Article can be accessed online at http: //www.publishingindia.com

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

Ayad Barsoum*

Abstract

Every hour many organizations generate terabytes of data, and that may exceed their storage ability. Storing and manipulating such big data is a costly process.That is why outsourcing data storage to remote cloud service providers (CSPs) is currently a preferred storage model to mitigate the burden of local data storage. 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. The cloud customers want to make sure that the CSP is not cheating by storing fewer number of data copies. 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. In this paper, we focus on two sets on PMDP schemes: protocols for static data, and schemes for 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.

Keyword : Cloud Computing, Outsourcing Data Storage, Multiple Data Copies, Integrity Verification

I. Introduction

In cloud computing (CC) model, many distributed computers in different geographic locations are interconnected to provide 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 come in different forms: SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service). The SaaS is the widely used model of CC services, where it provides software applications to the users over the internet. Google Docs, Google Calendar, and Zoho Writer are examples of SaaS model. PaaS provides operating system, frameworks, and libraries that support clients' applications. Using IaaS model, customers can rent and use the provider's resources. Therefore, the clients can install any applications including operating systems.

A number of key benefits can be gained by adopting the CC technology. For example, 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. Also, this model of computation enables clients to use so many online applications and mitigate the management burden. Moreover, CC provides cut-price maintenance cost by outsourcing the storage and the computation. Another key advantage of CC model is the ability of horizontal and vertical scaling. Also, cloud computing provides more mobility where customers can access information from

* Assistant Professor in Computer Science Department, St.Mary's University, San Antonio, Texas United States.
Email: abarsoum@stmarytx.educience

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 been moving forward by about 80% as companies are now giving more attention to their core businesses other than technology-related stuff [3].

On the other side, outsourcing data storage to remote servers (which may not be fully trusted) can raise new concerns related to the privacy and the integrity of the data. Data encryption techniques can be used to release the privacy concerns. 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: (i) to stay in business, it is not common for the CSP to announce any data corruption, (ii) 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; and (iii) the cloud infrastructure is a an attractive target for many attacks. 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). If we are not able to find proper solutions to this problem, the acceptance of cloud technology will be at risk.

Efficient verification of the integrity of outsourced data becomes a formidable challenge because 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 cloud customers want an evidence that they pay for a real service, *i.e.*, the CSP is not cheating by storing fewer number of data copies. 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 Proof of retrievability (POR) is an integral part for PDP, and will be highlighted throughout the paper.

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 has been proposed as a mechanism to audit data over remote sites. A PDP model has been formalized by Ateniese *et al*. [6]. According to that model, the data file is fragmented into blocks and each block will be associated with a block tag generated by the owner. These tags will be combined later with the data blocks to proof the integrity of outsourced data. The data file along with the tags will be sent to be stored by the remote servers (the owner may delete its local copy of the file and the tags). To proof that the outsourced data are not corrupted, the verifier (which may be the data owner or another trusted third party) challenges some random data blocks and the remote servers compute a response based on these blocks and their tags. References [6-14] are examples of different PDP schemes. The schemes proposed in [6-14] did not consider multiple replica of the outsourced file, their main focus was on a single data copy.

Taking multiple copies of the data file helps to achieve a higher degree of availability. This is a normal procedure when we are preparing an important report (we save more than one copy on different locations) to recover from any loss. Replicating data files is also a normal procedure for organizations/governments/universities to achieve a higher level of availability. In CC system model, the CSP is replicating the data across multiple servers in exchange for some pre-specified fees paid by the customers. Thus, the clients need an evidence that they get a real service for what they are paying for (*i.e.*, they need a poof that the CSP is not cheating by storing a fewer number of data copies). In addition, 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.

Customers may ask the CSP to keep more copies for some kind of data that will be accessed by many authorized users over the world. The CSPs formulate their pricing model according to the replication strategy. As an example, Amazon provides two data storage standards: Amazon S3 and Amazon RRS (Reduced Redundancy Storage) [15]. With RRS policy, a fewer number of copies are stored to reduce the storage bill. As a consequence, the Amazon S3 prices are higher than that of the RRS. Unfortunately, the CSP can cheat and not store all data copies that have been agreed upon to save the cloud storage space. Thus, the clients need a technique to detect such a misbehavior from the CSP (in case it occurs). 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. Many PMDP schemes consider *static* data, which are not subject to frequent changes as the case in digital libraries. 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.

