Provable Possession and Replication of Data over Cloud Servers

Ayad F.Barsoum and M.Anwar Hasan

Department of Electrical and Computer Engineering
University of Waterloo, Ontario, Canada.
afekry@engmail.uwaterloo.ca, ahasan@sisr.uwaterloo.ca

Abstract. Cloud Computing (CC) is an emerging computing paradigm that can potentially offer a number of important advantages. One of the fundamental advantages of CC is pay-as-you-go pricing model, where customers pay only according to their usage of the services. Currently, data generation is outpacing users' storage availability, thus there is an increasing need to outsource such huge amount of data. Outsourcing data to a remote Cloud Service Provider (CSP) is a growing trend for numerous customers and organizations alleviating the burden of local data storage and maintenance. Moreover, customers rely on the data *replication* provided by the CSP to guarantee the availability and durability of their data. Therefore, Cloud Service Providers (CSPs) provide storage infrastructure and web services interface that can be used to store and retrieve an unlimited amount of data with fees metered in GB/month. The mechanisms used for data replication vary according to the nature of the data; more copies are needed for critical data that cannot easily be reproduced. 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 pricing model is related to the replication strategy. Therefore, it is of crucial importance to customers to have a strong evidence that they actually get the service they pay for. Moreover, they need to verify that *all* their data copies are not being tampered with or partially deleted over time. Consequently, the problem of Provable Data Possession (PDP) has been considered in many research papers. Unfortunately, previous PDP schemes focus on a single copy of the data and provide no guarantee that the CSP stores multiple copies of customers' data. In this paper we address this challenging issue and propose Efficient Multi-Copy Provable Data Possession (EMC-PDP) protocols. We prove the security of our protocols against colluding servers. Through extensive performance analysis and experimental results, we demonstrate the efficiency of our protocols.

Keywords: Cloud computing, outsourcing data storage, data integrity, cryptographic protocols

1 Introduction

Cloud Computing (CC) is an emerging computing paradigm that can be viewed as a virtualized pool of computing resources (e.g. storage, processing power, memory, applications, services, and network bandwidth), where customers are provisioned and de-provisioned recourses as they need. CC represents the vision of providing computing services as public utilities like water and electricity. CC services can be categorized into [1]: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). The widely used model of CC services is the SaaS model in which the customers have access to the applications running on the cloud provider's infrastructure. Google Docs, Google Calendar, and Zoho Writer are publicly known examples of this model. In PaaS model, the customers can deploy their applications on the

provider infrastructure under condition that these applications are created using tools supported by the provider. IaaS model enables customers to rent and use the provider's resources (storage, processing, and network). Hence, the customers can deploy any applications including operating systems.

The considerable attention of Cloud Computing paradigm is due to a number of key advantages which make it a challenging research area in both academia and industry. This paradigm of Information Technology (IT) architecture supplies cost effective means of computing over a shared pool of resources, where users can avoid capital expenditure on hardware, software, and services as *they pay only for what they use* [1]. Moreover, CC model provides low management overhead and immediate access to abroad range of applications. In addition, maintenance cost is reduced as a third part is responsible for everything from running the cloud to storing data. It is not only the economic benefits that customers can gain from CC model, but also flexibility to scale up and down IT capacity over time to business needs. Furthermore, CC offers more mobility where customers can access information wherever they are, rather than having to remain at their desks.

CC allows organizations to store more data on remote servers than on private computer systems. Organizations will no longer be worried about constant server updates and other computing issues; they will be free to concentrate on innovations [2]. Outsourcing data storage to remote Cloud Service Provider (CSP) is a growing trend for more and more customers and organizations alleviating the burden of local data storage and maintenance. In addition, storing data remotely allows many authorized users to access the data from various different geographic locations making it more convenient to them. Also, some organizations may create large data files that must be achieved for many years but are rarely accessed, and so there is no need to store such files on the local storage of the organizations. According to a recent survey, IT outsourcing has grown by a staggering 79% as companies seek to reduce costs and focus on their core competencies [3].

However, the fact that data owners are no longer physically possess their sensitive data raises new formidable and challenging tasks related to data *security* and *integrity* protection in Cloud Computing. Data security can be achieved by encrypting sensitive data before outsourcing to remote servers. As such, it is a crucial demand of customers to have a strong evidence that the cloud servers still possess their data and it is not being tempered with or partially deleted over time, especially because the internal operation details of the CSP may not be known by cloud customers.

It is unarguable that, the completeness and correctness of customers' data in the cloud is being put at risk due to the following reasons. First, the CSP — whose goal is to make a profit and maintain a reputation — has an incentive to hide data loss or reclaim storage by discarding data that has not been or is rarely accessed. Second, a greedy CSP might delete some of the data or might not store all data in fast storage required by the contract with certain customers, i.e., place it on CDs or other offline media and thus using less fast storage. Third, the cloud infrastructures are subject to wide range of internal and external security threats. Examples of security breaches of cloud services appear from time to time [4, 5]. In short, although outsourcing data into the cloud is economically attractive for the cost and complexity of long-term large-scale data storage, it does not offer any guarantee on data *completeness* and *correctness*. This problem, if not properly handled, may hinder the successful deployment of cloud architecture.

Since customers' data has been outsourced to remote servers, efficient verification of the completeness and correctness of the outsourced data becomes a formidable challenge for data security in CC. The traditional cryptographic primitives for data integrity and availability based on hashing and signature schemes are not applicable on the outsourced data without having a local copy of the data. Of course, it is impractical for the clients to download all stored data in order to validate its integrity; this would require an expensive I/O cost and immense

communication overheads across the network. Therefore, clients need *efficient* techniques to verify the integrity of their outsourced data with minimum computation, communication, and storage overhead. Consequently, many researchers have focused on the problem of Provable Data Possession (PDP) and proposed different schemes to audit the data stored on remote servers.

Simply, Provable Data possession (PDP) is a technique for validating data integrity over remote servers. Ateniese et al. [6] have formalized a PDP model. In that model, the data owner pre-processes the data file to generate some metadata that will be used later for verification purposes through a challenge response protocol with the remote/cloud server. The file is then sent to be stored on an untrusted server, and the owner may delete the local copy of the file. Later, the server demonstrates that the data file has not been deleted or tampered with by responding to challenges sent from the verifier who can be the original data owner or other trusted entity that shares some information with the owner. Researchers have proposed different variations of PDP schemes under different cryptographic assumptions [7–14]. We will present various PDP schemes in the literature survey section, and will elaborate how they vary from different perspectives.

Unfortunately, previous PDP schemes focus on a *single copy* of the file and provide no proof that the CSP stores multiple copies of the owner's file. In this paper we address this challenging problem, propose two Efficient Multi-Copy Provable Data Possession (EMC-PDP) protocols, and prove the security (*correctness* and *soundness*) of our protocols against colluding servers. Extensive performance analysis — which is validated through implementation and experimental results — illustrates the efficiency of our protocols.

Curtmola et al. [15] proposed a Multiple-Replica PDP (MR-PDP) scheme, which is the only attempt in the literature that creates multiple replicas of owner's file and audit them. Through extensive investigation, we will detail the various features and limitations of the MR-PDP model. Unfortunately, Curtmola et al. [15] did not address how the authorized users of the data owner can access the file copies from the cloud servers noting that the internal operations of the CSP are opaque. This issue is not handled in their protocol, and thus we consider the protocol to be incomplete. We will demonstrate that our protocols are *complete* and outperform the MR-PDP model [15] from both the computations and communications cost. The MR-PDP is investigated in more details in section 5.2.

The remainder of the paper is organized as follows: in Section 2 we provide our research problem definitions, motivations, challenges, and our main contributions. Section 3 contains an extensive literature survey for the previous schemes of remote data integrity, the features of these schemes, and their limitations. Our system model and design goals are presented in Section 4. In Section 5 we present and analyze the basic natural scheme for multi-copy provable data possession and the MR-PDP scheme due to Curtmola et al. [15]. Our EMC-PDP schemes are elaborated in Section 6. In Section 7 we prove the security of our proposed schemes. The performance analysis and experimental results are done in Section 8. Our concluding remarks are given in Section 9.

2 Problem Definition, Motivation, and Main Contributions

Users resort to data *replication* to ensure the availability and durability of their sensitive data, especially if it cannot easily be reproduced. As a simple example, when we are writing a research paper, we are very careful to keep multiple copies of our paper to be able to recover it in case of any failure or physical damage. Likewise, organizations, governments, and universities replicate their financial, personal, and general data to guarantee its availability and durability over time. In the Cloud Computing paradigm, customers rely on the CSP to undertake the data replication task relieving the burden of local data storage and maintenance, but they have to pay for their usage of the CSP's storage infrastructure. On the other side, cloud customers should be *securely* and *efficiently* convinced that the CSP is actually possessing *all* data copies that are agreed

upon, these data copies are complete and intact, and thus customers are getting the service they are paying for. Therefore, in this paper we address the problem of creating multiple copies of owner's data file over untrusted CSP and auditing all these copies to verify their *completeness* and *correctness*.

2.1 Motivation and Challenges

The mechanisms used for data *replication* vary according to the nature of the data; more copies are needed for critical data that cannot easily be reproduced. The pricing model of the CSPs is related to the replication strategy. For example, Amazon S3 standard storage strategy [16] maintains copies of customers' data on multiple servers across multiple data centers, while with Amazon Reduced Redundancy Storage (RRS) strategy — which enables customers to reduce their costs — noncritical, reproducible data is stored at reduced level of redundancy. As a consequence, the pricing for the Amazon S3 standard storage is approximately 50% higher than that of the RRS. Cloud servers can collude to cheat the customers by showing that they are storing all copies, while in reality they are storing a single copy. Therefore, cloud customers need *secure* and *efficient* techniques to ensure that the CSP is actually keeping *all* data copies that are agreed upon, these copies are not corrupted, and thus they pay for real services.

2.2 Contributions

Our contributions can be summarized as the following:

1. We propose two Efficient Multi-Copy Provable Data Possession (EMC-PDP) protocols. These protocols efficiently provide the cloud customers with a strong evidence that the CSP is in reality possessing *all* data copies that are agreed upon and these copies are intact.
2. We prove the security (*correctness* and *soundness*) of our protocols against colluding servers. Cloud servers can provide valid responses to verifier's challenges only if they actually have *all* data copies in an uncorrupted state.
3. We justify the performance of our proposed protocols through concrete analysis and comparison with the state-of-the-art.
4. We implement our proposed protocols using cryptographic libraries. The experimental results validate our performance analysis.

To the best of our knowledge, the proposed protocols are the first complete and efficient protocols that address the storage integrity of multiple data copies over Cloud Computing.

3 Literature Survey

3.1 Provable Data Possession (PDP)

Provable data possession (PDP) is a methodology for validating the integrity of data in outsourcing storage service. The fundamental goal of the PDP scheme is to allow a verifier to efficiently, periodically, and securely validate that a remote server — which supposedly stores the owner's potentially very large amount of data — is not cheating the verifier. The problem of data integrity over remote servers has been addressed for many years and there is a simple solution to tackle this problem as follows. First, the data owner computes a message authentication code (MAC) of the whole file before outsourcing to a remote server. Then, the owner keeps only the computed MAC on his local storage, sends the file to the remote server, and deletes the local copy of the file. Later, whenever a verifier needs to check the data integrity, he sends a request

to retrieve the file from the archive service provider, re-computes the MAC of the whole file, and compares the re-computed MAC with the previously stored value. Alternatively, instead of computing and storing the MAC of the whole file, the data owner divides the file F into blocks $\{b_1, b_2, \ldots, b_m\}$, computes a MAC σ_i for each block $b_i : \sigma_i = MAC_{sk}(i||b_i)_{1 \leq i \leq m}$, sends both the data file F and the MACs $\{\sigma_i\}_{1 \leq i \leq m}$ to the remote/cloud server, deletes the local copy of the file, and stores only the secret key sk. During the verification process, the 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 [7]. The rationale behind the second approach is that checking part of the file is much easier than the whole of it. However both approaches suffer from severe drawback; the communication complexity is linear with the queried data size which is impractical especially when the available bandwidth is limited.

◇ **PDP Schemes of Deswarte et al.** Deswarte et al. [8] thought of a better solution by using two functions f and H'. H' is a one-way function and f is another function such that $f(C, H'(File)) = H(C||File)$, where H is any secure hash function and C is a random challenge number sent from the verifier to the remote server. Thus, the data owner has to compute $H'(File)$ and store it on his local storage. To audit the file, the verifier generates a random challenge C, computes $V = f(C, H'(File))$, and sends C to the remote server. Upon receiving the challenge C, the server computes $R = H(C||File)$ and sends the response R to the verifier. To validate the file integrity, the verifier checks $V \stackrel{?}{=} R$. At least one of the two functions f and H' must be kept secret because if both were public, it would be easy for a malicious server to compute and store only $H'(File)$ that is not the entire file, and then dynamically responds with a valid value $f(C, H'(File))$ that is not the expected one $H(C||File)$.

