 ms, 2.15 ms, and 0.32 ms. The reason is that the exponent part differs during the implementation of the MR-PDP scheme. For example, the data owner performs $\acute{g}^{\texttt{EXP}}$ mod N, where `EXP` is the exponent part of size 4KB (32768 bits). The owner does this operation during the tag generations and the verification phase. Utilizing the Fermat-Euler theorem [32], the owner can reduce the exponent part, where $\acute{g}^{\mathrm{EXP}} \equiv \acute{g}^{\mathrm{EXP} \bmod \phi(N)}$ mod N and $\phi(N) = (\acute{p}-1)(\acute{q}-1)$ is the Euler's totient function. On the other hand, the CSP cannot use this trick because $\phi(N)$ is not known in public. Therefore, $\mathcal{E}_{\mathbb{Z}_N}$ needs 68.92 ms and 2.15 ms at the CSP and the owner, respectively. Besides, a random value of size 160 bits is used in our slight modification to the MR-PDP scheme to prove the possession of the data blocks not only their sum. Thus, $\mathcal{E}_{\mathbb{Z}_N}$ needs 0.32 ms when the exponent part is 160 bits. Similar scenarios arise for $\mathcal{M}_{\mathbb{Z}}$ and $\mathcal{A}_{\mathbb{Z}}$ operations. $\mathcal{M}_{\mathbb{Z}}$ needs 0.02 ms for 32768-bits $\times$ 160-bits, 0.004 ms for 1024-bits $\times$ 1024-bits, and 0.0009 ms for 1024-bits $\times$ 160-bits. $\mathcal{A}_{\mathbb{Z}}$ needs 0.009 ms for 32768-bits + 32768-bits, and 0.00029 ms for 1024-bits + 1024-bits.

Experimental results. We compare the presented MR-PDP and PB-PMDP schemes in terms of both the proof computation times and the verification times. It has been reported in [3] that if the remote server is missing a fraction of the data, then the number of blocks that needs to be checked in order to detect server misbehavior with high probability is constant independent of the total number of file blocks. For example, if the server deletes 1% of the data file, the verifier needs only to check for c = 460-randomly chosen blocks of the file so as to detect this misbehavior with probability larger than 99%. Therefore, in our experiments, we use c = 460 to achieve a high probability of assurance.

For different number of copies, Figure 4a presents the proof computation times (in seconds). The timing curve of the proposed PB-PMDP scheme is less than that of the MR-PDP scheme. For 20 copies, the proof computation times for the MR-PDP and the PB-PMDP schemes are 1.68 and 0.86 seconds, respectively (about 49% reduction).



(a) CSP computation times (sec)

(b) Verifier computation times (sec)

Fig. 4: Computation costs of the MR-PDP and PB-PMDP.

Fig. 4b presents the verification times (in seconds). For 20 copies, the verification times for the MR-PDP and the PB-PMDP schemes are 0.40 and 0.29 seconds, respectively (about 27%

reduction).

More importantly, the verification timing curve of the PB-PMDP scheme is *almost* unchanged for the range of number of copies considered in our experiments. This is due to the fact that although the term $(n-1)s\,\mathcal{A}_{\mathbb{Z}_p}$ in the verification cost of the PB-PMDP scheme is linear in n (Table 2), in our experiments its numerical value is quite small compared to those of the other terms in the cost expression. This feature makes the the PB-PMDP scheme computationally cost-effective and more efficient when verifying a large number of file copies.

