Remote Integrity Verification for Multiple Data Copies in Cloud Environments: A Comparative Analysis and Open Research Issues

Ayad Barsoum
Computer Science Department, St. Mary's University, San Antonio, Texas, USA
Email: abarsoum@stmarytx.edu

Athanasios Vasilakos
Lulea University of Technology, Sweden
Email: vasilako@ath.fortnet.gr

Abstract—**Currently, we are living in a digital world where many organizations produce a large amount of sensitive data, which outpaces their storage ability. It is very expensive to manage such huge amount of data for that requires qualified personnel and high storage space. Therefore, many organizations prefer to outsource their data to remote cloud service providers (CSPs). To achieve a higher level of scalability, availability and durability, some clients ask the CSP to maintain multiple copies of their outsourced data on multiple data centers. In such a case, the clients will be charged more fees. Thus, the clients need a guarantee that the CSP keeps the data copies that they pay for. Moreover, all these copies should be consistent with the most recent modifications issued by the clients. The problem of provable data possession (PDP) for multiple data copies has been investigated by many researchers. In this paper, we review the concept of PDP and provide an extensive survey for different provable multi-copy data possession (PMDP) schemes. Moreover, the paper discusses the design principles for various PMDP constructions, highlights some limitations, and present a comparative analysis for numerous PMDP models. We classify PMDP schemes into protocols for *static* data, and models that support outsourcing of *dynamic* data. The paper also addresses the concept of proof of retrievability, which is a complementary approach to PDP. Furthermore, we highlight some open research issues that need to be investigated and tackled to achieve the wide acceptance and usage of outsourcing data storage.**

Index Terms— **Cloud computing, outsourcing data storage, multiple data copies, integrity verification**

I. INTRODUCTION

Cloud Computing (CC) can be perceived as a virtualized pool of computing resources. These resources include (but not limited to) storage space, computing power, memory, applications, services and network bandwidth. The resources will be provisioned and de-provisioned to customers according to their needs. According to [1], the CC services can be categorized into: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). The SaaS is the widely used model of CC services. These services provide customers with access to applications running on the cloud service provider's infrastructure. Publicly known examples of SaaS model are Google Docs, Google Calendar, and Zoho Writer. Through PaaS model, customers can deploy their applications on the CSP infrastructure given that these applications are created using tools supported by the provider. Using IaaS model, customers can rent and use the provider's resources. Therefore, the clients can install any applications including operating systems.

The CC paradigm provides a number of key advantages, which make it an attractive research area for both academia and industry. This model of information technology (IT) architecture enables customer to avoid capital expenditure on hardware, software and services by allowing them to share the resources with other clients and paying only for their actual usage. In addition, CC provides an immediate access to a broad range of applications and reduce the management overhead. Moreover, since a third party is responsible for storing data and running the cloud, the maintenance cost is highly reduced. Another key advantage of CC model is the ability to scale up and down IT capacity over time to business needs. Also, cloud computing provides more mobility where customers can access information from any part of the world. CC also enables organizations to outsource huge amount of data to remote sites, and they are no longer worried about constant server updates and other computing issues [2]

More and more customers and organizations are opting for outsourcing data to CSPs to alleviate the burden of local data storage and maintenance. In addition, outsourcing the data storage enables many authorized users to access the data from different geographic locations, which is more convenient for them. It has been indicated that IT outsourcing has grown by a staggering 79% as companies seek to reduce costs and focus on their core businesses [3].

However, once the data have been outsourced to the cloud, data owners are no longer physically possess their sensitive data, and that can lead to new challenges for data confidentiality and integrity protection in cloud computing. The owners can simply protect the confidentiality of their data by using encryption before outsourcing to remote servers. Thus, it is a crucial demand

of customers to have a strong evidence that the CPS still store the data intact (i.e., the data is not being tempered with or partially deleted over time)

It is clear that the integrity of outsourced data over the cloud is at risk due to different reasons. First, the CSP has an incentive to hide any data loss or corruption over cloud servers to maintain its reputation. Second, a CSP might not store all the data in fast storage (i.e., place it on CDs or other offline media) or even delete some of the data that is rarely accessed to reduce the used storage space and make more money. Third, the cloud infrastructures are subject to a 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 there are economic benefits for outsourcing data storage to CSP, there is no guarantee of data integrity (i.e., data completeness and correctness). This problem, if not properly handled, may hinder the successful deployment of cloud architecture.

Efficient verification of the integrity of outsourced data becomes a formidable challenge for the traditional techniques (based on cryptographic hashing and signature schemes) are not applicable. These traditional techniques require to have a local copy of the outsourced data, and it is impractical to download all the stored data to perform integrity validation. Thus, a crucial requirement for cloud customers is to have efficient techniques to verify the integrity of their outsourced data with minimum computation, communication, and storage overhead.

To achieve a higher level of scalability, availability and durability, some clients ask the CSP to maintain multiple copies of their outsourced data on multiple data centers. In such a case, the clients will be charged more fees. Therefore, the clients need a guarantee that the CSP keeps the data copies that they pay for. Moreover, all these copies should be consistent with the most recent modifications issued by the clients. The problem of provable data possession (PDP) for multiple data copies has been investigated by many researchers.

This paper reviews the basic concepts of PDP and provide an extensive survey for different provable multi-copy data possession (PMDP) schemes. In addition, the paper discusses the design principles for various PMDP constructions, highlights some limitations, and present a comparative analysis for numerous PMDP models. We classify the PMDP schemes according to the nature of the outsourced data: *static* data and *dynamic* data. The paper also highlights the concept of proof of retrievability, which is a complementary approach to PDP.

Main contributions. Our contributions can be summarized as follows:

- Provide an extensive survey for the domain of outsourcing replicated data to cloud servers, investigate the design principles for various PMDP schemes, and highlights some limitations
- Classify PMDP constructions according to the architectural design and the nature of outsourced data
- Present a comparative analysis for numerous PMDP models and discuss some open research issues

Paper organization. The remainder of the paper is organized as follow. The basic concepts of PDP and PMDP are presented in Section II. Section III provides different PMDP schemes. The comparative analysis is shown in Section IV. Section V highlights the concept of POR. Concluding remarks and open research issues are given in Section VI.

II. Provable Data Possession

In this section, we describe the basic concepts of PDP and PMDP, and provide the dimensions of our classification of PMDP schemes.

A. Basic Concepts

PDP is a technique for validating data integrity over remote servers. A PDP model has been formalized by Ateniese *et al*. [6]. According to that model, the data owner perform some pre-processing steps on the data file to generate some tags to be used later for verification purposes through a challenge response protocol with the remote server. Then the data owner outsource the file to be stored on a remote untrusted server (the owner may delete its local copy of the file). Later, the server prove that the data is in its possession by responding to challenges sent from a 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 [6-14]. These schemes focus on a single copy of the data file and provide no proof that the CSP stores multiple copies of the owner's file.