Unfortunately, Deswarte et al. [8] have not found such functions f, H, and H' satisfying the desired verification rule. To workaround this problem, a finite number $\widetilde{N}$ of random challenges are generated offline for the file to be checked, the corresponding responses $H(C_i||File)_{1 \leq i \leq \widetilde{N}}$ are pre-computed offline as well, and then the pre-computed responses are stored on the verifier local storage. To audit the file, one of the $\widetilde{N}$ challenges is sent to the remote server and the response received from the server is compared with the pre-computed response which is previously stored on the verifier side. However, this solution limits the number of times a particular data file can be checked by the number of random challenges $\widetilde{N}$. Once all random challenges $\{C_i\}_{1 \leq i \leq \widetilde{N}}$ are consumed, the verifier has to retrieve the data file F from the storage server in order to compute new responses $H(C'_i||File)$, but this is unworkable.

Deswarte et al.[8] proposed another protocol to overcome the problem of limited number of audits per file. In their protocol the data file is represented as an integer m. Figure 1 illustrates the scheme proposed in [8]

There are two main limitations in the protocol of Deswarte et al. [8]:

- In each verification, the remote server has to do the exponentiation over the entire file. Thus, if we are dealing with huge files, e.g., in order of Terabytes (as most practical applications require) this exponentiation will be heavy.
- Storage overhead on the verifier side; it has to store some metadata for each file to be checked. This could be a challenge for the verifier if it uses small devices, e.g., a PDA or a cell phone with limited storage capacity.

◇**More RSA Based PDP Schemes.** Filho et al. [9] proposed a scheme to verify data integrity using the RSA-based Homomorphic hash function. A function H is Homomorphic if, given two

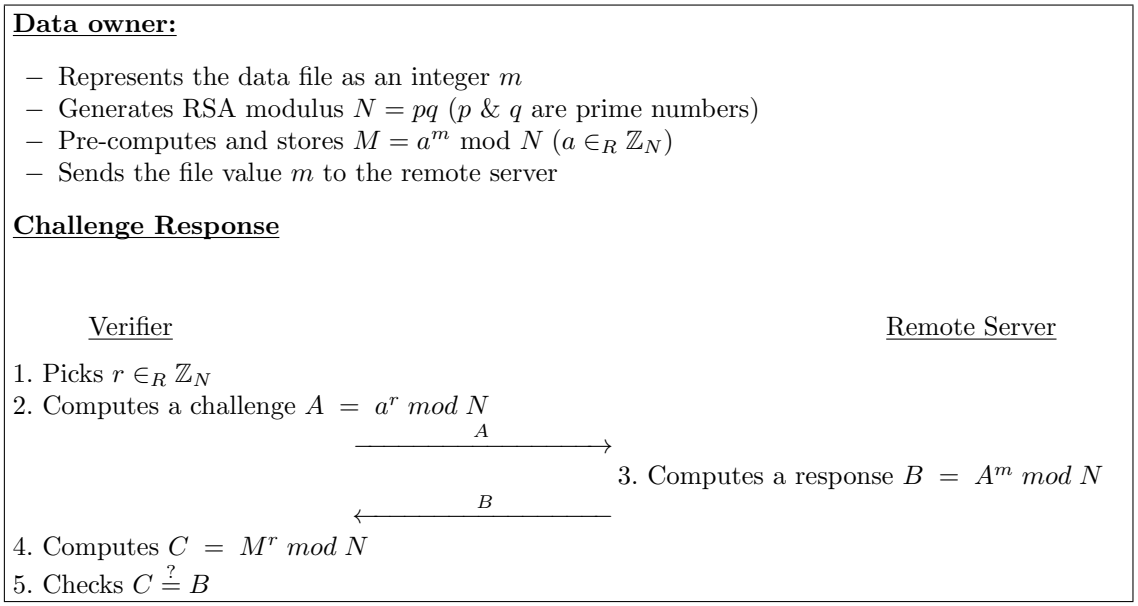

Fig. 1. The PDP protocol by Deswarte et al. [8].

operation + and × , we have $H(d+d') = H(d) \times H(d')$. The protocol proposed in [9] is illustrated in figure 2.

Note that the response $R = H(d)$ is a homomorphic function in the data file d; $H(d+d') \equiv r^{d+d'} \equiv r^d r^{d'} \equiv H(d)H(d') \ mod \ N$. To find a collision for this hash function, one has to find two messages d, d' such that $r^d \equiv r^{d'}$, i.e., $r^{d-d'} \equiv 1 \ mod \ N$. Thus, $d - d'$ must be multiple of $\phi(N)$. Finding such two messages d, d' is belived to be difficult since the factorization of N is unknown.

The limitations of the protocol proposed in [9] are similar to those of the protocol in [8]: the archive service provider has to exponentiate the entire data file plus the storage overhead on the verifier side.

To circumvent the problem of exponentiating the entire file Sebé et al. [10] proposed to verify data integrity by first fragmenting the file into blocks, fingerprinting each block, and then using an RSA-based hash function on the blocks. Thus, the file F is divided into a set of m blocks: $F = \{b_1, b_2, \ldots, b_m\}$, where m fingerprints $\{M_i\}_{1 \leq i \leq m}$ are generated for the file and stored on the verifier local storage. Their proposal does not require the exponentiation of the entire file. Figure 3 demonstrates the protocol proposed by Sebé et al. [10].

Limitations. Although the protocol proposed by Sebé et al. [10] does not require exponentiation of the entire file, a local copy of the fingerprints — whose size is linear in the number of file blocks — must be stored on the verifier side. The verifier has to store the fingerprints $\{M_i\}_{1 \leq i \leq m}$, each of size $|N|$ bits consuming $m|N|$ bits from the verifier local storage, which may impede the verification process when using small devices like PDAs or cell phones.

⋄Data Storage Commitment Schemes. Golle et al. [11] proposed a scheme to verify data storage commitment, a concept that is weaker than integrity. They investigated "storage-enforcing commitment scheme". Through their scheme a storage server demonstrates that it is making use of storage space as large as the client's data, but not necessarily the same exact

Data owner:

- Generates RSA modulus $N = pq$ (p & q are prime numbers)
- Computes $\phi(N) = (p-1)(q-1)$
- Pre-computes and stores $h(d) = d \bmod \phi(N)$ (d is the data file)
- Sends the data file d to the remote server

Challenge Response

Verifier		Remote Server
1. Picks $r \in_R \mathbb{Z}_N$		
	$\xrightarrow{r}$	
		2. Computes a response $R = H(d) = r^d \ mod \ N$
	$\xleftarrow{R}$	
3. Computes $R' = r^{h(d)} \ mod \ N$		
4. Checks $R' \stackrel{?}{=} R$		

Fig. 2. The PDP protocol by Filho et al. [9].

Data owner:

- Generates RSA modulus $N = pq$ (p & q are prime numbers)
- Computes $\phi(N) = (p-1)(q-1)$
- Splits the data file F into m blocks: $F = \{b_1, b_2, \ldots, b_m\}$
- Pre-computes and stores $M_i = b_i \bmod \phi(N)$ $(1 \leq i \leq m)$
- Sends the data file F to the remote server

Challenge Response

Verifier		Remote Server
1. Picks $a \in_R \mathbb{Z}_N$		
2. Generates $l (\leq m)$ random values $\{c_i\}_{1 \leq i \leq l}$		
	$\xrightarrow{a, \{c_i\}_{1 \leq i \leq l}}$	
		3. Computes $r = \sum_{i=1}^{l} c_i \cdot b_i$
		4. Computes $R = a^r \ mod \ N$
	$\xleftarrow{R}$	
5. Computes $r' = \sum_{i=1}^{l} c_i \cdot M_i \ mod \ \phi(N)$		
6. Computes $R' = a^{r'} \ mod \ N$		
7. Checks $R' \stackrel{?}{=} R$		

Fig. 3. The protocol by Sebé et al. [10].

data. The storage server does not directly prove that it is storing a file F, but proves that it has committed sufficient resources to do so. Their scheme is based on n-Power Computational Diffie-Hellman (n-PCDH) assumption: for a group $\mathbb{Z}_p$ with a generator g, there is no known probabilistic polynomial time algorithm A that can compute g^{x^n} given $g^x, g^{x^2}, \ldots, g^{x^{n-1}}$ with non-negligible probability. The main scheme proposed by Golle et al. [11] is illustrated in figure 4.

Setup

- File $F = \{b_1, b_2, \ldots, b_m\}, b_i \in \mathbb{Z}_p$
- Let $n = 2m + 1$
- Secret key $sk = x \in_R \mathbb{Z}_p$
- Public key $pk = (g^x, g^{x^2}, \ldots, g^{x^n}) = (g_1, g_2, \ldots, g_n)$
- Data owner computes and stores $f_0 = \prod_{i=1}^{m} g_i^{b_i} \bmod p$

Challenge Response

Verifier		Remote Server
1. Picks a random $k \in [0, m]$		
	$\xrightarrow{k}$	
		3. Computes $f_k = \prod_{i=1}^{m} g_{i+k}^{b_i}$
	$\xleftarrow{f_k}$	
4. Checks $f_0^{x^k} \stackrel{?}{=} f_k$		

Fig. 4. The protocol by Golle et al. [11].

Since each file block $b_i \in \mathbb{Z}_p$ can be represented by $\lceil \log_2\ p \rceil$ bits, then the total number of bits to store the file $F = m\lceil \log_2\ p \rceil$ bits. For the storage server to cheat by storing all the possible values of f_k ($m + 1$ values), it needs $(m + 1)\lceil \log_2\ p \rceil$ bits which is slightly larger than the size of the original file.

Limitations. The guarantee provided by the proposed protocol in [11] is weaker than data integrity since it only ensures that the server is storing something at least as large as the original data file but not necessarily the file itself. In addition, the verifier's public key is about twice as large as the data file.

⋄Privacy-Preserving PDP Schemes. Shah et al. [12, 13] proposed privacy-preserving PDP protocols. Using their schemes, an external Third Party Auditor (TPA) can verify the integrity of files stored by a remote server without knowing any of the file contents. The data owner first encrypts the file, then sends both the encrypted file along with the encryption key to the remote server. Moreover, the data owner sends the encrypted file along with a key-commitment that fixes a value for the key without revealing the key to the TPA. The primary purposes of the schemes proposed in [12, 13] are to ensure that the remote server is correctly possessing

the client's data along with the encryption key, and to prevent any information leakage to the TPA which is responsible for the auditing task. Thus, clients — especially with constrained computing resources and capabilities — can resort to external audit party to check the integrity of outsourced data, and this third party auditing process should bring in no new vulnerabilities towards the privacy of client's data. In addition to the auditing task of the TPA, it has another primary task which is extraction of digital contents. For the auditing task, the TPA interacts with the remote server to check that the stored data is intact. For the extraction task, the TPA interacts with both the remote server and the data owner to first check that the data is intact then delivers it to the owner. The protocols proposed by Shah et al. [12, 13] are illustrated in figure 5.

The protocols proposed by Shah et al. [12, 13] have the following key limitations:

- The number of times a particular data item can be verified is limited and must be fixed beforehand.
- Storage overhead on the TPA; it has to store $\widetilde{N}$ hash values for each file to be audited.
- Lack of support for *stateless* verification; the TPA has to update its state (the list L) between audits to prevent using the same random number or the same HMAC twice.
- Very high communication complexity to retrieve $E_K(F)$ if the TPA wants to regenerate a new list of hash values to achieve unbounded number of audits.

⋄PDP in Database Context. In database outsourcing scenario, the database owner stores data at a storage service provider and the database users send queries to the service provider to retrieve some tuples/records that match the issued query. Data integrity is an imperative concern in the database outsourcing paradigm; when a user receives a query result from the service provider, he wants to be convinced that the received tuples are not being tampered with by a malicious service provider. Mykletun et al. [14] investigated the notion of signature aggregation to validate the integrity of the query result. Signature aggregation enables bandwidth- and computation-efficient integrity verification of query replies. In the scheme presented in [14], each database record is signed before outsourcing the database to a remote service provider. Mykletun et al. [14] provided two aggregation mechanisms: one is based on RSA [17] and the other is based on BLS signature [18]. For the scheme based on the RSA signature, each record in the database is signed as: $\sigma_i = h(b_i)^d \ mod \ N$, where h is a one-way hash function, b_i is the data record, d is the RSA private key, and N is the RSA modulus. A user issues a query to be executed over the outsourced database, the server processes the query and computes an aggregated signature $\sigma = \sum_{i=1}^{t} \sigma_i \bmod N$, where t is the number of records in the query result. The server sends the query result along with the aggregated signature to the user. To verify the integrity of the received records, the user checks $\sigma^e \stackrel{?}{=} \prod_{i=1}^{t} \sigma_i \bmod N$, where e is the RSA public key. The second scheme proposed by Mykletun et al. [14] which is based on the BLS signature [18] is similar to the first scheme but the record signature $\sigma_i = h(m_i)^x$, where $x \in_R \mathbb{Z}_p$ is a secret key. To verify the integrity of the received records, the user checks $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(\prod_{i=1}^{t} h(b_i), y)$, where g is a generator of the group $\mathbb{Z}_p$, $y = g^x (public\ key)$, and $\hat{e}$ is a computable bilinear map that will be explained later.