9 IDENTIFYING CORRUPTED COPIES

Here we show how the proposed PB-PMDP scheme can be slightly modified to identify the indices of corrupted copies. The proof $\mathbb{P} = \{\sigma, \mu\}$ generated by the CSP will be valid and will pass the verification equation (1) only if all copies are intact and consistent. Thus, when there is one or more corrupted copies, the whole auditing procedure fails. To handle this situation and identify the corrupted copies, a slightly modified version of the PB-PMDP scheme can be used. In this version, the data owner generates a tag σ_{ij} for each block $\tilde{b}_{ij}$, but does not aggregate the tags for the blocks at the same indices in different copies, i.e., $\Phi = \{\sigma_{ij}\}_{\substack{1 \le i \le n \\ 1 \le j \le m}}$. During the response phase, the CSP computes $\mu = \{\mu_{ik}\}_{\substack{1 \le i \le n \\ 1 \le k \le s}}$ as before, but $\sigma = \prod_{(j,r_j)\in Q} [\prod_{i=1}^{n} \sigma_{ij}]^{r_j} \in \mathbb{G}_1$. Upon receiving the proof $\mathbb{P} = \{\sigma, \mu\}$, the verifier first validates $\mathbb{P}$ using equation (1). If the verification fails, the verifier asks the CSP to send $\sigma = \{\sigma_i\}_{1 \le i \le n}$, where $\sigma_i = \prod_{(j,r_j)\in Q} \sigma_{ij}^{r_j}$. Thus, the verifier has two lists $\sigma\mathsf{List} = \{\sigma_i\}_{1 \le i \le n}$ and $\mu\mathsf{List} = \{\mu_{ik}\}_{\substack{1 \le i \le n \\ 1 \le k \le s}}$ ($\mu\mathsf{List}$ is a two dimensional list).

Utilizing a recursive divide-and-conquer approach (binary search) [33], the verifier can identify the indices of corrupted copies. Specifically, σList and μList are divided into halves: σList→ (σLeft:σRight), and μList→ (μLeft:μRight). The verification equation (1) is applied recursively on σLeft with μLeft and σRight with μRight. Note that the individual tags in σLeft or σRight are aggregated via multiplication to generate one σ that is used during the recursive application of equation (1). The procedural steps of identifying the indices of corrupted copies are indicated in Algorithm 1.

The binary search algorithm takes four parameters: σList, μList, start that indicates the start index of the currently working lists, and end to indicate the last index of the working lists. The initial call to the search algorithm takes (σList, μList, 1, n). The invalid indices are stored in invalidList, which is a global data structure.

This slight modification to identify the corrupted copies will be associated with some extra storage overhead on the cloud servers, where the CSP has to store mn tags for the file copies $\widetilde{\mathbb{F}}$

Algorithm 1: BS(σList, μList, start, end)

```
begin
    $len \longleftarrow$ (end$-$start)$+1$          /* The list length */
    if $len$ = $1$ then
        $\sigma \longleftarrow \sigma$List[start]
        $\{\mu_k\}_{1 \leq k \leq s} \longleftarrow \mu$List[start][$k$]
        $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(\prod_{(j,r_j) \in Q} \mathcal{H}(ID_F||j)^{r_j} \cdot \prod_{k=1}^{s} u_k^{\mu_k}, y)$
        if NOT verified then
            invalidList.Add(start)
        end
    else
        $\sigma \longleftarrow \prod_{i=1}^{len} \sigma$List[start$+i-1$]
        $\{\mu_{ik}\}_{\substack{1 \leq i \leq len \\ 1 \leq k \leq s}} \longleftarrow \mu$List[start$+i-1$][$k$]
        $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}([\prod_{(j,r_j) \in Q} \mathcal{H}(ID_F||j)^{r_j}]^{len} \cdot \prod_{k=1}^{s} u_k^{\sum_{i=1}^{len} \mu_{ik}}, y)$
        if NOT verified then
            /* work with the left and right halves of $\sigma$List and $\mu$List */
            mid$\longleftarrow$ $\lfloor$(start$+$end)$/2\rfloor$          /* List middle */
            BS($\sigma$List, $\mu$List, start, mid)       /* Left part */
            BS($\sigma$List, $\mu$List, mid+1, end)   /* Right part */
        end
    end
end
```

(m tags in the original version). Moreover, the challenge-response phase may be done in two rounds if the initial round to verify all copies fails.

We have performed experiments to show the effect of identifying the corrupted copies on the verification time. We generate 100 copies, which are verified in 0.3 seconds when all copies are accurate. A percentage – ranging from 1% to 20% – of the file copies is *randomly* corrupt. Fig. 5 shows the verification time (in seconds) with different corrupted percentages. The verification time is about 3.87 seconds when 1% of the copies are invalid. As observed from Fig. 5, when the percentages of corrupted copies are up to 15% of the total copies, the performance of using the binary search algorithm in the verification is more efficient than individual verification for each copy. It takes about 0.29 seconds to verify one copy, and thus individual verifications of

100 copies requires 100×0.29 = 29 seconds.