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, during the preparation of a research paper, we store multiple copies of the paper to be able to recover it in case of any failure or physical damage. Also, organizations, governments, and universities replicate their financial, personal, and general data to guarantee its availability and durability over time. In the CC model, customers rely on the CSP to perform 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. Also, cloud customers should be securely and efficiently convinced that the CSP is actually possessing *all* data copies that are agreed upon in the service contact. Moreover, these data copies should be complete and intact. In other words, customers need to make sure that they are getting the service they are paying for.

The number of data copies kept by the CSP depends on the nature of the data; more copies are needed for critical data that cannot easily be reproduced. Thus, the CSPs formulate their pricing model according to the replication strategy. For example, Amazon S3 standard storage strategy [15] 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 Amazon S3 prices are higher than that of the RRS. Unfortunately, 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. Clients 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. Thus, many researchers have devoted their work to propose and develop schemes that can guarantee data integrity for multiple copies over remote servers.

B. Our Classification

In this sub-section we classify PMDP schemes according to the nature of outsourced data. Some PMDP models focus on archived and warehoused data, which is essential in many applications such as digital libraries and astronomical/medical /scientific/legal repositories. Such data are subject to infrequent change, so they are treated as *static*. Other PMDP protocols handle *dynamic* behavior of data. Each PMDP scheme has its own design principles: some schemes are based on the encryption techniques, some are based the RSA model, some are based on bilinear pairing, some are based on hashing techniques, and others are based on authenticated data structures. Figure 1 outlines our classification for different PMDP models.

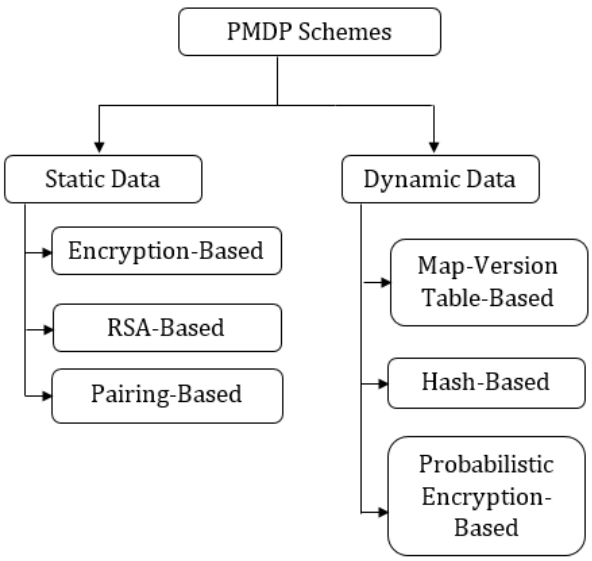


Figure 1. Classification of PMDP schemes.

III. PROVABLE MULTI-COPY DATA POSSESSION

For an increased level of scalability, availability and durability, some cloud customers may want their data to be replicated on multiple servers across multiple data centers. The more copies the CSP is asked to store, the more fees the customers are charged. Therefore, customers need to have a strong guarantee that the CSP is storing all data copies that are agreed upon in the service contract.

In this section, we address PMDP schemes for multiple copies of *static* data, and PMDP constructions for multiple copies of *dynamic* data. Generating unique differentiable copies of the data file is the core to design a provable multi-copy data possession scheme. Identical data copies enable the CSP to simply deceive the owner by storing only one copy and pretending that it stores multiple copies.

A. Provable Multi-Copy STATIC Data Possession

Encryption-Based Scheme

Barsoum et al. [16] presented a Basic Multi-Copy Provable Data Possession (BMC-PDP) scheme. The schemes allows the data owner to create n distinct copies by encrypting the file under n different keys and keeping these keys secret from the CSP. Hence, the cloud servers could not cheat by using one copy to answer the challenges for another. This scheme enables the verifier to separately challenge each copy on each server using any PDP protocol for a single data copy, 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 BMC-PDP scheme is equivalent to applying any PDP schemes to *n* different files. Thus, the computation and communication complexities of the verification process grow linearly with the number of data copies
- Key management is a critical problem with the BMC-PDP scheme. The data owner must keep the used *n* encryption keys secret from the CSP, and — at the same time — 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. The CSP uses a load balancing mechanism, which directs the authorized user's request to the server with the lowest congestion (that server may be different each time). Thus, each copy should contain some indicator about the key used in the encryption to enable the authorized users to properly decrypt the received copy.

RSA-Based Scheme.

The first multiple-replica provable data possession (MR-PDP) scheme was proposed by Curtmola *et al*. [17] to create multiple copies of an owner's file and audit them. The MR-PDP scheme increases data availability; a corrupted data copy can be reconstructed using duplicated copies on other servers. The single-copy PDP model of [6] is the basis for the MR-PDP scheme of [17]. According to the scheme of [17], distinct data copies are created by first *encrypting* the file using one key, then use different randomness generated from a pseudo-random function to mask the encrypted file copy (*n* times)

Initially, a file *F* is fragmented into *m* blocks $\{b_j\}_{1\le j\le m}$. The owner encrypts *F* using a key *K* to obtain an encrypted version $\tilde{F} = \{\tilde{b}_j\}_{1\le j\le m}$, where $\tilde{b}_j = E_K(b_j)$. The owner

generates n distinct copies $\{\widehat{F}_i\}_{1\le i\le n}$, where $\widehat{F}_i = \{\hat{b}_{ij}\}_{1\le j\le m}, \hat{b}_{ij} = \tilde{b}_j + r_{ij}$ (added as large integers in $\mathbb{Z}$), $r_{ij} = f_x(i||j)$, and f_x is a pseudo-random function keyed with a secret key x. Figure 2 gives a summary of the MR-PDP scheme.

Remark. The MR-PDP scheme [17] did not consider the interaction between authorized users (those who have the right to access the owner's file) and the CSP. If an authorized user sends a data access request to the CSP, the CSP retrieves one of the available copies. The authorized user has to know the copy index to properly *unmask* it before decryption.

Since the internal operations of the CSP is not known to clients, the authorized users cannot recognize which copy has been received. If the copy index i is attached with each copy forming the structure $(i||\widehat{F}_i)$, then corrupting (or even swapping) copy indices will block the accurate unmasking process. Thus, the authorized users are unable to access the data file.

Setup

– Generate $\acute{p}$ & $\acute{q}$ (prime numbers) and compute $N = \acute{p}\acute{q}$ (RSA modulus)

– $\acute{g}$ is a generator of QR_N (QR_N is the set of quadratic residues modulo N)

– public key $pk = (N, \acute{g}, e)$, secret key $sk = (d, v, x)$, $v, x \in_R \mathbb{Z}_N$, and $ed \equiv 1 \bmod (\acute{p}-1)(\acute{q}-1)$

– π_k is a pseudo-random permutation keyed with a key k, f_x is a pseudo-random function keyed with the secret key x, and H is a hash function ($H: \{0,1\}^* \longrightarrow QR_N$)

– File $F = \{b_j\}_{1\le j\le m}$, and E_K is an encryption algorithm under a key K

Data Owner

– Encrypts the data file F under the key K to obtain an encrypted version $\tilde{F} = \{\tilde{b}_j\}_{1\le j\le m}$, where $\tilde{b}_j = E_K(b_j)$.