Although data integrity (*correctness*) is an imperative concern in the database outsourcing paradigm, *completeness* is another crucial demand for database users. Completeness means that the service provider should send *all* records that satisfy the query criteria not just subset of them. The schemes proposed by Mykletun et al. [14] did not fulfill the completeness requirement. The completeness problem has been addressed by other researchers (see for example [3, 19]).

We emphasize that the techniques based on aggregated signatures [14] would fail to provide *blockless* verification, which is needed by any efficient PDP scheme. Indeed, the verifier has to

Setup

- Data owner sends a key K and the encrypted file $E_K(F)$ to the remote server
- Data owner sends a key-commitment value g^K and the encrypted file $E_K(F)$ to the TPA (g is a generator for $\mathbb{Z}_p$)
- The TPA generates a list L of random values and HMACs: $L = \{(R_i, \widetilde{H}_i)\}_{1 \le i \le \widetilde{N}}$, $\widetilde{H}_i = \text{HMAC}(R_i, E_K(F))$, and R_i is a random number.
- TPA keeps L, $H(E_K(F))$, g^K, and can discard $E_K(F)$ (H is a secure hash function)

TPA **Remote Server**

Checking Data Integrity

1. Picks any $R_i, \widetilde{H}_i$ from L

and $L = L \backslash \{(R_i, \widetilde{H}_i)\}$

$\xrightarrow{R_i}$

2. Computes $\widetilde{H}_s$=HMAC$(R_i, E_K(F))$

$\xleftarrow{\widetilde{H}_s}$

3. Checks $\widetilde{H}_i \stackrel{?}{=} \widetilde{H}_s$

Checking Key Integrit

1. Generates $\beta \in_R \mathbb{Z}_p$

$\xrightarrow{g^\beta}$

2. Computes $W_s = (g^\beta)^K$

$\xleftarrow{W_s}$

3. Checks $(g^K)^\beta \stackrel{?}{=} (W_s)$

Data Extraction

$\xleftarrow{D_s = E_K(F)}$

- Checks hashing of its local cached copy:

$H(E_K(F)) \stackrel{?}{=} H(D_s)$. If valid, sends $E_K(F)$ to the owner

Key Extraction

- Assume that the owner and the server agree on a shared random secret X

$\xleftarrow{K+X,\ g^X}$

- Checks $g^{K+X} \stackrel{?}{=} g^K \cdot g^X$. If valid, sends $K + X$ to the owner
- Owner gets $K = (K + X) - X$

Fig. 5. The protocols by Shah et al. [12, 13].

have the ability to verify data integrity even though he does not possess any of the file blocks. The schemes proposed in [14] depend on the retrieved records of the query result to verify the integrity of the outsourced database. *Blockless* verification is a main concern to minimize the required communication cost over the network.

Ateniese et al. [6] proposed a model to overcome some of the limitations of the previous protocols: limited number of audits per file determined by fixed challenges that must be specified in advance, expensive server computation by doing the exponentiation over the entire file, storage overhead on the verifier side by keeping some metadata to be used later in the auditing task, high communication complexity, and lack of support for blockless verification. Ateniese et al. [6] proposed a PDP model in which the data owner fragments the file F into blocks $\{b_1, b_2, \ldots, b_m\}$ and generates metadata (a tag) for each block to be used for verification. The file is then sent to be stored on a remote/cloud server which may be untrusted and the data owner may delete the local copy of the file. The remote server provides a proof that the data has not been tampered with or partially deleted by responding to challenges sent from the verifier. The scheme proposed by Ateniese et al. [6] provides *probabilistic* guarantee of data possession, where the verifier checks a random subset of stored file blocks with each challenge (spot checking).

⋄PDP Schemes Based on Homomorphic Verifiable Tags. Ateniese et al. [6] proposed using Homomorphic Verifiable Tags(HVTs)/Homomorphic Linear Authenticators(HLAs) as the basic building blocks of their scheme. In short, the HVTs/HLAs are unforgeable verification metadata constructed from the file blocks in such a way that the verifier can be convinced that a linear combination of the file blocks is accurately computed by verifying only the aggregated tag/authenticator. In their work, Ateniese et al. [6] differentiate between the concept of *public verifiability* and *private verifiability*. In public verifiability anyone — not necessarily the data owner — 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 side, private verifiability allows only the original owner (or a verifier with whom the original owner shares a secret key) to perform the auditing task. Ateniese et al. [6] proposed two main PDP schemes: Sampling PDP (S-PDP) and Efficient PDP (E-PDP) schemes. In fact, there is a slight difference between the S-PDP scheme and the E-PDP model, but the E-PDP model provides weaker guarantee of data possession. The E-PDP scheme only guarantees possession of the *sum* of file blocks and not necessarily possession of each one of the blocks being challenged. Both protocols proposed in [6] are illustrated in figure 6.

Although the models proposed by Ateniese et al. [6] have addressed many drawbacks of the previous protocols, they still have some limitations:

- HVTs in [6] are based on RSA and thus are relatively long; the HVT for each file block is in order of $|N|$ bits. Therefore, to achieve 80-bit security level, the generated tag should be of size 1024 bits.
- The time is takes to generate the tags is too long [7].
- Since there is no indicator for the file identifier in the block tag, a malicious server can cheat by using blocks from different files if the data owner uses the same secret keys d and v for all his files.
- Ateniese et al. [6] calculated that for a malicious server to attack their E-PDP scheme, it has to store 10^{140} values to be able to cheat with probability 100% [1]. Shacham and Waters [20] presented a simple attack against the E-PDP scheme which enables a malicious server to cheat with probability 91% requiring no more storage than an honest server to store the file.

[1] 10^{140} values if the number of the file blocks = 1000 and the number of challenged blocks = 101

I. S-PDP scheme

Setup

- $N = pq$ is the RSA modulus (p & q are prime numbers)
- g is a generator of QR_N (QR_N is the set of quadratic residues modulo N)
- Public key $pk = (N, g, e)$, secret key $sk = (d, v)$, $v \in_R \mathbb{Z}_N$, and $ed \equiv 1 \bmod (p-1)(q-1)$
- π is a pseudo-random permutation, f is a pseudo-random function, and H is a hash function
- File $F = \{b_1, b_2, \ldots, b_m\}$
- Data owner generates a tag T_i for each block b_i: $T_i = (H(v||i) \cdot g^{b_i})^d \bmod N$
- Data owner sends the data file $F = \{b_i\}$ and the tags $\{T_i\}$ $(1 \le i \le m)$ to the remote server

Challenge Response

Verifier — Remote Server

1. Picks two keys k_1(key for π), k_2(key for f), c(# of blocks to be challenged), and $g_s = g^s \bmod N (s \in_R \mathbb{Z}_N)$

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

2. Computes the challenged block indices: $i_j = \pi_{k_1}(j)_{1 \le j \le c}$
3. Computes the random values: $a_j = f_{k_2}(j)_{1 \le j \le c}$
4. Computes $T = \prod_{j=1}^{c} T_{i_j}^{a_j} \bmod N$
5. Computes $\rho = H(g_s^{\sum_{j=1}^{c} b_{i_j} \cdot a_j} \bmod N)$

$\xleftarrow{T,\ \rho}$

6. Computes $i_j = \pi_{k_1}(j), a_j = f_{k_2}(j) (1 \le j \le c)$
7. Computes $\tau = \dfrac{T^e}{\prod_{j=1}^{c} H(v||i_j)^{a_j}}$
8. Checks $H(\tau^s \ mod \ N) \stackrel{?}{=} \rho$

II. E-PDP scheme

The only difference between the E-PDP and the S-PDP is that : $\{a_j\}_{1 \le j \le c} = 1$, and thus

- Step 4 : $T = \prod_{j=1}^{c} T_{i_j} \bmod N$
- Step 5 : $\rho = H(g_s^{\sum_{j=1}^{c} b_{i_j}} \bmod N)$
- Step 7 : $\tau = \dfrac{T^e}{\prod_{j=1}^{c} H(v||i_j)}$

Fig. 6. The S-PDP and E-PDP protocols by Ateniese et al. [6].

Recently, Ateniese et al. [21] showed that the HLAs can be constructed from homomorphic identification protocols. They provided a "compiler-like" transformation to build HLAs from homomorphic identification protocols and showed how to turn the HLA into a PDP scheme. As a concrete example, they applied their transformation to a variant of an identification protocol proposed by Shoup [22] yielding a factoring-based PDP scheme.

◇**Comparison.** We present a comparison between the previous PDP schemes in table 1. The comparison is based on the following:

- Owner pre-computation: the operations performed by the data owner to process the file before being outsourced to a remote server.
- Verifier storage overhead: the extra storage required to store some metadata on the verifier side to be used later during the verification process.
- Server storage overhead: the extra storage on the server side required to store some metadata — not including the original file — sent from the owner.
- Server computation: the operations performed by the server to provide the data possession guarantee.
- Verifier computation: the operations performed by the verifier to validate the server's response.
- Communication cost: bandwidth required during the challenge response phase.
- Unbounded challenges: to indicated whether the scheme allows unlimited number to audit the data file.
- Fragmentation: to indicate whether the file is treated as one chunk or divided into smaller blocks.
- Type of guarantee: to indicate whether the guarantee provided from the remote server is *deterministic* guarantee which requires to access all file blocks or *probabilistic* guarantee that depends on spot checking.
- Prove data possession: to indicate whether the scheme proves the possession of the file itself or proves that the server is storing something at least as large as the original file.

We will use the notations EXF to indicate the EXponentiation of the entire File, DET to indicate deterministic guarantee, and PRO to indicate probabilistic guarantee. For simplicity, the security parameter is not included as a factor for the relevant costs.

3.2 Proof of Retrievability (POR)

A Proof of Retrievability (POR) scheme is an orthogonal/a complementary approach to a Provable Data Possession (PDP) system. A POR scheme is a challenge-response protocol which enables a remote server to provide an evidence that a verifier can retrieve or reconstruct the entire data file from the responses that are reliably transmitted from the server. The main idea of the POR schemes is to apply erasure code to data files before outsourcing to allow more error-resiliency. Thus, if it is a critical demand to detect any modification or deletion of tiny parts of the data file, then erasure code could be used before outsourcing.

The work done by Juels and Kaliski [23] was from the first papers to consider formal models for POR schemes. In their model, the data is first encrypted then disguised blocks (called sentinels) are embedded into the ciphertext. The sentinels are hidden among the regular file blocks in order to detect data modification by the server. In the auditing phase, the verifier requests for randomly picked sentinels and checks whether they are corrupted or not. If the server corrupts or deletes parts of the data, then sentinels would also be influenced with a certain probability. The main limitation of the scheme in [23] is that it allows only a limited number of challenges on the

Scheme	[8]	[9]	[10]	[11]	[12, 13]	[6]
Owner pre-computation	EXF	$O(1)$	$O(m)$	$O(m)$	$O(1)$	$O(m)$
Verifier storage overhead	$O(1)$	$O(1)$	$O(m)$	$O(1)$	$O(\widetilde{N})$	-
Server storage overhead	-	-	-	-	-	$O(m)$
Server computation	EXF	EXF	$O(c)$	$O(m)$	$O(1)$	$O(c)$
Verifier computation	$O(1)$	$O(1)$	$O(c)$	$O(1)$	$O(1)^{\dagger}$	$O(c)$
Communication cost	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Unbounded challenges	✓	✓	✓	✓	×	✓
Fragmentation	×	×	✓	✓	×	✓
Type of guarantee	DET	DET	DET/ PRO	DET	DET	PRO‡
Prove data possession	✓	✓	✓	×	✓	✓

Table 1. Comparison of PDP schemes for a file consisting of m blocks, c is the number of challenged blocks, and $\widetilde{N}$ is a finite number of random challenges.† Verifier *pre*-computation is $O(\widetilde{N})$ to generate a list L of HMACs. ‡ [6] can be easily modified to support deterministic guarantee

data files, which is specified by the number of sentinels embedded into the data file. This limited number of challenges is due to the fact that sentinels and their position within the file must be revealed to the server at each challenge and the verifier cannot reuse the revealed sentinels.

Schwartz and Miller [24] proposed using algebraic signature to verify data integrity across multiple servers using error-correcting codes. Through keyed algebraic encoding and stream cipher encryption, they are able to detect file corruptions. The main limitation of their proposal is that the communication complexity is linear with respect to the queried data size. Moreover, the security of their proposal is not proven and remains in question [7]. Shacham and Waters [20] proposed a compact proof of retrievability model that enables the verifier to unboundedly challenge the server solving the limitation of Juels and Kaliski [23]. The main contribution of Shacham and Waters [20] is the construction of HLAs that enable the server to aggregate the tags of individual file blocks and to generate a single short tag as a response to the verifier's challenge. Shacham and Waters [20] proposed two HLAs: one is based on the Pseudo-Random Function (PRF), and the other is based on the BLS signature [18]. Other various POR schemes can be found in the literature (see for example [25–28]).