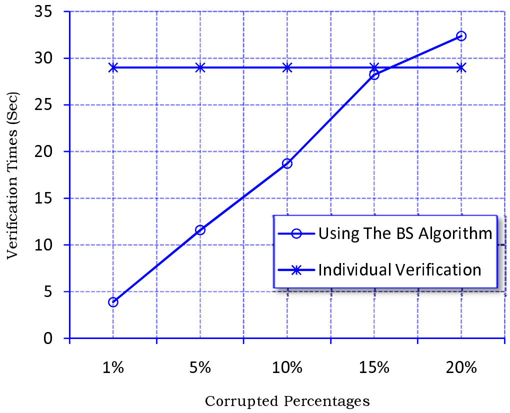


Fig. 5: Verification times with different percentages of corrupted copies.

In short, the proposed PB-PMDP scheme can be slightly modified to support the feature of identifying the corrupted copies at the cost of some extra storage, communication, and computation overheads. For the CSP to remain in business and maintain a good reputation, invalid responses to verifier's challenges are sent in very rare situations, and thus the original version of the proposed scheme is used in most of the time.

10 Summary and Concluding Remarks

In this work we have studied the problem of creating multiple copies of a data file and verifying those copies stored on cloud servers. We have proposed a pairing-based provable multi-copy data possession (PB-PMDP) scheme, which supports outsourcing of multiple data copies to untrusted CSP. The interaction between the authorized users and the CSP is considered in our scheme, where the authorized users can seamlessly access a data copy received from the CSP using a single secret key shared with the data owner. Moreover, the BP-PMDP scheme supports public verifiability, allows unlimited number of auditing, and provides *possession-free* verification where the verifier has the ability to verify the data integrity even though he neither possesses nor retrieves the file blocks from the server.

Through theoretical analysis, experimental results, and comparison with the previously proposed MR-PDP scheme, we have showed the improved performance of the proposed scheme. The verification time of PB-PMDP is practically independent of the number of file copies, which makes the scheme computationally cost-effective and more efficient when verifying a large number of file copies.

A slight modification can be done on the proposed scheme to support the feature of identifying the indices of corrupted copies of data. The corrupted copy can be reconstructed even from a complete damage using duplicated copies on other servers. Our analysis has shown that the proposed PB-PMDP scheme is provably secure.

ACKNOWLEDGMENTS

This work was supported in part through an NSERC grant awarded to Prof. Hasan.

REFERENCES

[1] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *CCSW '09: Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, New York,NY, USA, 2009, pp. 55–66.

[2] M. Xie, H. Wang, J. Yin, and X. Meng, "Integrity auditing of outsourced data," in *VLDB '07: Proceedings of the 33rd International Conference on Very Large Databases*, 2007, pp. 782–793.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2007, pp. 598–609.

[4] K. Zeng, "Publicly verifiable remote data integrity," in *Proceedings of the 10th International Conference on Information and Communications Security*, ser. ICICS '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 419–434.

[5] Y. Deswarte, J.-J. Quisquater, and A. Saïdane, "Remote integrity checking," in *6th Working Conference on Integrity and Internal Control in Information Systems (IICIS)*, S. J. L. Strous, Ed., 2003, pp. 1–11.

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

[7] F. Sebé, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Trans. on Knowl. and Data Eng.*, vol. 20, no. 8, 2008.

[8] P. Golle, S. Jarecki, and I. Mironov, "Cryptographic primitives enforcing communication and storage complexity," in *FC'02: Proceedings of the 6th International Conference on Financial Cryptography*, Berlin, Heidelberg, 2003, pp. 120–135.

[9] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in *HOTOS'07: Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, Berkeley, CA, USA, 2007, pp. 1–6.

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

[11] E. Mykletun, M. Narasimha, and G. Tsudik, "Authentication and integrity in outsourced databases," *Trans. Storage*, vol. 2, no. 2, 2006.