– Uses the encrypted version $\tilde{F}$ to create a set of tags $\{T_j\}_{1\le j\le m}$ for all copies: $T_j = \left(H(v||j).\acute{g}^{\tilde{b}_j}\right)^d \bmod N$

– Generates n distinct copies $\{\widehat{F}_i\}_{1\le i\le n}$, $\widehat{F}_i = \{\hat{b}_{ij}\}_{1\le j\le m}$ utilizing random masking:

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

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

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

Owner — Remote Server S_z

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

$c, k, \acute{g}_s$ →

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

← T, ρ

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

Figure 2. The MR-PDP scheme by Curtmola *et al*. [17].

Private verifiability (i.e., only the data owner can check data integrity/possession) is only supported by the MR-PDP scheme. A key feature of remote data checking schemes is to support public verifiability to avoid disputes that may arise between the data owner and the CSP. Delegating the auditing process (without revealing secret keys) to a trusted third party for verifying the data integrity can resolve such disputes. For verification purposes in [17], portion of the set $\{r_{ij}\}$ is needed to be generated ($r_{chal} = \sum_{j\in A} r_{zj}$ in Figure 2). These random values cannot be publicly known, otherwise the CSP can derive the encrypted version $\tilde{F}$, and store only one copy. Hence, private verifiability is only supported by [17].

Pairing-Based Scheme

Barsoum *et al*. [18, 38] have studied the problem of verifying multi-copy of outsourced data file on remote servers. They proposed a pairing-based provable multi-copy data possession (PB-PMDP) scheme. Their scheme supports public verifiability and considers the communications between authorized users and cloud servers. The PB-PMDP scheme utilizes the diffusion property of any secure encryption scheme to generate distinct file copies. The data owner generates *n* distinct copies $\tilde{\mathbb{F}} = \{\tilde{F}_i\}_{i\le i\le n}$. The file copy $\tilde{F}_i$ is created by prepending a copy index *i* to the file *F*, then applying encryption under E_K, i.e., $\tilde{F}_i = E_K(i||F)$. The PB-PMDP scheme divide the file copy $\tilde{F}_i$ into *m* blocks and each block is fragmented into *s* sectors. Thus, $\tilde{F}_i = \{\tilde{b}_{ijk}\}$, where $i: 1 \to n, j: 1 \to m, k: 1 \to s$, and $\tilde{b}_{ijk} \in \mathbb{Z}_p$ for some large prime *p*. The PB-PMDP scheme [18] utilizes the BLS HLAs [19]. The PB-PMDP scheme is presented in Figure 3.

Remark. To lower storage overhead on the CSP and reduce the communication cost, the data owner creates an aggregated tag σ_j for the blocks having the same index in all copies. The aggregated tag $\sigma_j = \prod_{i=1}^{n} \sigma_{ij} \in \mathbb{G}_1$. Thus, the PB-PMDP scheme proposed in [18] requires cloud servers to store only *m* tags for the files copies $\tilde{\mathbb{F}}$ (not *mn* tags). The owner outsources $\{\tilde{\mathbb{F}}, \Phi, ID_F\}$ to the CSP ($\Phi = \{\sigma_j\}_{1\le j\le m}$) and delete the local copies from its local storage.

Barsoum *et al*. [18, 38] discussed how to identify corrupted copies by slightly modifying the PB-PMDP scheme. The owner generates a tag for each data block but does not perform the aggregation step. i.e., $\Phi = \{\sigma_{ij}\}, i: 1 \to n$ and $j: 1 \to m$. The CSP computes a response $\mu = \{\mu_{ik}\}(i: 1 \to n, k: 1 \to s)$, and $\sigma = \prod_{(j,r_j)\in Q}\left[\prod_{i=1}^{n} \sigma_{ij}\right]^{r_j} \in \mathbb{G}_1$. The verifier receives a proof $\mathbb{P} = \{\sigma, \mu\}$, and validates $\mathbb{P}$ using the verification equation (step 6 of Figure 3). In case of failed verification, the CSP will be asked to send $\sigma = \{\sigma_i\}_{1\le i\le n}$, where $\sigma_i = \prod_{(j,r_j)\in Q} \sigma_{ij}^{r_j} \in \mathbb{G}_1$. Thus, the verifier has two lists σList = $\{\sigma_i\}_{1\le i\le n}$ and μList = $\{\mu_{ik}\}(i: 1 \to n, k: 1 \to s)$. By applying binary search technique, the verifier will be able to identify indices of corrupted copies.

B. Provable Multi-Copy DYNAMIC Data Possession

In this subsection, we address PDP constructions that handle multiple copies of *dynamic* data over cloud servers. Dynamic data means that the owner issues requests to update/delete/add some blocks to the outsourced copies. The owner needs to make sure that all copies are consistent with the most recent modification requests that have been issued.

Map-Version Table-Based Scheme.

Barsoum *et al*. [21] proposed a map-based provable multi-copy dynamic data possession (MB-PMDDP) scheme that provides an evidence to the customers that the CSP is not cheating by storing fewer copies, and supports outsourcing of dynamic data. The MB-PMDDP scheme is based on using a small data structure, which they call a map-version table.

The map-version table (MVT) is a small *dynamic* data structure stored on the verifier side to validate the integrity and consistency of all file copies outsourced to the CSP. The MVT contains three columns: serial number (*SN*), block number (*BN*), and block version (*BV*). The *SN* is used to indicate the block position. The *BN* is used to make *logical* numbering/indexing to the file blocks. The *SN* and *BN* represent a mapping from physical index to logical index. The *BV* is used to indicate the current version of the data blocks. The *BV* is initialized to 1 for each block, and will be increment by 1 if the block has been updated.

Figure 4 presents how the MVT is changing with the dynamic operations on file copies $\tilde{\mathbb{F}}$ of a file $F = \{b_j\}_{1\le j\le 8}$. When the copies are initially created (Figure 4a), $SN_j = BN_j$, and $BV_j = 1$: $1 \le j \le 8$. In Figure 4b, BV_5 is incremented by 1 due to modifying the block at index 5 for all copies. Figure 4c shows that a new table entry $\langle 4, 9, 1 \rangle$ has been inserted in the MVT after SN_3 due to inserting a new data block after position 3 in the file copies $\tilde{\mathbb{F}}$. For the entry $\langle 4, 9, 1 \rangle$, 4 is the physical position of the newly inserted block, 9 is the new logical block number computed by incrementing the maximum of all previous logical block numbers, and 1 is the version of the new block. To delete a block at index 2 from all copies, the MVT deletes the table entry at SN_2 and shifts all subsequent entries one position up (Figure 4d).