3.3 Dynamic Provable Data Possession (DPDP)

The PDP and POR schemes focus on static or warehoused data which is essential in numerous different applications such as libraries, archives, and astronomical/medical/scientific/legal repositories. On the other side, Dynamic Provable Data Possession (DPDP) schemes investigate the dynamic file operations such as update, delete, append, and insert operations. There are some DPDP constructions in the literature satisfying different system requirements. Ateniese et al. [29] proposed a DPDP model based on cryptographic hash function and symmetric key encryption. The main drawback of their scheme is that the number of updates and challenges is limited and fixed in advance. Moreover, their model does not support block insertion operation. Erway et al. [30] extended the work of Ateniese et al. [29] to support data dynamics using authenticated skip list. However, the efficiency of their scheme remains in question. Wang et al. [31] presented a DPDP scheme by integrating the compact proof of retrievability model due to Shacham and Waters [20] and Merkle Hash Tree (MHT). Zhu et al. [32] addressed the construction of DPDP schemes on hybrid clouds to support scalability of service and data migration. A hybrid cloud is a

CC deployment model in which an organization provides and handles some internal and external resources. For example, an organization can use a public cloud service as Amazon EC2 [33] to perform the general computation, while the data files are stored within the organization's local data center in a private cloud. Zhu et al. [32] proposed two DPDP schemes: Interactive Provable Data Possession (IPDP) scheme and Cooperative Provable Data Possession (CPDP) scheme.

It is important to note that none of the above approaches (PDP, POR, DPDP) address the problem of creating multiple copies of data file over cloud servers, auditing all these copies to verify their completeness and correctness, and providing a strong evidence to the owners that they are getting what they are paying for. There are some previous work on maintaining the file copies throughout distributed systems to achieve availability and durability, but none of them guarantee that multiple copies of the data file are actually maintained [34, 35].

4 System Model and Design Goals

4.1 System Model

In our work we consider a model of cloud computing data storage system consisting of three main components as illustrated in figure 7: (i) a data owner — a customer or an organization originally possessing large amount of data to be stored in the cloud; (ii) Cloud Servers (CSs) managed by CSP which provides paid storage space on its infrastructure to store owner's files; and (iii) authorized users — a set of owner's clients allowed to use the owner's files and share some keying material with the data owner. The authorized users receive the data from the cloud servers in an encrypted form, and using the secret key — shared with the owner — they get the plain data. The authorization between the data owner and the legal users is out of our scope, and we assume that it is appropriately done. Through the paper we do not differentiate between cloud server and cloud service provider.

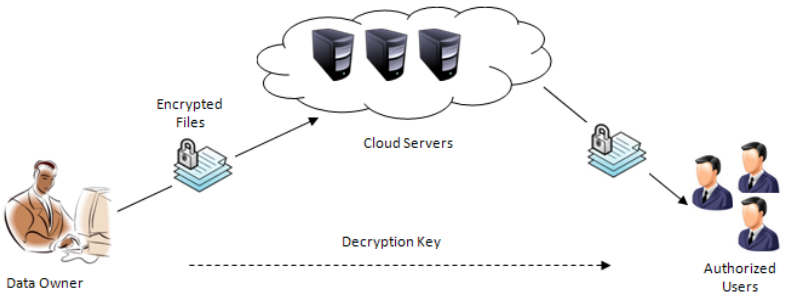

Fig. 7. Cloud Computing Data Storage System Model

There are many applications that can be envisioned to adopt this model of outsourced data storage system. For eHealth applications, a database containing sensitive and large amount of information about patients' medical history is to be stored on the cloud servers. We can consider the eHealth organization to be the data owner and the physicians to be the authorized users with appropriate access right to the database. Financial applications, scientific applications, and educational applications containing sensitive information about students' transcripts can also be envisioned in similar settings.

4.2 Design Goals

From the extensive literature survey of the previous PDP schemes, we have explored and investigated the various features and limitations of such schemes. Now, we aim at designing efficient and secure protocols that overcome the identified limitations in the previous models and — at the same time — to address the problem of creating multiple copies of data files over cloud servers, auditing all these copies to verify their completeness and correctness, and providing a strong evidence to the data owner that he is getting what he is actually paying for. Our design goals can be summarized in the following points:

1. Designing Efficient Multi-Copy Provable Data Possession (EMC-PDP) protocols. These protocols should *efficiently* and *securely* provide the owner with a strong evidence that the CSP is in reality possessing *all* data copies that are agreed upon and these copies are intact.
2. Allowing the authorized users of the data owner to seamlessly access the file copy received from the CSP.
3. Scaling down the storage overhead on both the server and the verifier sides.
4. Minimizing the computational complexity on both the server and the verifier sides.
5. Limiting the bandwidth required by the protocols during the auditing phase.
6. Enabling public verifiability where anyone — not necessarily the data owner — who knows the owner's public key can challenge the remote servers and verify that the CSP is still possessing the owner's files.
7. Supporting *blockless* verification where the verifier has to have the ability to verify the data integrity even though he neither possesses nor retrieves the file blocks from the server.
8. Allowing unbounded number of auditing rather than imposing a fixed limit on the number of interactions between the verifier and the CSP.
9. Facilitating *stateless* verification where the verifier is not needed to hold and upgrade state between audits. Maintaining such state is unmanageable in case of physical damage of the verifier's machine or if the auditing task is delegated to a TPA.
10. Enabling both *probabilistic* and *deterministic* guarantees. In probabilistic guarantee the verifier checks a random subset of stored file blocks with each challenge (spot checking), while the verifier checks all the stored file blocks in the deterministic guarantee.

Remark 1. We are considering economically-motivated CSPs that may attempt to use less storage than required by the 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. As a result, in our work we do not encode the data file before outsourcing. Such encoding, for example using erasure codes, is mainly to reconstruct the file from limited amount of damages. In our proposed schemes, multiple copies of the file are generated and stored on a number of servers; hence a server's copy can be reconstructed even from a complete loss using duplicated copies on other servers. More importantly, unlike erasure codes, duplicated files enable scalability — a vital issue in the Cloud Computing system. Also, 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.

Remark 2. In this work we focus on *sensitive* archival and *warehoused* data, which is essential in numerous different applications such as digital libraries and astronomical/medical/scientific/legal repositories. Such data are subject to infrequent change, so we treat them as static. Since we are dealing with sensitive data like patients' medical history or financial data, this data must be encrypted before outsourcing to cloud servers. In our schemes we utilize the BLS authenticators [20] to build a public verifiable model.

5 Multi-Copy Provable Data Possession (MC-PDP) schemes

Suppose that a CSP offers to store n copies of an owner's file on n *different* servers — to prevent simultaneous failure of all copies and to achieve the availability aspect — in exchange for pre-specified fees metered in GB/month. 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 naïve solution to this problem is to use any of the previous PDP schemes to separately challenge and verify the integrity of each copy on each server. This is certainly not a workable solution; cloud servers can conspire to convince the data owner that they store n copies of the file while indeed they only store 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 is actually storing the single copy. The CSP can use another trick to prove data availability by generating the file copies upon a verifier's challenge; however, there is no evidence that the actual copies are stored all the time. The main core of this cheating is that the n copies are identical making it trivial for the servers to deceive the owner. Therefore, one step towards the solution is to leave the control of the file copying operation in the owner's hand to create unique *distinguishable/differentiable* copies.

Before presenting our main protocols, we start with a warmup scheme — Basic Multi-Copy Provable Data Possession (BMC-PDP) scheme — which leads to severe computational, communication, and management overhead. We then present the Multiple-Replica Provable Data Possession (MR-PDP) scheme due to Curtmola et al. [15]. To the best of our knowledge, this is the only scheme in the literature that addressed the auditing task of multiple copies of a data file. Through extensive analysis, we will elaborate the various features and limitations of the MR-PDP model especially from the authorized users' side. Curtmola et al. [15] did not consider how the authorized users of the data owner can access the file copies from the cloud servers noting that the internal operations of the CSP are opaque. Moreover, we will demonstrate the efficiency of our protocols from the storage, computation, and communication aspects. We believe that the investigation of both BMC-PDP and MR-PDP models will lead us to our main schemes.

5.1 Basic Multi-Copy Provable Data Possession (BMC-PDP) scheme

As explained previously, the cloud servers can conspire to convince the data owner that they store n copies of the file while indeed they store fewer than n copies, and this is due to the fact that the n copies are identical. Thus, the BMC-PDP scheme tries to solve the problem by leaving the control of the file copying operation in the owner's hand to generate unique distinguishable/differentiable copies of the data file. To this end, the data owner creates n distinct copies by encrypting the file under n different keys keeping these keys secret from the CSP. Hence, the cloud servers could not conspire by using one copy to answer the challenges for another. This *natural* solution enables the verifier to separately challenge each copy on each server using any of the PDP schemes, and to ensure that the CSP is possessing not less than n copies. Although the BMC-PDP scheme is a workable solution, it is impractical and has the following critical drawbacks:

- The computation and communication complexities of the verification task grow linearly with the number of copies. Essentially, the BMC-PDP scheme is equivalent to applying any PDP schemes to n different files.
- *Key management* is a severe problem with the BMC-PDP scheme. Since the data file is encrypted under n different keys, the data owner has to keep these keys secret from the CSP and — at the same time — to share these n keys with *each* authorized user. Moreover, when the 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 the key used in the encryption to enable the authorized users to properly decrypt the received copy.

5.2 Multiple-Replica Provable Data Possession (MR-PDP) scheme

Curtmola et al. [15] proposed Multiple-Replica PDP (MR-PDP) scheme where a data owner can verify that several copies of a file are stored by a storage service provider. The MR-PDP scheme is an extension to the PDP models proposed by Ateniese et al. [6]. Curtmola et al. [15] proposed creating distinct replicas/copies of the data file by first *encrypting* the file then *masking* the encrypted version with some randomness generated from a Pseudo-Random Function (PRF). The MR-PDP scheme of creating and auditing n distinct copies is illustrated in figure 8.

Key limitations of the MR-PDP scheme of Curtmola et al. [15] are as follows:

- Since the MR-PDP scheme is an extension to the PDP models proposed by Ateniese et al. [6], it inherits all the limitations of these models identified earlier in the literature survey section: long tags (1024 bits to achieve 80-bit secuirty level), computation overhead on both the verifier and server side, ability of CSP to cheat by using blocks from different files if the data owner uses the same secret key (d, v) for all his files.
- A slightly modified version of the critical key management problem of the BMC-PDP scheme is another concern in the MR-PDP scheme. The authorized users have to know which copy has been specifically retrieved from the CSP to properly *unmask* it before decryption. Due to the opaqueness nature of the internal operations of the CSP, — the server on which a specific copy is exactly stored is unknown to cloud customers — the MR-PDP scheme does not address how the authorized users of the data owner can access the file copies from the cloud servers.
- The MR-PDP supports only private verifiability, where just the data owner (or a verifier with whom the original owner shares a secret key) can do the auditing task.
- Essentially, the MR-PDP is an extension to the E-PDP version proposed by Ateniese et al. [6], and thus it only proves possession of the sum of the challenged blocks not the blocks themselves. Moreover, the CSP can corrupt the data blocks and the summation is still valid. For the CSP to prove possession of the blocks, it should multiply each of the challenged blocks with a random value.

6 The Proposed Efficient Multi-Copy Provable Data Possession (EMC-PDP) schemes

To achieve the design goals outlined in section 4.2, we propose Efficient Multi-Copy Provable Data Possession (EMC-PDP) schemes utilizing the BLS Homomorphic Linear Authenticators (HLAs)[20]. In short, the HLAs enable a data owner to fingerprint each block b_i of a file F in such a way that for any challenge vector $C = \{c_1, c_2, \ldots, c_r\}$ the server can homomorphically construct a tag authenticating the value $\sum_{i=1}^{r} c_i \cdot b_i$. The direct adoption of the HLAs proposed in [20] is not suitable and leads to two main attacks. The first attack arises when the data owner delegates the auditing task to a TPA: by collecting enough number of linear combinations of the same blocks, the TPA can obtain the data blocks by solving a system of linear equations breaking the privacy of the owner's data. This attack can be prevented by encrypting the data file before outsourcing to the CSP, and thus the TPA cannot get any information about the collected blocks. The second attack arises if the file identifier is not included in the block tag and

Setup

- $N = pq$ is the RSA modulus (p & q are prime numbers)
- g is a generator of QR_N (QR_N is the set of quadratic residues modulo N)
- Public key $pk = (N, g, e)$, secret key $sk = (d, v, x)$, $v, x \in_R \mathbb{Z}_N$, and $ed \equiv 1 \bmod (p-1)(q-1)$
- π is a pseudo-random permutation, f_x is a pseudo-random function keyed with the secret key x, and H is a hash function
- File $F = \{b_1, b_2, \ldots, b_m\}$
- E_K is an encryption algorithm under a key K