[12] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: multiple-replica provable data possession," in *28th IEEE ICDCS*, 2008, pp. 411–420.

[13] A. F. Barsoum and M. A. Hasan, "Provable possession and replication of data over cloud servers," Centre For Applied Cryptographic Research (CACR), University of Waterloo, Report 2010/32, 2010, http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-32.pdf.

[14] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ser. ASIACRYPT '08. Springer-Verlag, 2008, pp. 90–107.

[15] A. Juels and B. S. Kaliski, "PORs: Proofs of Retrievability for large files," in *CCS'07: Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 584–597.

[16] R. Curtmola, O. Khan, and R. Burns, "Robust remote data checking," in *StorageSS '08: Proceedings of the 4th ACM international workshop on Storage security and survivability*. New York, NY, USA: ACM, 2008, pp. 63–68.

[17] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: theory and implementation," in *CCSW '09: Proceedings of the 2009 ACM workshop on Cloud computing security*. New York, NY, USA: ACM, 2009, pp. 43–54.

[18] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *TCC '09: Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 109–127.

[19] K. D. Bowers, A. Juels, and A. Oprea, "Hail: a high-availability and integrity layer for cloud storage," in *CCS '09: Proceedings of the 16th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2009, pp. 187–198.

[20] M. K. Aguilera, R. Janakiraman, and L. Xu, "Using erasure codes efficiently for storage in a distributed system," in *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, ser. DSN '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 336–345.

[21] N. Gohring, "Amazon's S3 down for several hours," Online at http://www.pcworld.com/businesscenter/article/142549/amazons_s3_down_for_severalhours.html, 2008.

[22] B. Krebs, "Payment processor breach may be largest ever," Online at http://voices.washingtonpost.com/securityfix/2009/01/payment_processor_breach_may_b.html, Jan. 2009.

[23] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst.Tech. J.*, vol. 28, no. 4, 1949.

[24] A. Menezes, "An introduction to pairing-based cryptography," Lecture Notes 2005, Online at http://www.math.uwaterloo.ca/~ajmeneze/publications/pairings.pdf.

[25] G. Ateniese, S. Kamara, and J. Katz, "Proofs of storage from homomorphic identification protocols," in *ASIACRYPT '09: Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security*, Berlin, Heidelberg, 2009, pp. 319–333.

[26] P. S. L. M. Barreto and M. Naehrig, "IEEE P1363.3 submission: Pairing-friendly elliptic curves of prime order with embedding degree 12," New Jersey: IEEE Standards Association, 2006.

[27] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *SecureComm '08: Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, New York, NY, USA, 2008, pp. 1–10.

[28] Amazon elastic compute cloud (Amazon EC2), http://aws.amazon.com/ec2/.

[29] Amazon simple storage service (Amazon S3), http://aws.amazon.com/s3/.

[30] Amazon EC2 Instance Types, http://aws.amazon.com/ec2/.

[31] A. Miyaji, M. Nakabayashi, and S. TAKANO, "New explicit conditions of elliptic curve traces for FR-reduction," *IEICE Transactions on fundamental*, pp. 1234–1243, 2001.

[32] S. Porubsk *et al.*, "Fermat-Euler theorem in algebraic number fields," *Journal of Number Theory*, vol. 60, no. 2, pp. 254–290, 1996.

[33] A. L. Ferrara, M. Green, S. Hohenberger, and M. Pedersen, "Practical short signature batch verification," in *The Cryptographer's Track at RSA Conference*, 2009, pp. 309–324.

Appendix A
Security Analysis

Here we present the security analysis for the PB-PMDP scheme. First, in the proposed scheme, we utilize PRP (π) and PRF (ψ) to compress the challenge, and thus reducing the communication cost. Instead of sending the set Q of c pairs of random indices and values to the CSP, the verifier sends only two keys k_1 and k_2 (over secure communication). Using π and ψ in this manner is proved to be secure [25].