In [21], for a file F = $\{b_1, b_2, \dots, b_m\}$, the data owner generates *n* distinct copies $\tilde{\mathbb{F}} = \{\tilde{F}_1, \tilde{F}_2, \dots, \tilde{F}_n\}$, where a copy $\tilde{F}_i = \{\tilde{b}_{ij}\}_{1\le j\le m}$. The block $\tilde{b}_{ij} = E_K(i||b_j)$, is divided into *s* sectors $\{\tilde{b}_{ij1}, \tilde{b}_{ij2}, \dots, \tilde{b}_{ijs}\}$. Thus, the file copy $\tilde{F}_i = \{\tilde{b}_{ijk}\}$ $(i: 1 \to n, j: 1 \to m, \text{and } k\ 1 \to s)$, where each sector $\tilde{b}_{ijk} \in \mathbb{Z}_p$ for some large prime p.

The owner generates a tag σ_{ij} for each block $\tilde{b}_{ij}$ as $\sigma_{ij} = \left(H(ID_F||BN_j||BV_j) \cdot \prod_{k=1}^{s} u_k^{\tilde{b}_{ijk}}\right)^x \in \mathbb{G}_1$ $(i: 1 \to n, j: 1 \to m, \text{and } k\ 1 \to s)$. In order to lower the storage overhead on cloud servers and reduce the communication cost, the owner computes an aggregated tag σ_j for the blocks having the same index in each copy $\tilde{F}_i$ as $\sigma_j = \prod_{i=1}^{n} \sigma_{ij} \in \mathbb{G}_1$.

Setup

– File $F = \{b_j\}_{1 \le j \le m}$

– A bilinear map $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, g is a generator for $\mathbb{G}_2$, private key $x \in \mathbb{Z}_p$, and public key $\{y = g^x \in \mathbb{G}_2, (u_1, u_2, \dots, u_s) \in_R \mathbb{G}_1\}$

– File identifier $ID_F = Filename||n||m||u_1|| \dots ||u_s$

– π_{key}, ψ_{key} are keyed pseudo-random permutation and pseudo-random functions, respectively

Data Owner

– Generate *n* distinct copies $\widetilde{\mathbb{F}} = \{\tilde{F}_1, \tilde{F}_2, \dots, \tilde{F}_n\}$, where $\tilde{F}_i = E_K(i||F)_{1 \le i \le n}$. Each copy is divided into blocks of sectors. i.e., $\tilde{F}_i = \{\tilde{b}_{ijk}\}$, where $\tilde{b}_{ijk} \in \mathbb{Z}_p (1 \le j \le m\ , 1 \le k \le s)$

– Generate a block tag $\sigma_{ij} = (H(ID_F||j) \cdot \prod_{k=1}^{s} u_k^{\tilde{b}_{ijk}})^x \in \mathbb{G}_1$.

– Generate a set of aggregated block tags $\Phi = \{\sigma_j\}_{1 \le j \le m}$ for the blocks having the same index in each file copy $\tilde{F}_i$, where $\sigma_j = \prod_{i=1}^{n} \sigma_{ij} \in \mathbb{G}_1$.

– Sends $\{\widetilde{\mathbb{F}}, \Phi, ID_F\}$ to the cloud servers and delete local data copies and block tags

Challenge Response

Verifier		Cloud Servers
2. Picks c (# of blocks to be challenged), and two fresh keys k_1 and k_2		
3. Generates $Q = \{(j, r_j)\}$, $j = \pi_{k1}(l)_{1<=l<=c}$ and $\{r_j\} = \psi_{k2}(l)_{1<=l<=c}$		
	c, k_1, k_2 →	
		3. Create Q as done by the verifier
		4. Computes $\sigma = \prod_{(j,r_j) \in Q} \sigma_j^{r_j} \in \mathbb{G}_1$
		5. Computes $\mu = \{\mu_{ik}\}(i: 1 \to n, k: 1 \to s)$ $\mu_{ik} = \sum_{(j,r_j) \in Q} r_j \cdot \tilde{b}_{ijk} \in \mathbb{Z}_p$
	← σ, μ	
6. Checks $\hat{e}(\sigma, g) \stackrel{?}{=} \hat{e}(\left[\prod_{(j,r_j) \in Q} H(ID_F||j)^{r_j}\right]^n \cdot \prod_{k=1}^{s} u_k^{\sum_{i=1}^{n} \mu_{ik}}, y)$		

Figure 3. The PB-PMDP scheme by Barsoum *et al*. [18, 38].

The dynamic operations in the MB-PMDDP scheme [21] are performed at the block level via a request in the general form $\langle ID_F, BlockOp, j, \{b_i^*\}_{1 \le i \le n}, \sigma_j^* \rangle$, where ID_F is the file identifier and $BlockOp$ corresponds to BM (block modification), BI (block insertion), or BD (block deletion). The parameter j indicates the position of the block to be modified, $\{b_i^*\}_{1 \le i \le n}$ are the new block values for all file copies, and , σ_j^* is the new aggregated tag for the new blocks. The challenge response protocol in the MB-PMDDP scheme [21] is summarized in Figure 5.

Hash-Based Scheme.

Barsoum *et al*. [21] proposed another scheme — Tree-Based Provable Multi-Copy Dynamic Data Possession (TB-PMDDP) — to verify the integrity of multiple copies of dynamic data based on using Merkle hash trees (MHTs), which area binary tree structures used to efficiently verify the integrity of the data. The MHT is a tree of hash values where the leaves of the tree are the hashes of the data blocks. Figure 6a presents an example of an MHT used for verifying the integrity of a file *F* divided into 8 blocks (*h* is cryptographic hash function, *e.g.*, SHA-2).

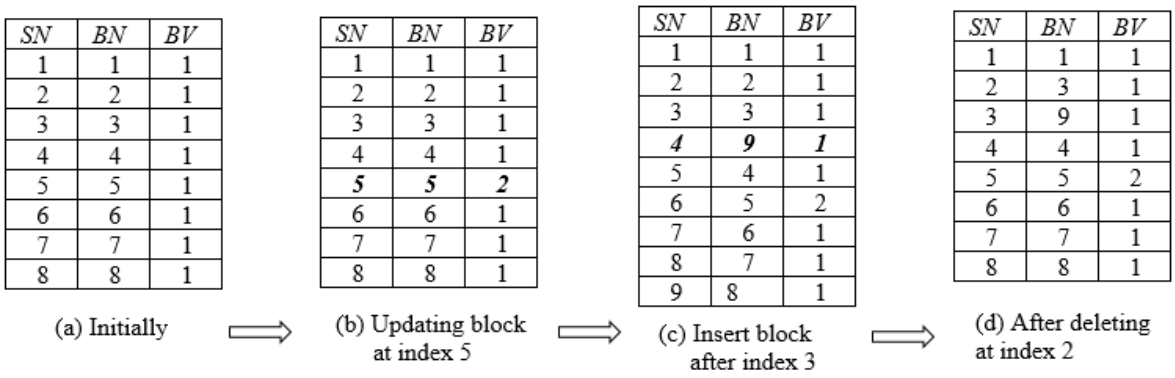


Figure 4. Changes in the MVT due to different dynamic operations on copies of a file F = $\{b_j\}_{1\le j\le 8}$