Data Owner

- Obtains an encrypted file $\widetilde{F}$ by encrypting the original file F using the encryption key K: $\widetilde{F} = \{\tilde{b}_1, \tilde{b}_2, \ldots \tilde{b}_m\}$, where $\tilde{b}_j = E_K(b_j)$, $1 \le j \le m$.
- Uses the encrypted version of the file $\widetilde{F}$ to create a set of tags $\{T_j\}_{1 \le j \le m}$ for all copies to be used in the verification process: $T_j = (H(v||j) \cdot g^{\tilde{b}_j})^d \bmod N$
- Generates n distinct replicas $\{\widehat{F}_i\}_{1 \le i \le n}$, $\widehat{F}_i = \{\hat{b}_{i,1}, \hat{b}_{i,2}, \ldots, \hat{b}_{i,m}\}$, using random masking as follows.
  for $i = 1$ to n **do**
    for $j = 1$ to m **do**
      1. Computes a random value $r_{i,j} = f_x(i||j)$
      2. Computes the replica's block $\hat{b}_{i,j} = \tilde{b}_j + r_{i,j}$ (added as large integers in $\mathbb{Z}$)
- Data owner sends the ***specific*** replica $\widehat{F}_i$ to a ***specific*** server S_i, $1 \le i \le n$

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

To check the possession of the replica $\widehat{F}_z = \{\hat{b}_{z,1}, \hat{b}_{z,2}, \ldots \hat{b}_{z,m}\}$, the owner challenges the *specific* server S_z as follows:

Owner — Remote Server S_z

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

$\xrightarrow{c,\ k,\ g_s}$

2. Computes the challenged block indices: $j_u = \pi_k(u)_{1 \le u \le c}$
3. Computes $T = \prod_{u=1}^{c} T_{j_u} \bmod N$
4. Computes $\rho = g_s^{\sum_{u=1}^{c} \hat{b}_{z,j_u}} \bmod N$

$\xleftarrow{T,\ \rho}$

5. Computes $j_u = \pi_k(u)_{1 \le u \le c}$
6. Checks $\left(\frac{T^e}{\prod_{u=1}^{c} H(v||j_u)} \cdot g^{r_{chal}}\right)^s \stackrel{?}{=} \rho$,

where $r_{chal} = \sum_{u=1}^{c} r_{z,j_u}$ is the sum of the random values used to obtain the blocks in the replica $\widehat{F}_z$

Fig. 8. The MR-PDP protocol by Curtmola et al. [15].

the same owner's secret key is used with all the owner's files. This allows the CSP to cheat by using blocks from different files during verification.

Since the main core to design a multi-copy provable data possession model is to generate unique distinguishable/differentiable copies of the data file, we use a simple yet *efficient* way to generate these distinct copies. In our EMC-PDP models we resort to the *diffusion* property of any secure encryption scheme. Diffusion means that the output bits of the ciphertext should depend on the input bits of the plaintext in a very complex way. In an encryption scheme with strong diffusion property, if there is a change in one single bit of the plaintext, then the ciphertext should completely change in an unpredictable way [36]. Our methodology of generating distinct copies is not only efficient, but also successful in solving the authorized users problem of the MR-PDP scheme [15] to access the file copy received from the CSP. In our schemes, the authorized users need only to keep a single secret key — shared with the data owner — to decrypt the file copy; it is not necessarily to recognize the specific server from which the copy is received.

In our work, the data owner has a file F and the CSP offers to store n copies, $\{F_1, F_2, \ldots, F_n\}$, of the owner's file in exchange for pre-specified fees metered in GB/month. We provide two versions of our EMC-PDP scheme: Deterministic EMC-PDP (DEMC-PDP) and Probabilistic EMC-PDP (PEMC-PDP). In the DEMC-PDP version, the CSP has to access all the blocks of the data file, while in the PEMC-PDP we depend on spot checking by validating a random subset of the file blocks. It is a trade-off between the performance of the system and the strength of the guarantee provided by the CSP.

6.1 Preliminaries and Notations

- F is a data file to be outsourced, it is composed of a sequence of m blocks, i.e., $F = \{b_1, b_2, \ldots, b_m\}$, where each block $b_i \in \mathbb{Z}_p$ for some large prime p.
- $\pi_{key}(\cdot)$ is a Pseudo-Random Permutation(PRP): $key \times \{0,1\}^{\log(m)} \rightarrow \{0,1\}^{\log(m)}$
- $\psi_{key}(\cdot)$ is a Pseudo-Random Function (PRF): $key \times \{0,1\}^{*} \rightarrow \mathbb{Z}_p$
- **Bilinear Map/Pairing.** The bilinear map is one of the essential building blocks of our proposed schemes. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of prime order p. Let g_1 and g_2 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 [37]:
  1. *Bilinear*: $\hat{e}(u_1 u_2, v_1) = \hat{e}(u_1, v_1) \cdot \hat{e}(u_2, v_1)$, $\hat{e}(u_1, v_1 v_2) = \hat{e}(u_1, v_1) \cdot \hat{e}(u_1, v_2)$ $\forall\, u_1, u_2 \in \mathbb{G}_1$ and $v_1, v_2 \in \mathbb{G}_2$
  2. *Non-degenerate*: $\hat{e}(g_1, g_2) \neq 1$
  3. *Computable*: there exists an efficient algorithm for computing $\hat{e}$
  4. $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$ $\forall\, u \in \mathbb{G}_1, v \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$
- $H(\cdot)$ is a map-to-point hash function : $\{0,1\}^{*} \rightarrow \mathbb{G}_1$
- E_K is an encryption algorithm with strong *diffusion* property, e.g., AES

6.2 Deterministic EMC-PDP (DEMC-PDP) scheme

The DEMC-PDP consists of five polynomial time algorithms: KeyGen, CopyGen, TagGen, Proof, and Verify.

- $(pk, sk) \leftarrow \mathsf{KeyGen}()$. This algorithm is run by the data owner to generate public key pk and private key sk.
- $\mathbb{F} = \{F_i\}_{1 \leq i \leq n} \leftarrow \mathsf{CopyGen}(CN_i, F)_{1 \leq i \leq n}$. This algorithm is run by the data owner. It takes as input a copy number CN_i and a file F and generates *unique differentiable* copies $\mathbb{F} = \{F_i\}_{1 \leq i \leq n}$, where each copy F_i is an ordered collection of blocks $\{b_{ij}\}_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}}$.

– $\Phi \leftarrow \mathsf{TagGen}(sk, \mathbb{F})$. This algorithm is run by the data owner. It takes as input the private key sk and the unique file copies $\mathbb{F}$, and outputs the tags/authenticators set Φ, which is an ordered collection of tags/authenticators $\{\sigma_{ij}\}$ of the blocks $\{b_{ij}\}$ $(1 \leq i \leq n,\ 1 \leq j \leq m)$
– $\mathbb{P} \leftarrow \mathsf{Proof}(\mathbb{F}, \Phi)$. This algorithm is run by the CSP. It takes as input the file copies $\mathbb{F}$ and the tags set Φ, and returns a proof $\mathbb{P}$ which guarantees that the CSP is actually storing n copies and *all* these copies are intact
– $\{1, 0\} \leftarrow \mathsf{Verify}(pk, \mathbb{P})$. Since our schemes support public verifiability, this algorithm can be run by any verifier (data owner or any other auditor). It takes as input the public key pk and the proof $\mathbb{P}$ returned from the CSP, and outputs 1 if the integrity of *all* file copies is correctly verified or 0 otherwise.

DEMC-PDP Construction

The procedures of our DEMC-PDP protocol execution are as follows:

- *Key Generation.* Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map, u is a generator of $\mathbb{G}_1$, 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 public key $y = g^x \in \mathbb{G}_2$
- *Distinct Copies Generation.* The data owner runs the $\mathsf{CopyGen}$ algorithm to generate n differentiable copies. The $\mathsf{CopyGen}$ algorithm creates a unique copy F_i by concatenating a copy number i with the file F, then encrypting using an encryption scheme with strong diffusion property like AES, i.e., $F_i = E_K(i||F)_{1 \leq i \leq n}$. The authorized users need only to keep a single secret key K to decrypt the file copy in later time.
- *Tag Generation.* Given the distinct file copies $\mathbb{F} = \{F_i\}_{1 \leq i \leq n}$, where each copy $F_i = \{b_{i1}, b_{i2}, \ldots, b_{im}\}$, the data owner runs the TagGen algorithm to create a tag σ_{ij} for each block b_{ij} $(1 \leq i \leq n, 1 \leq j \leq m)$ as $\sigma_{ij} = (H(F_{id}).u^{b_{ij}})^x \in \mathbb{G}_1$, where F_{id} is a unique file identifier for each owner's file. The F_{id} is constructed as $Filename||n||m||u$, i.e., the F_{id} is a unique fingerprint for each file comprising the file name, the number of copies for this file, the number of blocks in the file, and the generator u. Embedding the F_{id} into the block tag prevents the CSP from cheating using blocks from different files as the case in the MR-PDP [15]. Denote the set of tags $\Phi = \{\sigma_{ij}\}$ $(1 \leq i \leq n,\ 1 \leq j \leq m)$. The data owner sends $\{\mathbb{F}, \Phi, F_{id}\}$ to the CSP and deletes the copies and the tags from its local storage.
- *Challenge.* The verifier can check the integrity of *all* outsourced copies by challenging the CSP to ensure that the CSP is storing not less than n copies and these copies are not corrupted. At each challenge, the verifier sends a fresh PRF (ψ) key k to the CSP. *Both* the verifier and the CSP use the PRF(ψ) keyed with k to generate $n \times m$ random values $\{r_{ij}\} = \psi_k(l)_{1 \leq l \leq n \times m}$ $(1 \leq i \leq n,\ 1 \leq j \leq m)$
- *Response.* After generating the set of random values $\{r_{ij}\}(1 \leq i \leq n,\ 1 \leq j \leq m)$, the CSP runs the Proof algorithm to generate an evidence that it is still correctly possessing the n copies. The CSP responds with a proof $\mathbb{P} = \{\sigma, \mu\}$, where

$$\sigma = \prod_{i=1}^{n} \prod_{j=1}^{m} \sigma_{ij}^{r_{ij}} \in \mathbb{G}_1 \qquad \mu = \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij} \cdot b_{ij} \in \mathbb{Z}_p$$

- *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:

$$\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(H(F_{id})^R \cdot u^{\mu}, y), \qquad R = \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij} \tag{1}$$

If the verification equation passed, the Verify algorithm returns 1, otherwise 0.

The correctness of the above verification equation can be illustrated as follows:

$$\begin{aligned}
\hat{e}(\sigma, g) &= \hat{e}(\prod_{i=1}^{n}\prod_{j=1}^{m}\sigma_{ij}^{r_{ij}}, g) \\
&= \hat{e}(\prod_{i=1}^{n}\prod_{j=1}^{m}[H(F_{id})\cdot u^{b_{ij}}]^{x\cdot r_{ij}}, g) \\
&= \hat{e}(\prod_{i=1}^{n}\prod_{j=1}^{m}[H(F_{id})]^{r_{ij}}\cdot\prod_{i=1}^{n}\prod_{j=1}^{m}u^{r_{ij}\cdot b_{ij}}, y) \\
&= \hat{e}(H(F_{id})^{\sum_{i=1}^{n}\sum_{j=1}^{m}r_{ij}}\cdot u^{\sum_{i=1}^{n}\sum_{j=1}^{m}r_{ij}\cdot b_{ij}}, y) \\
&= \hat{e}(H(F_{id})^{R}\cdot u^{\mu}, y)
\end{aligned}$$

Remark 3. It is important to divide the file F into blocks in the proposed DEMC-PDP. If the file F is treated as one chunk and the data owner generates one tag $\sigma_{F_i} = (H(F_{id}).u^{F_i})^x \in \mathbb{G}_1$ for each file copy F_i $(1 \leq i \leq n)$, then the CSP can simply cheat the verifier as follows. To challenge the CSP in this situation, both the verifier and the CSP generate n random values $\{r_{F_i}\} = \psi_k(l)_{1\leq l\leq n}$. Then the CSP responds by computing $\sigma = \prod_{i=1}^{n}\sigma_{F_i}^{r_{F_i}} \in \mathbb{G}_1$, and $\mu = \sum_{i=1}^{n} r_{F_i}\cdot F_i \in \mathbb{Z}_p$. But this scenario will allow the CSP to cheat by storing only F *mod* p and not the entire file (file size is much large than p). Therefore, we divid the file F into blocks as a countermeasure against this cheating. The DEMC-PDP scheme is presented in figure 9.

6.3 Probabilistic EMC-PDP (PEMC-PDP) scheme

As we have previously demonstrated, the PEMC-PDP depends on spot checking by validating a random subset of the file blocks instead of validating all the blocks to achieve better performance in the computation overhead. In the PEMC-PDP scheme, we use the same indices for the challenged blocks across all copies. The rationale behind the PEMC-PDP scheme is that checking part of the file is much easier than the whole of it, and thus reducing the computation and storage overhead on the servers side. The PEMC-PDP scheme also consists of five algorithms: KeyGen, CopyGen, TagGen, Proof, and Verify.