For the *correctness* security requirement, we have previously shown the correctness of equation (1)). For the *soundness* security requirement, we will show that if a polynomial-time adversary $\mathcal{A}$ can win the data possession game (with non-negligible probability) with a challenger $\mathcal{C}$, then $\mathcal{A}$ is actually storing the n data copies $\widetilde{\mathbb{F}}$ in an uncorrupted state. For an adversary $\mathcal{A}$ to cheat the verifier, he has to respond with a malicious proof $\mathbb{P}' \neq \mathbb{P}$ and $\mathsf{Verify}(pk, \mathbb{P}')$ returns 1.

The soundness of the PB-PMDP scheme is based on the *unforgeability* of the used HLAs, which depends on the security of the computational Diffie-Hellman (CDH) and the discrete logarithm (DL) problems.

Definitions.

1) **CDH problem:** given $g,\ g^x,\ h \in \mathbb{G}$ for some group $\mathbb{G}$ and $x \in \mathbb{Z}_p$, compute h^x
2) **DL problem:** given $g,\ h \in \mathbb{G}$ for some group $\mathbb{G}$, find x such that $h = g^x$.

The following theorem proves the unforgeability of the HLAs used in the proposed PB-PMDP scheme. Our approach to prove the theorem is by investigating all possible combinations of malicious CSP responses $\langle \{\sigma', \mu'\}, \{\sigma, \mu'\}, \{\sigma', \mu\} \rangle$, and checking whether any of these combinations can pass the verification equation (1).

Theorem 1. *Assuming the hardness of both the CDH and the DL problems in bilinear groups, the verifier of the proposed PB-PMDP scheme accepts a response to a challenge vector only if a correctly computed proof* $\mathbb{P} = \{\sigma,\ \mu\}$*, where* $\mu = \{\mu_{ik}\}_{\substack{1 \le i \le n \\ 1 \le k \le s}}$ *is sent from the CSP.*

Proof. We prove the theorem by contradiction. The goal of an adversary $\mathcal{A}$ (malicious CSP) is to generate a response that is not correctly computed and pass the verification process done by a challenger $\mathcal{C}$ (verifier). Let $\mathbb{P}' = \{\sigma',\ \mu'\}$ be $\mathcal{A}$'s response, where $\mu' = \{\mu'_{ik}\}_{\substack{1 \le i \le n \\ 1 \le k \le s}}$. Let $\mathbb{P} = \{\sigma,\ \mu\}$ be the expected response from an honest CSP, where $\sigma = \prod_{(j,r_j) \in Q} \sigma_j^{r_j}$, $\mu = \{\mu_{ik}\}_{\substack{1 \le i \le n \\ 1 \le k \le s}}$, and $\mu_{ik} = \sum_{(j,r_j) \in Q} r_j \cdot \widetilde{b}_{ijk}$.

According to the correctness of PB-PMDP scheme, the expected proof $\mathbb{P} = \{\sigma,\ \mu\}$ satisfies the verification equation, *i.e.,*

$$\hat{e}(\sigma, g) = \hat{e}([\prod_{(j,r_j)\in Q} \mathcal{H}(ID_F||j)^{r_j}]^n \cdot \prod_{k=1}^{s} u_k^{\sum_{i=1}^{n}\mu_{ik}}, y).$$

Assume that $\sigma' \neq \sigma$, and σ' passes the verification equation, then we have

$$\hat{e}(\sigma', g) = \hat{e}([\prod_{(j,r_j)\in Q} \mathcal{H}(ID_F||j)^{r_j}]^n \cdot \prod_{k=1}^{s} u_k^{\sum_{i=1}^{n}\mu'_{ik}}, y).$$

Obviously, if $\mu'_{ik} = \mu_{ik}\ \forall(i,k)$, it follows from the above verification equations that $\sigma' = \sigma$, which contradicts our assumption. Let us define $\Delta\mu_{ik} = \mu'_{ik} - \mu_{ik}$ $(1 \leq i \leq n, 1 \leq k \leq s)$. It must be the case that at least one of $\{\Delta\mu_{ik}\}_{\substack{1\leq i\leq n\\1\leq k\leq s}}$ is nonzero. Dividing the verification equation for the malicious response by the verification equation for the expected response, we obtain

$$\begin{aligned}
\hat{e}(\sigma' \cdot \sigma^{-1}, g) &= \hat{e}(\prod_{k=1}^{s} u_k^{\sum_{i=1}^{n}\Delta\mu_{ik}}, y)\\
\hat{e}(\sigma' \cdot \sigma^{-1}, g) &= \hat{e}(\prod_{k=1}^{s} u_k^{x\cdot\sum_{i=1}^{n}\Delta\mu_{ik}}, g)\\
\sigma' \cdot \sigma^{-1} &= \prod_{k=1}^{s} u_k^{x\cdot\sum_{i=1}^{n}\Delta\mu_{ik}}.
\end{aligned}$$