– $\pi_{key}(.)$ is a pseudo-random permutation (PRP): $key \times \{0,1\}^{\log_2(m)} \rightarrow \{0,1\}^{\log_2(m)}$

– $\psi_{key}(.)$ is a pseudo-random function (PRF): $key \times \{0,1\}^{*} \rightarrow \mathbb{Z}_p$

Verifier **Cloud Servers**

4. Generates a PRP key k_1 and a PRF key k_2
5. Picks c (# of blocks in the challenge vector)
6. Generates $Q = \{(j, r_j)\}$,
   $j = \pi_{k1}(l)_{1\le l\le c}$ and $\{r_j\} = \psi_{k2}(l)_{1\le l\le c}$

c, k_1, k_2 →

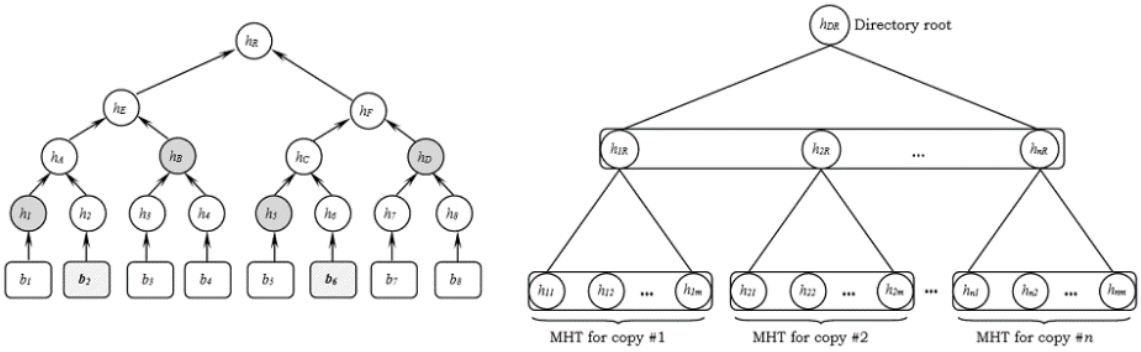
4. Uses k_1 and k_2 to generate the set Q
5. Computes $\sigma = \prod_{(j,r_j)\in Q} \sigma_j^{r_j} \in \mathbb{G}_1$
6. Computes $\mu = \{\mu_{ik}\}(i: 1 \rightarrow n, k: 1 \rightarrow s)$
   $\mu_{ik} = \sum_{(j,r_j)\in Q} r_j \cdot \tilde{b}_{ijk} \in \mathbb{Z}_p$

← σ, μ

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

Figure 5. Challenge response protocol in the MB-PMDDP scheme [21].

In the first level, the hash $h_j = h(b_j)$ $(1 \le j \le 8)$. At upper levels, $h_A = h(h_1||h_2), h_B = h(h_3||h_4)$, and so on. The root $h_R = h(h_E||h_F)$ is the hash of the root node that is used to validate the integrity of all data blocks. The data blocks $\{b_1, b_2, \dots, b_8\}$ are stored on a remote server, and only the root value h_R is stored locally on the verifier side. To validate the integrity of the blocks b_2 and b_6, the server responds by sending b_2 and b_6 with the authentication paths $\{h_1, h_B, h_5, h_D\}$ that are used to reconstruct the root of the MHT. The verifier regenerate the root hash value using the received blocks and the authentication paths. The verifier computes $h_2 = h(b_2), h_6 = h(b_6), h_A = h(h_1||h_2), h_C = h(h_5||h_6)$,



(a) Merkle tree (a) Directory tree

Figure 6. Merkle hash trees

$h_E = h(h_A||h_B), h_F = h(h_C||h_D),$ and $h_R = h(h_E||h_F)$. The re-computing h_R, it is compared with the authentic value stored locally on the verifier side.

The MHT is commonly used to authenticate the *values* of the data blocks. For outsourced *dynamic* data, the verifier needs to authenticate both the *values* and the *positions* of the data blocks. In order to validate the positions of the blocks, the leaf nodes of the MHT are considered to have a specific order, *e.g.*, left-to-right sequence [23]. Thus, the hash value of any internal node = $h(left\ child\ ||right\ child)$, *e.g.*, $h_A = h(h_1||h_2) \neq h(h_2||h_1)$. In addition, the authentication path is viewed as an *ordered* set, and thus any leaf node is uniquely specified by following the used sequence of constructing the root of the MHT.

In the TB-PMDDP scheme [21], an MHT is created for each copy and then the roots of the individual trees are used to construct a hash tree which is called a directory MHT. The leaf nodes of the directory MHT are the root nodes of the MHTs of the file copies, and thus, the directory MHT can be used to authenticate the integrity of all file copies in a hierarchical manner. Figure 6b shows the directory tree generated for *n* file copies. The verifier can maintain only one hash value $\mathcal{M} = h(ID_F||h_{DR})$, where ID_F is a unique file identifier, and h_{DR} is the authenticated directory root value that can be used to periodically check the integrity of all file copies.

Probabilistic Encryption-Based Scheme.

To verify the integrity and completeness of multiple copies of dynamic data, Mukundan et al. [24] proposed a dynamic multi-replica provable data possession (DMR-PDP) scheme. Their scheme is based on utilizing Paillier encryption algorithm, which is a probabilistic encryption model. The Paillier encryption is used to generate distinct replicas of a file $F = \{b_1, b_2, \dots, b_m\}$. A file replica F_i is generated as $F_i = \left\{ (1+N)^{b_j}\left(k_i r_{i,j}\right)^N \right\}_{j=1}^{m}$, where m is the number of file blocks, i is the replica number, j is the block index, k_i is a random number used to identify the replica number, $r_{i,j}$ indicates a random number used for the Paillier encryption algorithm, and N is the owner public key. The owner generates a tag for each data block b_j as $\sigma_j = (h(F).u^{b_j \cdot N\}})^d$, where h is a hash function, d is the owner's private key, and u is a generator for a bilinear group $\mathbb{G}$. To modify a data block b_j with b_j', the data owner calculates $\Delta b_j = b_j' - b_j$, encrypts Δb_j using Paillier encryption as $E(\Delta b_j) = (1 + \Delta b_j N) r^N$, and generates a new tag for the modified block b_j', as $\sigma_j' = \left(h(F).u^{b_j' \cdot N}\right)^d$. Then, the data owner sends to the CSP $E(\Delta b_j)$, σ_j', and a random challenge to ensure the integrity of the modification operation. When the CSP receives the modification request, all file copies are updated by performing a homomorphic addition operation, *i.e.*, $E(b_j') = E(b_j).E(\Delta b_j)$ on all file copies.