PEMC-PDP Construction

The procedures of our PEMC-PDP protocol execution are as follows:

- *Key Generation*. The same as in the DEMC-PDP.
- *Distinct Copies Generation*. The same as in the DEMC-PDP.
- *Tag Generation*. Since the PEMC-PDP checks only a random subset of the file blocks, the block index is needed to be embedded into the block tag to prevent the CSP from cheating by using blocks at different indices. Given the distinct file copies $\mathbb{F} = \{F_i\}_{1\leq i\leq n}$, where each copy $F_i = \{b_{i1}, b_{i2}, \ldots, b_{im}\}$, the data owner runs the TagGen algorithm to create a tag σ_{ij} for each block b_{ij} $(1 \leq i \leq n, i \leq j \leq m)$ as $\sigma_{ij} = (H(F_{id}||j).u^{b_{ij}})^x \in \mathbb{G}_1$, where

Setup

- $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map, u and g are two generators for $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, $x \in \mathbb{Z}_p$ is a private key, and $y = g^x \in \mathbb{G}_2$ is a public key.
- File $F = \{b_1, b_2, \ldots, b_m\}$, where each block $b_i \in \mathbb{Z}_p$

Data Owner

- Creates distinct file copies $\mathbb{F} = \{F_i\}_{1 \le i \le n}$, where $F_i = E_K(i||F)_{1 \le i \le n}$. Each copy F_i is an ordered collection of blocks $\{b_{ij}\}_{1 \le j \le m}$
- Calculates the block tags $\Phi = \{\sigma_{ij}\}_{\substack{1 \le i \le n \\ 1 \le j \le m}}$, where $\sigma_{ij} = (H(F_{id}).u^{b_{ij}})^x \in \mathbb{G}_1$
- Sends $\{\mathbb{F}, \Phi, F_{id}\}$ to the CSP and deletes the copies and the tags from its local storage.

Challenge Response

Verifier CSP

1. Picks a fresh key k
2. Generates $n \times m$ random values $\{r_{ij}\}_{\substack{1 \le i \le n \\ 1 \le j \le m}} = \psi_k(l)_{1 \le l \le n \times m}$

$\xrightarrow{\quad k \quad}$

3. Generates $n \times m$ random values as the verifier did
4. Computes $\sigma = \prod_{i=1}^{n} \prod_{j=1}^{m} \sigma_{ij}^{r_{ij}} \in \mathbb{G}_1$
5. Computes $\mu = \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij} \cdot b_{ij} \in \mathbb{Z}_p$

$\xleftarrow{\quad \sigma,\ \mu \quad}$

6. Checks $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(H(F_{id})^R \cdot u^{\mu}, y)$, where $R = \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij}$

Fig. 9. The proposed DEMC-PDP scheme.

$F_{id} = Filename||n||m||u$ is a unique file identifier for each owner's file to prevent the CSP from cheating using blocks from different files as the case in the MR-PDP [15]. The data owner then generates an aggregated tag σ_j for the blocks at the same indices in each copy F_i as $\sigma_j = \prod_{i=1}^{n} \sigma_{ij} \in \mathbb{G}_1$. Denote the set of aggregated tags $\Phi = \{\sigma_j\}_{1 \le j \le m}$. The data owner sends $\{\mathbb{F}, \Phi, F_{id}\}$ 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 the PRP(π) keyed with k_1 and the PRF(ψ) keyed with k_2 to generate a set $Q = \{(j, r_j)\}$ of pairs of random indices and random values, where $\{j\} = \pi_{k_1}(l)_{1 \le l \le c}$ and $\{r_j\} = \psi_{k_2}(l)_{1 \le l \le c}$
- *Response.* After generating the set $Q = \{(j, r_j)\}$ of random indices and values, the CSP runs the Proof algorithm to generate an evidence that it is still correctly possessing the n copies. The CSP responds with a proof $\mathbb{P} = \{\sigma, \mu\}$, where

$$\sigma = \prod_{(j,r_j) \in Q} \sigma_j^{r_j} \in \mathbb{G}_1 \ , \qquad \mu_i = \sum_{(j,r_j) \in Q} r_j \cdot b_{ij} \in \mathbb{Z}_p, \qquad \text{and } \mu = \{\mu_i\}_{1 \le i \le n}$$

- *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:

$$\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}([\prod_{(j,r_j) \in Q} H(F_{id}||j)^{r_j}]^n \cdot u^{\xi}, y), \text{ where } \xi = \sum_{i=1}^{n} \mu_i \tag{2}$$

If the verification equation passed, the Verify algorithm returns 1, otherwise 0.

The correctness of the above verification equation can be illustrated as follows:

$$\begin{aligned}
\hat{e}(\sigma, g) &= \hat{e}(\prod_{(j,r_j) \in Q} \sigma_j^{r_j}, g) \\
&= \hat{e}(\prod_{(j,r_j) \in Q} \prod_{i=1}^{n} \sigma_{ij}^{r_j}, g) \\
&= \hat{e}(\prod_{i=1}^{n} \prod_{(j,r_j) \in Q} [H(F_{id}||j).u^{b_{ij}}]^{x \cdot r_j}, g) \\
&= \hat{e}(\prod_{i=1}^{n} \prod_{(j,r_j) \in Q} [H(F_{id}||j)]^{r_j} \cdot \prod_{i=1}^{n} \prod_{(j,r_j) \in Q} u^{r_j \cdot b_{ij}}, y) \\
&= \hat{e}([\prod_{(j,r_j) \in Q} H(F_{id}||j)^{r_j}]^n \cdot \prod_{i=1}^{n} u^{\sum_{(j,r_j) \in Q} r_j \cdot b_{ij}}, y) \\
&= \hat{e}([\prod_{(j,r_j) \in Q} H(F_{id}||j)^{r_j}]^n \cdot u^{\sum_{i=1}^{n} \mu_i}, y) \\
&= \hat{e}([\prod_{(j,r_j) \in Q} H(F_{id}||j)^{r_j}]^n \cdot u^{\xi}, y)
\end{aligned}$$

A question may arise, why have we not let the CSP to compute and send $\xi = \sum_{i=1}^{n} \mu_i$, and thus minimizing the communication overhead? If we allow the CSP to compute ξ , then the CSP can simply cheat the verifier as follows:

$$\xi = \sum_{i=1}^{n} \mu_i = \sum_{i=1}^{n} \sum_{(j,r_j)\in Q} r_j \cdot b_{ij} = \sum_{(j,r_j)\in Q} r_j \cdot \sum_{i=1}^{n} b_{ij}$$

Thus the CSP can just keep the blocks summation $\sum_{i=1}^{n} b_{ij}$ not the blocks themselves. Moreover, the CSP can corrupt the data blocks and the summation is still valid. Therefore, we force the CSP to send $\mu = \{\mu_i\}_{1\leq i\leq n}$ and the value ξ is computed at the verifier side. The PEMC-PDP scheme is presented in figure 10.

Remark 4. A slightly modified version of the proposed PEMC-PDP scheme can reduce the communication overhead during the response phase by allowing the CSP to compute and send only one μ instead of $\mu = \{\mu_i\}_{1\leq i\leq n}$. In this slight modification the block tags $\Phi = \{\sigma_{ij}\}_{\substack{1\leq i\leq n \\ 1\leq j\leq m}}$ will *not* be aggregated into a set $\{\sigma_j\}_{1\leq j\leq m}$, and the data owner will send $\{\mathbb{F}, \Phi, F_{id}\}$ to the CSP. During the challenge phase, both the verifier and the CSP generate two sets Q_1 and Q_2 of c random indices and $c \times n$ random values, respectively. The set $Q_1 = \{j\} = \pi_{k_1}(l)_{1\leq l\leq c}$, and the set $Q_2 = \{r_{j_i}\}_{j\in Q_1,\, 1\leq i\leq n}$ which is constructed as follows: *foreach* $j \in Q_1$ generate n random values $\{r_{j_i}\}_{1\leq i\leq n} = \psi_{k_2}(l)_{1\leq l\leq c\times n}$. Then the CSP responds by computing σ and μ, where $\sigma = \prod_{j\in Q_1} \prod_{i=1}^{n} \sigma_{ij}^{r_{j_i}} \in \mathbb{G}_1$, $\mu = \sum_{j\in Q_1} \sum_{i=1}^{n} r_{j_i} \cdot b_{ij} \in \mathbb{Z}_p$, and $r_{j_i} \in Q_2$. The verification equation (2) will be modified to $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(\prod_{j\in Q_1} H(F_{id}||j)^{\sum_{i=1}^{n} r_{j_i}} \cdot u^{\mu}, y)$, $r_{j_i} \in Q_2$. This modification will lower the communication overhead by sending only one μ, and thus reducing the communication cost by a factor of n. Since the challenge-response phase can occure many times during the life time of the data file, such reduced communication overhead feature may be useful for resource constrained applications. On the other hand, this reduced communication will be at the cost of the computations done on the CSP side; the number of cryptographic operations to compute σ will be cn exponentiations and cn multiplications versus c exponentiations and c multiplications in the original PEMC-PDP scheme. Therefore, it is a trade off between the computation and communication cost.

7 Security Analysis of the Proposed EMC-PDP schemes

In this section we present a formal security analysis for our proposed EMC-PDP schemes depending on the hardness of the Computational Diffie-Hellman (CDH) problem and the Discrete Logarithm (DL) problem.

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 $h = g^x$.

7.1 DEMC-PDP Security Analysis

We present a proof showing that the CSP can never give a forged response back to the verifier, i.e., the output of the verification equation (1) will always be 0 except when the CSP honestly compute $\sigma,\ \mu$.

Theorem 1. If the CDH problem is hard in bilinear groups, then there is no CSP that can pass the verification equation (1) of our DEMC-PDP scheme except by responding with correctly computed proof $\mathbb{P} = \{\sigma,\ \mu\}$.

Setup

- $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map, u and g are two generators for $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, $x \in \mathbb{Z}_p$ is a private key, and $y = g^x \in \mathbb{G}_2$ is a public key.
- File $F = \{b_1, b_2, \ldots, b_m\}$, where each block $b_i \in \mathbb{Z}_p$

Data Owner

- Creates distinct file copies $\mathbb{F} = \{F_i\}_{1 \leq i \leq n}$, where $F_i = E_K(i||F)_{1 \leq i \leq n}$. Each copy F_i is an ordered collection of blocks $\{b_{ij}\}_{1 \leq j \leq m}$
- Calculates the block tag $\sigma_{ij} = (H(F_{id}||j).u^{b_{ij}})^x \in \mathbb{G}_1$
- Computes a set of aggregated tags $\Phi = \{\sigma_j\}_{1 \leq j \leq m}$ for the blocks at the same indices in each copy F_i, where $\sigma_j = \prod_{i=1}^{n} \sigma_{ij} \in \mathbb{G}_1$
- Sends $\{\mathbb{F}, \Phi, F_{id}\}$ 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 $\{\mu_i\}_{1 \leq i \leq n} = \sum_{(j,r_j) \in Q} r_j \cdot b_{ij} \in \mathbb{Z}_p$

$\xleftarrow{\sigma,\ \mu=\{\mu_i\}_{1 \leq i \leq n}}$

6. Checks $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}([\prod_{(j,r_j) \in Q} H(F_{id}||j)^{r_j}]^n \cdot u^{\xi}, y)$,

where $\xi = \sum_{i=1}^{n} \mu_i$

Fig. 10. The proposed PEMC-PDP scheme.

Proof. First, since the F_{id} is embedded into the block tag, it is infeasible for the CSP to use blocks from different files and to pass the auditing procedures even if the owner uses the same secret key x with all his files. Second, the goal of a malicious CSP is to generate a proof that is not correctly computed and to pass the verification process. Let $\mathbb{P}' = \{\sigma',\ \mu'\}$ be the malicious CSP's response. Let $\mathbb{P} = \{\sigma,\ \mu\}$ be the expected response from an honest CSP, where $\sigma = \prod_{i=1}^{n}\prod_{j=1}^{m}\sigma_{ij}^{r_{ij}}$ and $\mu = \sum_{i=1}^{n}\sum_{j=1}^{m} r_{ij} \cdot b_{ij}$. If the malicious CSP's response $\mathbb{P}' = \{\sigma',\ \mu'\}$ passes the verification equation (1), then we can find a solution to the CDH problem. According to the correctness of our DEMC-PDP scheme, the expected proof $\mathbb{P} = \{\sigma,\ \mu\}$ satisfies the verification equation, i.e.,

$$\hat{e}(\sigma, g) = \hat{e}(H(F_{id})^R \cdot u^{\mu}, y).$$

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

$$\hat{e}(\sigma', g) = \hat{e}(H(F_{id})^R \cdot u^{\mu'}, y).$$

It is clear that $\mu' \neq \mu$, otherwise $\sigma' = \sigma$ which contradicts our assumption. Define $\Delta\mu = \mu' - \mu \neq 0$. Dividing the verification equation for the malicious response by the verification equation for the expected response, we obtain

$$\begin{aligned}
\hat{e}(\sigma' \cdot \sigma^{-1}, g) &= \hat{e}(u^{\triangle\mu}, y) \\
\hat{e}(\sigma' \cdot \sigma^{-1}, g) &= \hat{e}(u^{x\triangle\mu}, g) \\
\sigma' \cdot \sigma^{-1} &= u^{x\triangle\mu} \\
u^x &= (\sigma' \cdot \sigma^{-1})^{\frac{1}{\triangle\mu}}
\end{aligned}$$

Set $u = h$. $\quad \therefore h^x = (\sigma' \cdot \sigma^{-1})^{\frac{1}{\triangle\mu}}$

Hence, we have found a solution to the CDH problem unless evaluating the exponent causes a division by zero, but we have $\Delta\mu \neq 0$. Therefore, if $\sigma' \neq \sigma$ we can use the malicious CSP to break the CDH problem, and thus we guarantee that σ' must be equal to σ. It is only the values μ and μ' that can differ. Assume that the malicious CSP responds with $\sigma' = \sigma$ and $\mu' \neq \mu$. Now we have

$$\begin{gathered}
\hat{e}(\sigma, g) = \hat{e}(H(F_{id})^R \cdot u^{\mu}, y) = \hat{e}(\sigma', g) = \hat{e}(H(F_{id})^R \cdot u^{\mu'}, y) \\
\hat{e}(H(F_{id})^R \cdot u^{\mu}, y) = \hat{e}(H(F_{id})^R \cdot u^{\mu'}, y) \\
1 = \hat{e}(u^{\Delta\mu}, y) \\
1 = u^{\Delta\mu} \Longrightarrow \Delta\mu = 0 \Longrightarrow \mu' = \mu \qquad \square
\end{gathered}$$

7.2 PEMC-PDP Security Analysis

The essence of the PEMC-PDP security analysis is similar to that of the DEMC-PDP, but here we depend on the hardness of both the CDH problem the DL problem.