We set $u_k = g^{\alpha_k}h^{\beta_k}$ for $\alpha_k, \beta_k \in \mathbb{Z}_p$, and thus

$$\begin{aligned}
\sigma' \cdot \sigma^{-1} &= \prod_{k=1}^{s} (g^{\alpha_k}h^{\beta_k})^{x\cdot\sum_{i=1}^{n}\Delta\mu_{ik}}\\
\sigma' \cdot \sigma^{-1} &= \prod_{k=1}^{s} (y^{\alpha_k}h^{x.\beta_k})^{\sum_{i=1}^{n}\Delta\mu_{ik}}\\
\sigma' \cdot \sigma^{-1} &= y^{\sum_{k=1}^{s}\alpha_k\cdot\sum_{i=1}^{n}\Delta\mu_{ik}} \,.\, h^{x\cdot\sum_{k=1}^{s}\beta_k\cdot\sum_{i=1}^{n}\Delta\mu_{ik}}\\
h^x &= (\sigma' \cdot \sigma^{-1} \cdot y^{-\sum_{k=1}^{s}\alpha_k\cdot\sum_{i=1}^{n}\Delta\mu_{ik}})^{\frac{1}{\sum_{k=1}^{s}\beta_k\cdot\sum_{i=1}^{n}\Delta\mu_{ik}}}.
\end{aligned}$$

Hence, we have found a solution to the CDH problem unless evaluating the exponent causes a division by zero. However, we noted that not all of $\{\Delta\mu_{ik}\}_{\substack{1\leq i\leq n\\1\leq k\leq s}}$ can be zero and the probability that $\beta_k = 0$ is $\frac{1}{p}$, which is negligible. Therefore, if $\sigma' \neq \sigma$, we can use the adversary $\mathcal{A}$ to break the CDH problem, and thus we guarantee that σ' must be equal to σ.

It is only the values $\mu' = \{\mu'_{ik}\}_{\substack{1\leq i\leq n\\1\leq k\leq s}}$ and $\mu = \{\mu_{ik}\}_{\substack{1\leq i\leq n\\1\leq k\leq s}}$ that can differ. Assume that the adversary $\mathcal{A}$ responds with $\sigma' = \sigma$ and $\mu' \neq \mu$. Now we have

$$\hat{e}(\sigma, g) = \hat{e}([\prod_{(j,r_j)\in Q} \mathcal{H}(ID_F||j)^{r_j}]^n \cdot \prod_{k=1}^{s} u_k^{\sum_{i=1}^{n}\mu_{ik}}, y) = \hat{e}(\sigma', g) =$$
$$\hat{e}([\prod_{(j,r_j)\in Q} \mathcal{H}(ID_F||j)^{r_j}]^n \cdot \prod_{k=1}^{s} u_k^{\sum_{i=1}^{n}\mu'_{ik}}, y),$$

from which we conclude that

$$\hat{e}([\prod_{(j,r_j)\in Q}\mathcal{H}(ID_F||j)^{r_j}]^n\cdot\prod_{k=1}^{s}u_k^{\sum_{i=1}^{n}\mu_{ik}},y)=\hat{e}([\prod_{(j,r_j)\in Q}\mathcal{H}(ID_F||j)^{r_j}]^n\cdot\prod_{k=1}^{s}u_k^{\sum_{i=1}^{n}\mu'_{ik}},y).$$