IV. Comparative Analysis

In this sub-section we evaluate the performance and compare the PMDP schemes: MR-PDP scheme [17], PB-PMDP scheme [18], MB-PMDDP scheme [21], TB-PMDDP scheme [24], and DMR-PDP scheme [24]. The first two schemes, namely MR-PDP and PB-PMDP, supports only static data. In our analysis, we assume that the desired security level for the comparison is 128-bit. Thus, we utilize an elliptic curve defined over finite field $GF(p)$ with $|p|$ = 256 bits (a point on this curve can be represented by 257 bits using compressed representation [25]), a cryptographic hash of size 256 bits (*e.g.*, SHA-256), and the size of the RSA modulus N is 3072 bits.

The computation costs for the schemes are estimated in terms of used cryptographic operations: $\mathcal{H}_{\mathbb{G}}$ (hashing to $\mathbb{G}$, where $\mathbb{G}$ indicates a group of points over a suitable elliptic curve in the bilinear pairing), $\mathcal{H}_{QR_N}$ (hashing to QR_N, where QR_N is a set of quadratic residues modulo N), $\mathcal{E}_{\mathbb{G}}$ (exponentiation in $\mathbb{G}$), $\mathcal{E}_{\mathbb{Z}_N}$ (exponentiation in $\mathbb{Z}_N$), $\mathcal{M}_{\mathbb{G}}$ (multiplication in $\mathbb{G}$), $\mathcal{M}_{\mathbb{Z}}$ (multiplication in $\mathbb{Z}$), $\mathcal{M}_{\mathbb{Z}_p}$ (multiplication in $\mathbb{Z}_p$), $\mathcal{D}_{\mathbb{Z}}$ (division in $\mathbb{Z}$), $\mathcal{A}_{\mathbb{Z}_p}$ (addition in $\mathbb{Z}_p$), $\mathcal{A}_{\mathbb{Z}}$ (addition in $\mathbb{Z}$), $\mathcal{P}$ (bilinear pairing), E_K (encryption using K), D_K (decryption using K), $\mathcal{R}$ (random-number generation), and h_{SHA} (cryptographic hashing).

Let n, m, and s denote the number of copies, the number of blocks per copy, and the number of sectors per

Table 1: Comparison for PDP schemes for multiple data copies

Cost		MR-PDP [17]	PB-PMDP [18]	MB-PMDDP [21]	TB-PMDDP [21]	DMR-PDP [24]
System Setup	Copies Generation	$E_K + nm\,\mathcal{R} + nm\mathcal{A}_{\mathbb{Z}}$	nE_K	nmE_K	nmE_K	nmE_K
	Tag Generation	$2m\mathcal{E}_{\mathbb{Z}_N} + m\mathcal{M}_{\mathbb{Z}} + m\mathcal{H}_{QR_N}$	$(s+1)nm\mathcal{E}_{\mathbb{G}} + nm\mathcal{H}_{\mathbb{G}} + (ns+n-1)m\mathcal{M}_{\mathbb{G}}$	$(s+1)nm\mathcal{E}_{\mathbb{G}} + nm\mathcal{H}_{\mathbb{G}} + (sn+n-1)m\mathcal{M}_{\mathbb{G}}$	$(s+1)nm\mathcal{E}_{\mathbb{G}} + nm\mathcal{H}_{\mathbb{G}} + (sn+n-1)m\mathcal{M}_{\mathbb{G}}$	$nm(\mathcal{H}_{\mathbb{G}} + \mathcal{M}_{\mathbb{G}}) + nm(2\mathcal{E}_{\mathbb{G}} + \mathcal{M}_{\mathbb{Z}_p})$
	Metadata Generation	—	—	—	$nm\mathcal{H}_{\mathbb{G}} + (2m+1)nh_{SHA}$	—
Storage	File Copies	$n\|F\|$	$n\|F\|$	$n\|F\|$	$n\|F\|$	$n\|F\|$
	CSP Overhead	$3072m$ bits	$257m$ bits	$257m$ bits	$(257+512n)m$ bits	$257m$ bits
	Verifier Overhead	—	—	$64m$ bits	256 bits	—
Communication Cost	Challenge	$3328 + \log_2(c)$ bits	$256 + \log_2(c)$ bits	$256 + \log_2(c)$ bits	$256 + \log_2(c)$ bits	$256 + \log_2(c)$ bits
	Response	$3072(n+1)$ bits †	$257 + 256ns$ bits	$257 + 256sn$ bits	$257 + 256sn + (256\log_2(m) + 257)cn$ bit‡	3328 bits
Computation Cost	Proof	$(c+n)\mathcal{E}_{\mathbb{Z}_N} + (cn+c-1)\mathcal{M}_{\mathbb{Z}} + (c-1)n\mathcal{A}_{\mathbb{Z}}$	$c\mathcal{E}_{\mathbb{G}} + (c-1)\mathcal{M}_{\mathbb{G}} + csn\mathcal{M}_{\mathbb{Z}_p} + (c-1)sn\mathcal{A}_{\mathbb{Z}_p}$	$c\mathcal{E}_{\mathbb{G}} + (c-1)\mathcal{M}_{\mathbb{G}} + csn\mathcal{M}_{\mathbb{Z}_p} + (c-1)sn\mathcal{A}_{\mathbb{Z}_p}$	$c\mathcal{E}_{\mathbb{G}} + (c-1)\mathcal{M}_{\mathbb{G}} + cn\mathcal{H}_{\mathbb{G}} + csn\mathcal{M}_{\mathbb{Z}_p} + (c-1)sn\mathcal{A}_{\mathbb{Z}_p}$	$c(\mathcal{H}_{\mathbb{G}} + \mathcal{M}_{\mathbb{G}} + 2\mathcal{E}_{\mathbb{G}} + \mathcal{M}_{\mathbb{Z}_p}) + (2nc+n)\mathcal{E}_{\mathbb{G}} + nc\mathcal{M}_{\mathbb{G}}$
	Verify	$(2n+c+1)\mathcal{E}_{\mathbb{Z}_N} + c\mathcal{H}_{QR_N} + \mathcal{D}_{\mathbb{Z}} + (cn+c+n-1)\mathcal{M}_{\mathbb{Z}} + (c-1)n\mathcal{A}_{\mathbb{Z}}$	$2\mathcal{P} + (c+s+1)\mathcal{E}_{\mathbb{G}} + c\mathcal{H}_{\mathbb{G}} + (c+s-1)\mathcal{M}_{\mathbb{G}} + (n-1)s\mathcal{A}_{\mathbb{Z}_p}$	$2\mathcal{P} + (c+s+1)\mathcal{E}_{\mathbb{G}} + c\mathcal{H}_{\mathbb{G}} + (c+s-1)\mathcal{M}_{\mathbb{G}} + (n-1)s\mathcal{A}_{\mathbb{Z}_p}$	$2\mathcal{P} + (c+s)\mathcal{E}_{\mathbb{G}} + cn\mathcal{H}_{\mathbb{G}} + (cn+s-1)\mathcal{M}_{\mathbb{G}} + (n-1)s\mathcal{A}_{\mathbb{Z}_p} + (c\log_2(m) + 2)nh_{SHA}$ ‡	$(2\mathcal{M}_{\mathbb{Z}_p} + \mathcal{E}_{\mathbb{G}})n + n\mathcal{A}_{\mathbb{Z}_p} + D_K + 3\mathcal{E}_{\mathbb{G}} + \mathcal{H}_{\mathbb{G}} + \mathcal{M}_{\mathbb{Z}_p}$
Dynamic support		×	×	✓	✓	✓
Dynamic Operations	Communication	N/A	N/A	"Request"	"Request" + $O(n\log_2(m))$	"Request" + $O(1)$
	State update	N/A	N/A	—	$O(n\log_2(m))h_{SHA}$	—

† There is an optimization for this response to be $3072 + 256n$ bits using hashing.