Theorem 2. If both the CDH problem and the DL problem are hard in bilinear groups, then there is no CSP that can pass the verification equation (2) of our PEMC-PDP scheme except by responding with correctly computed proof $\mathbb{P} = \{\sigma,\ \mu\}$, where $\mu = \{\mu_i\}_{1 \leq i \leq n}$.

Proof. First, since the F_{id} and the block index are embedded into the block tag, it is infeasible for the CSP to pass the auditing procedures using blocks from different files or using blocks at different indices even if the owner uses the same secret key x with all his files. Second, the goal of a malicious CSP is to generate a proof that is not correctly computed and to pass the

verification process. Let $\mathbb{P}' = \{\sigma',\ \mu'\}$, where $\mu' = \{\mu_i'\}_{1\leq i\leq n}$ be the malicious CSP's response. Let $\mathbb{P} = \{\sigma,\ \mu\}$ be the expected response from an honest CSP, where $\sigma = \prod_{(j,r_j)\in Q} \sigma_j^{r_j}$, $\mu = \{\mu_i\}_{1\leq i\leq n}$, and $\mu_i = \sum_{(j,r_j)\in Q} r_j \cdot b_{ij}$. If the malicious CSP's response $\mathbb{P}' = \{\sigma',\ \mu'\}$ passes the verification equation (2), then we can find a solution to both the CDH problem and the DL problem. According to the correctness of our PEMC-PDP scheme, the expected proof $\mathbb{P} = \{\sigma,\ \mu\}$ satisfies the verification equation, i.e.,

$$\hat{e}(\sigma, g) = \hat{e}([\prod_{(j,r_j)\in Q} H(F_{id}||j)^{r_j}]^n \cdot u^{\xi}, y),\ \xi = \textstyle\sum_{i=1}^{n} \mu_i$$

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

$$\hat{e}(\sigma', g) = \hat{e}([\prod_{(j,r_j)\in Q} H(F_{id}||j)^{r_j}]^n \cdot u^{\xi'}, y),\ \xi' = \textstyle\sum_{i=1}^{n} \mu_i'$$

Obviously if $\mu_i' = \mu_i\ \forall i$, it follows from the verification equation that $\sigma' = \sigma$ which contradicts our assumption. Define $\Delta\xi = \xi' - \xi = \sum_{i=1}^{n} \Delta\mu_i$, it must be the case that at least one of $\{\Delta\mu_i\}_{1\leq i\leq n}$ is nonzero. Dividing the verification equation for the malicious response by the verification equation for the expected response, we obtain

$$\begin{aligned} e(\sigma' \cdot \sigma^{-1}, g) &= e(u^{\triangle\xi}, y) \\ e(\sigma' \cdot \sigma^{-1}, g) &= e(u^{x\triangle\xi}, g) \\ \sigma' \cdot \sigma^{-1} &= u^{x\triangle\xi} \end{aligned}$$

set $u = g^{\alpha} h^{\beta}$ for $\alpha, \beta \in \mathbb{Z}_p$ then

$$\begin{aligned} (g^{\alpha} h^{\beta})^{x\triangle\xi} &= \sigma' \cdot \sigma^{-1} \\ h^{\beta x\triangle\xi} &= \sigma' \cdot \sigma^{-1} \cdot y^{-\alpha\triangle\xi} \\ h^{x} &= (\sigma' \cdot \sigma^{-1} \cdot y^{-\alpha\triangle\xi})^{\frac{1}{\beta\triangle\xi}} \end{aligned}$$

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 $\{\Delta\mu_i\}_{1\leq i\leq n}$ can be zero and the probability that $\beta = 0$ is $1/p$ which is negligible. Therefore, if $\sigma' \neq \sigma$ we can use the malicious CSP to break the CDH problem, and thus we guarantee that σ' must be equal to σ. It is only the values $\mu' = \{\mu_i'\}$ and $\mu = \{\mu_i\}$ $(1 \leq i \leq n)$ that can differ. Assume the malicious CSP responds with $\sigma' = \sigma,\ \mu' \neq \mu$. Now we have

$$\hat{e}(\sigma, g) = \hat{e}([\prod_{(j,r_j)\in Q} H(F_{id}||j)^{r_j}]^n \cdot u^{\xi}, y) = \hat{e}(\sigma', g) = \hat{e}([\prod_{(j,r_j)\in Q} H(F_{id}||j)^{r_j}]^n \cdot u^{\xi'}, y)$$

From which we conclude that

$$\begin{aligned} \hat{e}([\prod_{(j,r_j)\in Q} H(F_{id}||j)^{r_j}]^n \cdot u^{\xi}, y) &= \hat{e}([\prod_{(j,r_j)\in Q} H(F_{id}||j)^{r_j}]^n \cdot u^{\xi'}, y) \\ 1 &= \hat{e}(u^{\triangle\xi}, y) \\ 1 &= u^{\triangle\xi} \\ 1 &= (g^{\alpha} h^{\beta})^{\triangle\xi} \\ h^{\beta\triangle\xi} &= g^{-\alpha\triangle\xi} \\ h &= g^{\frac{-\alpha\triangle\xi}{\beta\triangle\xi}} \end{aligned}$$

Hence, we have found a solution to the DL problem unless evaluating the exponent causes a division by zero. However, we noted that not all of $\{\Delta\mu_i\}_{1\leq i\leq n}$ can be zero and the probability that $\beta = 0$ is $1/p$ which is negligible. Therefore, if there is at least one difference between $\{\mu'_i\}$ and $\{\mu_i\}$ $(1 \leq i \leq n)$, we can use the malicious CSP to break the DL problem, and thus we guarantee that $\{\mu'_i\}$ must be equal to $\{\mu_i\}$ $\forall i$ $\square$

8 Performance Analysis and Experimental Results

8.1 Performance Analysis

In this section we evaluate the performance of our proposed EMC-PDP schemes and the MR-PDP scheme proposed in [15]. The computation cost of our schemes and the scheme in [15] is estimated in terms of the cryptographic operations notated in table 2.

$\mathcal{H}$	Hashing into group $\mathbb{G}$
$\mathcal{A}$	Addition in group $\mathbb{G}$
$\mathcal{M}$	Multiplication in group $\mathbb{G}$
$\mathcal{D}$	Division in group $\mathbb{G}$
$\mathcal{E}$	Exponentiation in group $\mathbb{G}$
$\mathcal{P}$	Pairing in group $\mathbb{G}$

Table 2. Notation of cryptographic operations

Without loss of generality, assume the desired security level is 80-bit then the elliptic curve group we work on has a 160-bit group order and the size of the RSA modulus N is 1024 bits. Let the key used with the PRP and the PRF be of size 128 bits. Table 3 presents a comparison between our proposed schemes (DEMC-PDP and PEMC-PDP) and the MR-PDP scheme [15]. The comparison is held from these perspectives: the storage and generation cost of the block tags, the communication cost for the challenge and response phase, and the computation cost at both the verifier and the CSP side.

To establish a fair comparison between our schemes and the MR-PDP scheme [15], we assume tiny modifications to the original protocol proposed in [15]. First, we assume that the indices of the challenged blocks are the same across all replicas (*this assumption is an optimization for the verification computations of the original MR-PDP [15]*). Second, for the CSP to prove the possession of the blocks, each one of the challenged blocks should be multiplied by a random value, i.e, modifying the MR-PDP scheme to be an extension to the S-PDP version not the E-PDP version due to Ateniese et al.[6]. The second modification guarantees that the CSP is possessing each of the challenged blocks not just only their sum. Let n, m denote the number of copies and the number of blocks per copy, respectively, and c denotes the number of challenged blocks in both the PEMC-PDP and the MR-PDP.

		DEMC-PDP	PEMC-PDP	MR-PDP[15]
Tag	Size	160 nm bits	160 m bits	1024 m bits
	Generation	$2nm\mathcal{E} + nm\mathcal{M} + nm\mathcal{H}$	$2nm\mathcal{E} + 2nm\mathcal{M} + nm\mathcal{H}$	$2m\mathcal{E} + m\mathcal{M} + m\mathcal{H}$
Communication Overhead	Challenge	128 bits	$256 + \log_2(c)$ bits	$1280 + \log_2(c)$ bits
	Response	320 bits	$160(n+1)$ bits	1024(n+1) bits†
Computation Overhead	Proof	$nm\mathcal{E} + 2nm\mathcal{M} + nm\mathcal{A}$	$c\mathcal{E} + c(n+1)\mathcal{M} + cn\mathcal{A}$	$(c+n)\mathcal{E} + c(n+1)\mathcal{M} + cn\mathcal{A}$
	Verify	$2\mathcal{P} + 2\mathcal{E} + 1\mathcal{M}$ $+nm\mathcal{A} + 1\mathcal{H}$	$2\mathcal{P} + (c+2)\mathcal{E} + (c+1)\mathcal{M}$ $+n\mathcal{A} + c\mathcal{H}$	$(2n+c+1)\mathcal{E} + (cn+c+n)\mathcal{M}$ $+cn\mathcal{A} + c\mathcal{H} + 1\mathcal{D}$

Table 3. Storage, communication, and computation costs for the three schemes. The symbols used in the comparison are defined in Table 2. † There is an optimization for this response to be $1024 + 160n$ bits using hashing.

The main contribution of [15] is the reduced computation of generating the block tags. This contribution 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 distinguishable copies. Unfortunately, this contribution resulted in many various limitations that have been explained earlier, and one of these critical resulting limitations is the inability of the authorized users to access the file copies for the opaqueness nature of the CSP. Moreover, this reduced computation is unlikely to have significant impact on the overall system performance because the task of generating file tags is done only once during the life time of the file which may be for tens of years.

On the other hand, our schemes are *complete* and much more efficient than the MR-PDP[15] from the following perspectives:

- By *Completeness*, we mean that our schemes handle all entities — data owners, CSPs, and authorized users — that comprise almost all practical applications, while the protocol in [15] missed an essential entity in the outsourced data storage system which is the authorized users.
- The storage overhead (total tags size) of our PEMC-PDP is 6 times less than that of the MR-PDP. In general our tag size is 1/6 the tag size of the scheme due to Curtmola et al. [15], and this mitigates the extra storage cost of our DEMC-PDP. For a 64-MB file with 4-KB blocks and 10 copies, the total tags sizes of the MR-PDP, DEMC-PDP, and PEMC-PDP are 2 MB, 3.125 MB, and 0.3125 MB, respectively.
- The communication overhead of our schemes is much less than that of the MR-PDP; the way we construct our schemes enables us to aggregate the responses from the servers on which the copies are stored. For the challenge phase, the communication cost of our DEMC-PDP is 10 times less than that of the MR-PDP, and it is 5 times less for our PEMC-PDP. Moreover — for just only 5 copies — we compress the response of our DEMC-PDP to about 1/19 of the MR-PDP response, and the compression ratio between the response of our PEMC-PDP to that of the MR-PDP is about 1:6. Hence, our schemes are efficient and much more practical especially when the available bandwidth is limited.
- Our PEMC-PDP is the most efficient scheme from the computation cost perspective. The experimental results demonstrate the performance efficiency of our schemes even with the existence of the paring operations which consume more time than other cryptographic operations (in the verification phase we use just only two pairing operations).

8.2 Experimental Results

In this section, we present and discuss the experimental results of our research. The experiments are conducted using C++ on a system with an Intel(R) Xeon (R) 2-GHZ processor and 3 GB

RAM running Windows XP. In our implementation we use MIRACL library version 5.4.2 and 64-MB file. To achieve 80-bit security level, the elliptic curve group we work on has a 160-bit group order and the size of the RSA modulus N is 1024 bits. In our implementation we do not consider the time to access the file blocks, as the state-of-the-art- hard drive technology allows as much as 1 MB to be read in just few nanoseconds [10]. Hence, the total access time is unlikely to have substantial impact on the overall system performance.