Thus,

$$\begin{aligned}1 &= \hat{e}(\prod_{k=1}^{s}u_k^{\sum_{i=1}^{n}\Delta\mu_{ik}},y)\\ 1 &= \prod_{k=1}^{s}(g^{\alpha_k}h^{\beta_k})^{\sum_{i=1}^{n}\Delta\mu_{ik}}\\ 1 &= g^{\sum_{k=1}^{s}\alpha_k\cdot\sum_{i=1}^{n}\Delta\mu_{ik}}\cdot h^{\sum_{k=1}^{s}\beta_k\cdot\sum_{i=1}^{n}\Delta\mu_{ik}}\\ h &= g^{-\frac{\sum_{k=1}^{s}\alpha_k\cdot\sum_{i=1}^{n}\Delta\mu_{ik}}{\sum_{k=1}^{s}\beta_k\cdot\sum_{i=1}^{n}\Delta\mu_{ik}}}.\end{aligned}$$

Now, we have found a solution to the DL problem unless evaluating the exponent causes a division by zero. However, the probability that $\beta_k = 0$ is $\frac{1}{p}$, which is negligible. Therefore, if there is at least one difference between $\{\mu'_{ik}\}_{\substack{1\le i\le n\\1\le k\le s}}$ and $\{\mu_{ik}\}_{\substack{1\le i\le n\\1\le k\le s}}$, we can use the adversary $\mathcal{A}$ to break the DL problem. As a result, we guarantee that $\{\mu'_{ik}\}$ must be equal to $\{\mu_{ik}\}$ $\forall(i,k)$. □

Data Extraction. We have shown that if a polynomial-time adversary $\mathcal{A}$ can win the data possession game (with non-negligible probability) with a challenger $\mathcal{C}$, then $\mathcal{A}$ is actually storing the data in an uncorrupted state. For the purpose of data extraction, the challenger $\mathcal{C}$ interacts with $\mathcal{A}$ to extract data blocks. Suppose that $\mathcal{C}$ challenges c blocks, namely the blocks with indices $\{j_1, j_2, \ldots, j_c\}$, then $\mathcal{A}$ responds with a proof $\mathbb{P}$ that contains $\sigma=\sigma_{j_1}^{r_{j_1}}\cdot\sigma_{j_2}^{r_{j_2}}\ldots\sigma_{j_c}^{r_{j_c}}$ and $\mu=\{\mu_{ik}\}_{\substack{1\le i\le n\\1\le k\le s}}$, where $\mu_{ik}=r_{j_1}\cdot\tilde{b}_{ij_1k}+r_{j_2}\cdot\tilde{b}_{ij_2k}+\cdots+r_{j_c}\cdot\tilde{b}_{ij_ck}$. The challenger $\mathcal{C}$ can extract the actual data blocks $\{\tilde{b}_{ijk}\}$ in polynomially-many interactions with $\mathcal{A}$. If the challenge-response phase has been repeated c times (each time we challenge c blocks), then there will be c proofs $\{\mathbb{P}^1, \mathbb{P}^2, \ldots, \mathbb{P}^c\}$. Thus, a system of linear equations can be constructed as follows.

$$\begin{aligned}\mu_{11}^1 &= r_{j_1}^1\cdot\tilde{b}_{1j_11}+r_{j_2}^1\cdot\tilde{b}_{1j_21}+\cdots+r_{j_c}^1\cdot\tilde{b}_{1j_c1}\\ &\vdots\\ \mu_{11}^c &= r_{j_1}^c\cdot\tilde{b}_{1j_11}+r_{j_2}^c\cdot\tilde{b}_{1j_21}+\cdots+r_{j_c}^c\cdot\tilde{b}_{1j_c1}\\ &\vdots\\ \mu_{ns}^c &= r_{j_1}^c\cdot\tilde{b}_{nj_1s}+r_{j_2}^c\cdot\tilde{b}_{nj_2s}+\cdots+r_{j_c}^c\cdot\tilde{b}_{nj_cs}\end{aligned}$$

Solving this system of linear equations yields the data blocks $\{\tilde{b}_{ijk}\}$.

Finally, the PB-PMDP scheme is secure against file swapping attack. The file identifier ID_F is embedded into the block tag, and thus the CSP cannot use blocks from different files and pass the auditing procedures even if the owner uses the same secret key x with all his files.