‡ $\log_2(m)$ is the upper bound of the authentication path length when c >1.

block, respectively. Let c denotes the number of blocks in the challenge vector, and $|F|$ denotes the size of the file copy. Let the keys used with the PRP (pseudo-random permutation) and the PRF (pseudo-random function) be of size 128 bits. Table 1 presents a theoretical analysis for the setup, storage, communication, computation, and dynamic operations costs of the presented schemes.

V. Proof of Retrievability

Proof of retrievability (POR) is a complementary approach to PDP and is considered to be stronger than PDP in the sense that the entire data file can be reconstructed from the responses that are reliably transmitted from the remote servers. The fact that POR schemes encode the data file (e.g., using erasure codes) before outsourcing to remote sites allows more error-resiliency. Thus, if it is a crucial demand for the system to detect any modification/deletion of tiny parts of the data file, then encoding could be applied before outsourcing data to remote servers.

It is important to note that we can achieve data redundancy by using replication or coding schemes, where the former is the simplest way that can be adopted by many storage systems. The storage cost to make n copies of a file of size $|F|$ bits is $n|F|$ bits. Using erasure code schemes, the file is fragmented into m blocks, encoded into ℓ blocks ($\ell > m$) [26], and stored on ℓ different servers (one code block per server). Thus, the storage cost is $\frac{|F|}{m}\ell$ bits. Any m out of the ℓ servers can be used to regenerate the original file.

For PDP schemes that handle multiple data copies, they consider economically-motivated CSPs that may attempt to store a fewer number of file copies to reduce the used space on their infrastructure. The CSPs have almost no financial benefit by deleting only a small portion of a copy of the file. Using erasure codes to achieve redundancy has less storage cost; however, duplicating data file across multiple servers achieves scalability (i.e., if the number of users grows, then with more copies of data the user access time can be kept below a certain threshold). This scalability feature is a crucial customer requirement in cloud computing systems. A file that is replicated and stored on multiple servers at different geographic locations can help reduce access time and communication cost for users. Furthermore, in coding-based systems, the CSP has to access at least m servers to reconstruct the original data file, and thus increased time overhead (network latency and computation time to decode data blocks) occurs at the CSP side.

To authenticate data across multiple servers, Schwartz and Miller [27] have proposed the use of algebraic signatures. In their scheme, file corruptions can be detected using keyed algebraic encoding and stream cipher encryption. The communication complexity in the scheme of [27] is an issue for it is linear with respect to the queried data size. In addition, the security of their model is not proven and remains in question [7].

One of the first research to consider formal models for POR schemes is the work done by Juels and Kaliski [28]. In their model, the data is first encrypted then disguised blocks (called sentinels) are embedded into the ciphertext. In order to detect data corruption done by the server, the sentinels are hidden among the regular file blocks. During the verification phase, the verifier asks for randomly selected sentinels and checks whether they are modified or not. In case the server corrupts/deletes parts of the data, then sentinels would also be influenced with a certain probability. The scheme in [28] allows only for a limited number of challenges on the data files, which is specified by the number of sentinels embedded into the data file. This puts limitation on the number of challenges 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.

To unboundedly challenge the remote servers, Shacham and Waters [19] proposed a compact proof of retrievability model. In their scheme, they proposed 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. Two HLAs have been proposed in [19]: one is based on the pseudo-random function, and the other is based on the BLS signature [29].

POR schemes can be classified into two main classes: erasure coding-based schemes and network coding-based schemes. Bowers *et al.* [30] and Wang *et al.* [31] are examples of the POR models that belongs to the first class. In [30] a distributed cryptographic system known as HAIL (High-Availability and Integrity Layer) has been presented, which improves upon POR deployed on individual servers. Their system allows a set of servers to prove to a data owner that the outsourced data is intact and retrievable. A Secure Distributed Storage (SDS), which is a flexible and lightweight auditing scheme has been proposed by Wang *et al.* [31]. The SDS is based on the homomorphic tags and the Reed-Solomon erasure-correcting code to guarantee the data integrity and availability in an erasure code distributed storage system. Other erasure coding-based schemes include [32-34]. The communication overhead is the main concern of erasure coding-based schemes. To reduce the communication overhead, some proposals use network coding to develop POR schemes [35-37].

VI. Summary And Open Research Issues

In this paper, we have reviewed the concept of PDP as a technique that allows an entity to prove that the data is in its possession for validating data integrity over remote servers. We have provided an extensive survey and a comparative analysis for numerous provable multi-copy data possession (PMDP) schemes. The paper also discusses the design principles for different PMDP models and highlights some limitations. In our study, we have addressed PMDP protocols for *static* data, and PMDP schemes that handle *dynamic* behavior of outsourced data over cloud servers. PMDP models for dynamic data

support block-level operations, such as block modification, insertion, deletion, and append. In these models, the verifier is enabled to make sure that the outsourced data is consistent with the most recent modifications issued by the owner.

We have provided a comparative analysis for different PMDP schemes. The comparative analysis was done from different perspectives: (i) the computation cost to generate data copies, tags, and metadata; (ii) the storage overhead on both the verifier and server sides; (iii) the communication cost to send a challenge vector and receive a response; (iv) the computation overhead on the server side to prove data possession and the verifier's computations complexity to check server responses; and (v) the cost of dynamic update request.

Furthermore, the paper highlighted the concept of POR as a complementary approach to PDP. The main idea of POR schemes is to apply encoding to data files before outsourcing, which allows to reconstruct the entire data file utilizing the responses that are reliably transmitted from the server.

A lot of research work has taken some steps towards mitigating the concerns of outsourcing data storage, but much effort remains to achieve the wide acceptance of such a growing paradigm. Below we summarize some open research problems that need to be investigated and tackled:

- **Ensuring data replication across diverse geographic location.** Clients need to be sure that the data copies are actually replicated on different multiple drives or different multiple data centers. Replicating data in different geographic locations is crucial to prevent simultaneous failure caused by natural disasters or power outages. Moreover, it is effective in reducing access time and communication cost for users in different parts in the world. This work may require collaboration between researchers from both industry (to build data center components, services, and software) and academia (to provide a theoretical framework and mathematical models for the verification process).
- **Self-organized data replication over cloud servers.** Current data centers are subject to failure of any type, and higher access to the data stored can be one reason for such failure. As the number of access requests to outsourced data increases, its availability becomes more complex.