We compare our schemes (DEMC-PDP and PEMC-PDP) with the MR-PDP scheme from different perspectives: total tags size, tags generation cost, communication cost that includes both challenge and response, CSP computation cost, and verifier computation cost. It has been reported in [6] 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 only needs 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.

Figure 11 shows the total tag size ($\mathcal{TS}$) for our proposed schemes and for the MR-PDP model using different number of copies. For the PEMC-PDP and MR-PDP schemes, the $\mathcal{TS}$ is independent of the number of copies, while $\mathcal{TS}$ is linear in the number of copies n for the DEMC-PDP. Our PEMC scheme has the lowest $\mathcal{TS}$; its $\mathcal{TS}$ is 6 times less than that of the MR-PDP. The way we aggregate the tags makes our PEMC-PDP scheme to have the lowest storage overhead on the server side. Our DEMC-PDP has the strongest guarantee that all blocks of all copies are actually being stored by the CSP and they are intact, but this strongest guarantee is at the expense of the storage overhead.

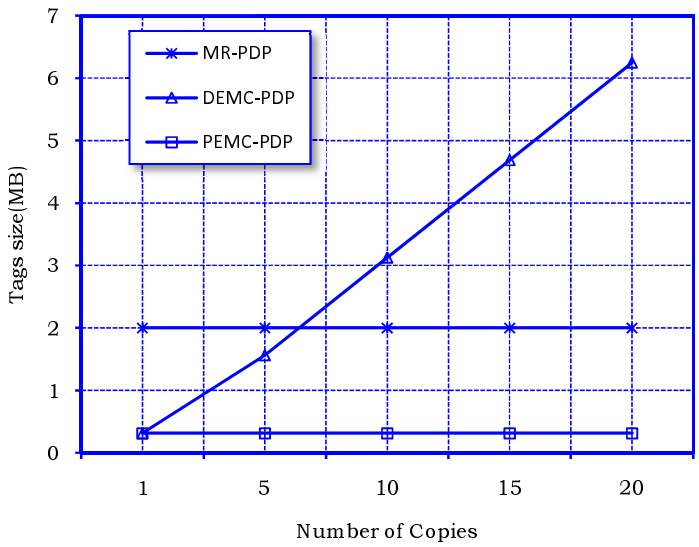

Fig. 11. $\mathcal{TS}$ (in MB) of our proposed schemes and the MR-PDP model

Figure 12 presents the tags generation time (in seconds) for our proposed schemes and for the MR-PDP model using different number of copies. The tags generation time of the MR-PDP is the lowest one, and this is because it generates a single set of tags for all copies. But as mentioned earlier, this reduced computation of tags generation resulted in precluding the authorized users

from seamlessly accessing the owner's files. Moreover, the tags generation time is unlikely to have significant impact on the overall system performance; tags generation task is done only once during the files life time which may be for tens of years. As noted from figure 12, the time of our PEMC-PDP scheme to generate tags is slightly higher than that of our DEMC-PDP model. This slight difference is due to the additional $nm\mathcal{M}$ operations that are used to aggregate the tags of the blocks at the same indices.

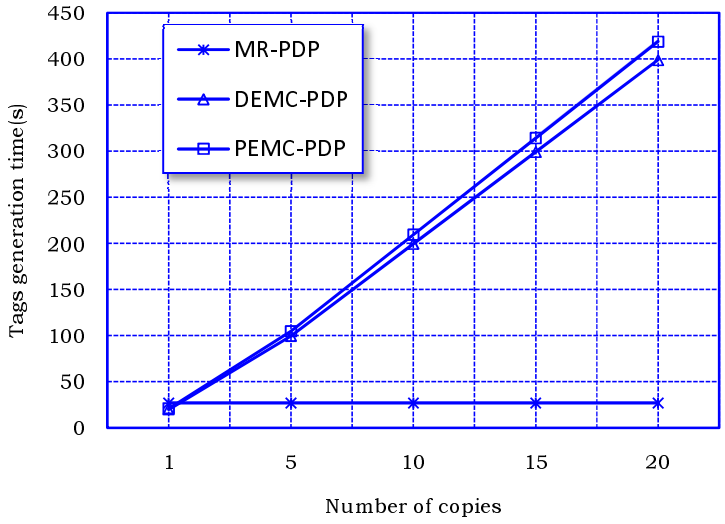

Fig. 12. Tags generation times (in sec) of our proposed schemes and the MR-PDP model

The communication cost of the three schemes is illustrated in figure 13. The MR-PDP scheme has the highest communication overhead. On the other hand, our DEMC-PDP has constant communication cost (56 bytes) independent of both the number of copies and the number of blocks per copy, and its communication cost is the lowest among the three schemes. As presented in figure 13, the communication cost of our PEMC-PDP is significantly less than that of the MR-PDP model. For 10 copies, the communication cost of our PEMC-PDP scheme is about 253 bytes compared to 1569 bytes for the MR-PDP (6 times less). Hence, our schemes are much more practical especially when the available bandwidth is limited and there are millions of verifiers who need to audit their data files over the CSPs.

Table 4 presents the CSP computation times (in ms) to provide an evidence that it is actually possessing the file copies in an uncorrupted state. Of course, the computation cost of our DEMC-PDP scheme is the largest one; it computes the proof using all the blocks of all copies, and thus provides the strongest guarantee. The *speed-up* $= \frac{Time_{\text{MR-PDP}} - Time_{\text{PEMC-PDP}}}{Time_{\text{MR-PDP}}} \times 100$ of our PEMC-PDP with respect to the MR-PDP [15] is also presented.

The proposed PEMC-PDP scheme outperforms the MR-PDP model; it reduces the server computation time up to 29% making our scheme much more efficient for practical applications where large number of verifiers are connected to the CSP causing a huge computation overhead over the servers.

Table 5 presents the verifier computation times (in ms) to check the responses received from the CSP. The proposed DEMC-PDP scheme has the shortest verification time among the three schemes; its verification process costs only two pairings, two exponentiations, and one multiplication (other operations can be neglected). In addition, the PEMC-PDP model reduces the

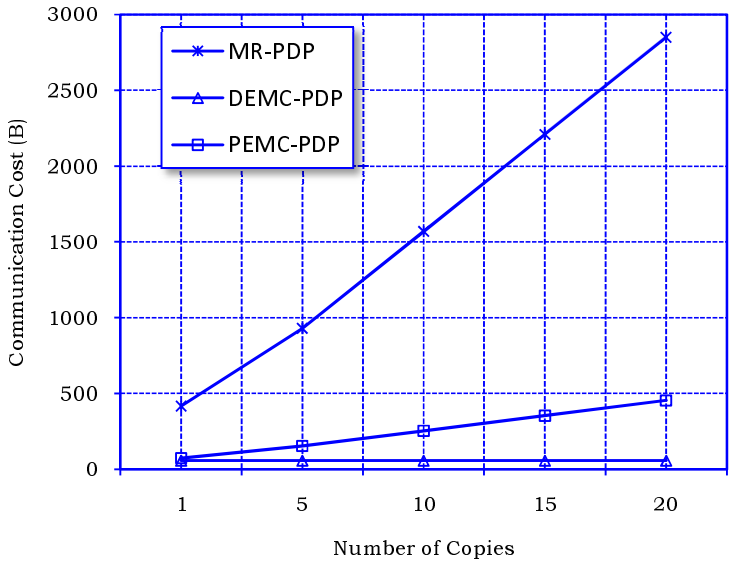

Fig. 13. Communication cost (in bytes) of our proposed schemes and the MR-PDP model

# of Copies	DEMC-PDP	PEMC-PDP	MR-PDP [15]	*Speed-up*
1	10485.76	294.4	394.57	25.39%
5	52428.8	323.76	440.13	26.44%
10	104857.6	360.43	497.08	27.49%
15	157286.4	395.9	554.03	28.54%
20	209715.2	430.19	610.98	29.59%

Table 4. CSP computation times (ms) of the three schemes for different number of copies

verifier computation cost up to 53% with respect to the MR-PDP scheme. As illustrated in table 5, both of our schemes have a very tiny increase in the verifier computation time with increasing number of copies, and this is due to the additional $nm\mathcal{A}$ and $n\mathcal{A}$ operations — which can be neglected — for the DEMC-PDP and the PEMC-PDP models, respectively. On the other hand, the verifier computation cost of the MR-PDP scheme has a linear relation between the expensive $\mathcal{E}$, $\mathcal{M}$ operations and the number of copies.

# of Copies	DEMC-PDP	PEMC-PDP	MR-PDP [15]	*Speed-up*
1	10.21	294.04	396.21	25.79%
5	10.27	294.042	445.11	33.94%
10	10.32	294.046	506.22	41.91%
15	10.37	294.049	567.34	48.17%
20	10.42	294.52	628.45	53.14%

Table 5. Verifier computation times (ms) of the three schemes for different number of copies

To sum up, the PEMC-PDP has the best overall performance among the three schemes considered here; it surpasses the MR-PDP model from a number of perspectives: storage overhead, communication cost, and computation cost. The only advantage of the MR-PDP scheme is the reduction of the tag generation cost which is done only once during the life time of the outsourced storage system, while it causes high overhead for other tasks that are done more frequently: communication, proof, and verify. Besides, if the computation cost over the server side is less important (the CSP has unlimited computational resources), the owner needs the strongest guarantee that all block of all copies are intact, and the verification process is done in a constrained environment with limited bandwidth and limited verifier's computational power (e.g., a PDA or a cell phone), then our DEMC-PDP scheme is the best choice to be applied in such circumstances.

9 Conclusion

In this work we have studied the problem of creating multiple copies of data file over the cloud servers and auditing all these copies to verify their correctness and completeness. We presented two *secure*, *complete*, and *efficient* protocols: DEMC-PDP and PEMC-PDP schemes that achieve our design goals outlined in section 4.2. We held an extensive comparative analysis between our schemes and the MR-PDP [15] which is the only scheme in the literature that previously addressed the same problem, and proved that our proposed schemes alleviate key limitations of the MR-PDP model that precludes the authorized users from seamlessly accessing the owner's files. Through extensive security analysis and experimental results, we have showed that our proposed schemes are provably secure and much more efficient than the MR-PDP to be applied in practical applications. To the best of our knowledge, the proposed protocols are the first secure, complete, and efficient protocols that address the storage integrity of multiple data copies over Cloud Computing. By using the proposed schemes, the data owners can have a strong evidence that they actually get the service they pay for.

References

1. P. Mell and T. Grance, "Draft NIST working definition of cloud computing," Online at http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, 2009.

2. R. O. Weichao Wang, Zhiwei Li 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. K. Zeng, "Publicly verifiable remote data integrity," in *ICICS*, 2008, pp. 419–434.
8. 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.
9. D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," Cryptology ePrint Archive, Report 2006/150, 2006.
10. 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.
11. 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.
12. 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.
13. M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008.
14. E. Mykletun, M. Narasimha, and G. Tsudik, "Authentication and integrity in outsourced databases," *Trans. Storage*, vol. 2, no. 2, 2006.
15. R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession," in *28th IEEE ICDCS*, 2008, pp. 411–420.
16. Amazon Simple Storage Service (Amazon S3), http://aws.amazon.com/s3/.
17. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 26, no. 1, 1983.
18. D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, London, UK, 2001, pp. 514–532.
19. F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin, "Dynamic authenticated index structures for outsourced databases," in *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2006, pp. 121–132.
20. H. Shacham and B. Waters, "Compact proofs of retrievability," Cryptology ePrint Archive, Report 2008/073, 2008, http://eprint.iacr.org/.
21. 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.
22. V. Shoup, "On the security of a practical identification scheme," in *EUROCRYPT'96: Proceedings of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*, Berlin, Heidelberg, 1996, pp. 344–353.
23. 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*, 2007, pp. 584–597.
24. T. S. J. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, Washington, DC, USA, 2006.

25. 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, 2008, pp. 63–68.
26. 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, 2009, pp. 43–54.
27. ——, "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.
28. 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, 2009, pp. 109–127.
29. G. Ateniese, R. Di 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.
30. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *CCS '09: Proceedings of the 16th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2009, pp. 213–222.
31. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *ESORICS'09: Proceedings of the 14th European Conference on Research in Computer Security*, Berlin, Heidelberg, 2009, pp. 355–370.
32. Z. H. G.-J. A. H. H. Yan Zhu, Huaixi Wang and S. S. Yau, "Cooperative provable data possession," Cryptology ePrint Archive, Report 2010/234, 2010.
33. Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/.
34. B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris, "Efficient replica maintenance for distributed storage systems," in *NSDI'06: Proceedings of the 3rd Conference on Networked Systems Design & Implementation*, Berkeley, CA, USA, 2006.
35. P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, and M. Baker, "The LOCKSS peer-to-peer digital preservation system," *ACM Trans. Comput. Syst.*, vol. 23, no. 1, pp. 2–50, 2005.
36. C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst.Tech. J.*, vol. 28, no. 4, 1949.
37. A. Menezes, "An introduction to pairing-based cryptography," Lecture Notes 2005, Online at http://www.math.uwaterloo.ca/~ajmeneze/publications/pairings.pdf.