  One of the future directions is to address the problem of designing self-managed storage schemes that can dynamically adapt to varying query load by allocating/deallocating storage space for data copies on cloud servers. An optimization model is needed to specify the optimal number of copies and their storage locations across the servers. Through this model we can minimize the response time for data access requests, and optimize the use of CSP's storage capacity.
- **User authentication for cloud computing systems.** The development of cloud computing encourages the use of resource-constrained devices (PDA or cell phones) on the client side. Rather than data storage and software installation on local devices, users will authenticate in order to be able to access the data and use cloud applications. This computing model makes software piracy more difficult and enables centralized monitoring. Although cloud computing architecture stimulates mobility of users, it increases the need of secure authentication.

  User authentication based on passwords in not an efficient approach for sensitive data/applications on the cloud. The use of passwords is a major point of vulnerability in computer security, as passwords are often easy to guess by automated programs running dictionary attacks. Moreover, users cannot remember very long passwords, and usually they use some meaningful passwords making them subject to dictionary attacks.

  Implicit authentication is one of the visions to address user authentication problem. One can use learning algorithms to construct a model for the user based on the past behavior, and then the recent behavior is compared with the user model to authorize legitimate users.
- **Outsourcing computation to untrusted cloud servers.** Outsourcing computation is a growing desire for resource-constrained clients to benefit from powerful cloud servers. Such clients prefer to outsource computationally-intensive operations, e.g., image manipulation, to the cloud and yet obtaining a strong assurance that the computations are correctly performed. To save the computational resources, a dishonest cloud service provider may totally ignore the computations, or execute just a portion of them. Sometimes the computations outsourced to the cloud are so critical that it is essential to preclude accidental errors during the processing.

  The ability to verify computations and validate the returned results is a key requirement of cloud customers. Another imperative point is that the amount of work performed by the clients to verify the outsourced computations must be substantially cheaper than performing the actual computations on the client side. The area of verifiable computations and outsourcing computational tasks to untrusted cloud servers is a crucial domain to be investigated. It is also interesting to address mutual trust feature, so a client who receives incorrect results from cloud servers can detect and prove this misbehavior. Moreover, a dishonest client must not be able to falsely accuse a cloud service provider and claim that the outsourced computations are malformed.

REFERENCES

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

[2] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, ser. CCSW '09. ACM, 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 http://www.pcworld.com/businesscenter/article/142549/amazons s3 down for severalhours.html, 2008.

[5] B. Krebs, "Payment processor breach may be largest ever."

[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 *Proceedings of the 10th International Conference on Information and Communications Security*, ser. ICICS '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 419–434.

[8] Y. Deswarte, J.-J. Quisquater, and A. Sa¨ıdane, "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. Sebe , J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 8, 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] Amazon Simple Storage Service (Amazon S3), http://aws.amazon.com/s3/.

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

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

[18] A. F. Barsoum and M. A. Hasan, "Integrity verification of multiple data copies over untrusted cloud servers," *in 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*. IEEE Computer Society, 2012, pp. 829–834.

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

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

[21] A. Barsoum and M. Hasan, "Provable multi-copy dynamic data possession in cloud computing systems," *Information Forensics and Security, IEEE Transactions on*, vol. 10, no. 3, pp. 485–497, March 2015.

[22] R. C. Merkle, "Protocols for public key cryptosystems," *IEEE Symposium on Security and Privacy*, vol. 0, p.122, 1980.

[23] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine, "A general model for authenticated data structures," *Algorithmica*, vol. 39, 2001.

[24] R. Mukundan, S. Madria, M. Linderman, and N. Rome, "Replicated data integrity verification in cloud." *IEEE Data Eng. Bull.*, vol. 35, no. 4, pp. 55–64, 2012.

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

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

[27] 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.

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

[29] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from theWeil 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.

[30] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," in *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2009, pp. 187–198.

[31] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *Services Computing, IEEE Transactions on*, vol. 5, no. 2, pp. 220–232, April 2012.

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

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

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

[35] B. Chen, R. Curtmola, G. Ateniese, and R. Burns, "Remote data checking for network coding-based distributed storage systems," in *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, ser. CCSW '10. New York, NY, USA: ACM, 2010, pp. 31–42.

[36] L. Anh and A. Markopoulou, "Nc-audit: Auditing for network coding storage," in *International Symposium on Network Coding* (NetCod), 2012, pp. 155–160.

[37] H. C. H. Chen and P. P. C. Lee, "Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation," *IEEE Trans. Parallel Distrib. Syst*., vol. 25, no. 2, pp. 407–416, Feb. 2014.

[38] A. F. Barsoum and M. A. Hasan, "Verifying outsourced replicated data in cloud computing storage systems," *International Journal of Computer Applications*, vol. 99, no. 7, pp. 1–13, August 2014.

Authors' Profiles

Ayad Barsoum is an Assistant Professor in Computer Science department at St. Mary's University, San Antonio, Texas. Dr. Barsoum received his Ph.D. degree from the Department of Electrical and Computer Engineering at the University of Waterloo (UW), Ontario, Canada in 2013. He is a member of the Centre for Applied Cryptographic Research at UW. Dr. Barsoum has been awarded The Amazon Web Services in Education Faculty Grant for funding his research and teaching activities. At UW, Barsoum has received the Graduate Research Studentship, the International Doctoral Award, and the University of Waterloo Graduate Scholarship. Dr. Barsoum received his B.Sc. and M.Sc. degrees in computer science from Ain Shams University, Cairo, Egypt.

Athanasios V. Vasilakos is currently a professor at Lulea University of Technology, Sweden. He has authored or co-authored over 200 technical papers in major international journals and conferences. He is the author/coauthor of five books and 20 book chapters in the areas of communications. Prof. Vasilakos has served as General Chair, Technical Program Committee Chair, TPC member for many international conferences. He served or is serving as an Editor or/and Guest Editor for many technical journals, such as the IEEE Transactions on Network and Services Management, IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, IEEE Transactions on Information Technology in Biomedicine, IEEE Transactions on Computers, ACM Transactions on Autonomous and Adaptive Systems, the IEEE JSAC special issues of May 2009, Jan 2011, March 2011, the IEEE Communications Magazine, ACM/Springer Wireless Networks(WINET), ACM/Springer Mobile Networks and Applications(MONET). He is founding Editor-in-Chief of the International Journal of Adaptive and Autonomous Communications Systems (IJAACS, http://www.inderscience.com/ijaacs) and the International Journal of Arts and Technology (IJART, http://www.inderscience.com/ijart). He is the General Chair of the Council of Computing of the European Alliances for Innovation.