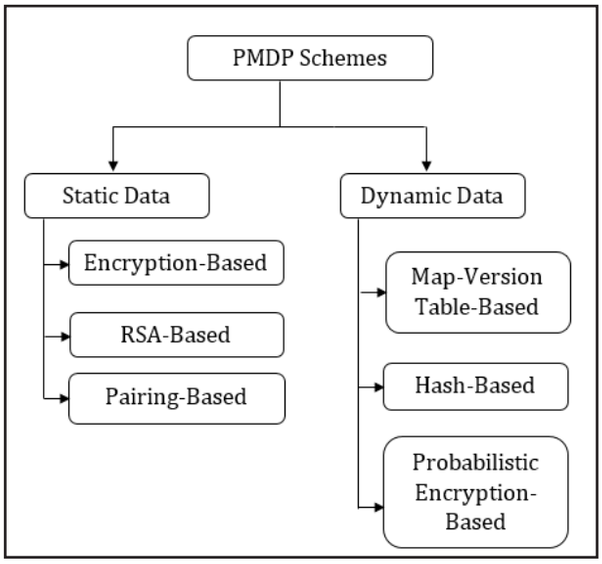

Fig. 1. Classification of PMDP Schemes

III. Provable Multi-Copy Data Possession

In cloud computing storage models, some customers want to achieve a higher level of scalability and availability for their data, so they ask the CSP to keep more copies across multiple servers. In such a case, the clients will be charged more fees. Thus, the clients need an evidence that the CSP is not cheating by storing a fewer number of copies and that clients are paying for real services.

In this section, we address PMDP schemes for multiple copies of *static* data, and PMDP constructions for multiple

copies of *dynamic* data. The main design principle for a provable multi-copy data possession protocol is to create distinct copies. If the outsourced copies are not differentiable (*i.e.*, they are just identical images of the original data file), the CSP can just keep one image/copy and pretend that all copies have been stored.

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. In their scheme, Barsoum *et al*. [16] used an encryption scheme with *n* distinct encryption keys to generate *n* data copies that are not identical (changing the key with each encryption creates a new data copy). The used encryption keys will be kept secret from the CSP. Hence, the cloud servers could not cheat by using one copy to answer the challenges for another. The basic scheme proposed in [16] can detect if the CSP is not storing all the *n* copies by verifying each stored copy on remote servers. 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 used *n* encryption keys will be known to (shared with) users who are authorized to access the outsourced data, but will be kept secret from the CSP. Moreover, it is common that when the user sends a data access request, he may receive a different copy with each request (*i.e.*, the same copy will not be retrieved each time to the same user). In typical cloud storage model, data copies are stored on more than one server and the server with the lightest load will respond to the user's request and retrieve the data copy. Thus, for the authorized user to be able to decrypt the received copy and access the plain data, there should be a flag about which decryption key should be used (*i.e.*, the retrieved copy should indicate its decryption key).

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. In case of corruption/failure of some outsourced copies, other copies stored on other server can be used for data recovery, and thus the MR-PDP model can help to achieve a higher level of availability. 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. 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 $\widehat{F}_i = \{\hat{b}_{ij}\}_{1 \le j \le m}, \hat{b}_{ij} = \hat{b}_j + r_{ij}$ (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 cloud storage model in the MR-PDP scheme [17] had a missing link, which is the link between the CSP and the users who have the right to access outsourced owner's file (authorized uses). When an authorized user wants to access the outsourced data, he will interact with the CSP to retrieve one of copies stored by the cloud servers.

Before decrypting the received copy, it needs to be *unmasked* (subtracting the random value r_{ij}), thus the authorized user needs to know the copy number i.

The CSP just retrieves the data copy and not the copy number, and thus the authorized user will not be able to properly decrypt it. Even if the data owner made a copy number i as a prefix to the outsourced copy (*i.e.*, outsourcing) $(i \| \widehat{F}_i)$, this will not help the authorized user to properly decrypt the received copy; the CSP can simply mess up with these numbers.

Private verifiability (*i.e.*, only the data owner can check data integrity/possession) is only supported by the MR-PDP scheme. Public verifiability is a main requirement for remote data auditing models to achieve mutual trust between the owner and the CSP. Performing the verification process through a trusted third party can help to achieve such mutual trust between the owner and the CSP. In Figure 2, to compute the value $r_{chal} = \sum_{j \in A} r_{zj}$, the verifier needs a subset of the random values $\{r_{ij}\}$. The

random values $\{r_{ij}\}$ are not public parameters, otherwise the CSP can compromise the storage system and keep only one data copy. That is why the MR-PDP scheme of [17] can support only private verifiability.

Setup

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

– $g$is a generatorofQR_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 $ed \equiv 1 \bmod (p-1)(q-1)$

– $\pi_k$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$F = b_j\ _{1 \le j \le m}$, where $b_j = E_K(b_j)$.

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

– Generates n distinct copies $F_i\ _{1 \le i \le n}$, $F_i = b_{ij}\ _{1 \le j \le m}$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$b_{ij} = b_j + r_{ij}$ (addition in $\mathbb{Z}$)

– Sends the copy F_i to a server$S_i, i: 1 \to n$

Checking possession of a replica F_z

OwnerRemote Server S_z

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

c, k, 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 $\rho = g_s^{\sum_{j \in A} b_{zj}} \bmod N$

← T, ρ

5. Computes $A = j = \pi_k(l)_{1 \le l \le c}$

6. Checks $\left(\frac{T^e}{\prod_{j \in A} H(v \| j)} \cdot g^{r_{chal}}\right)^s \stackrel{?}{=} \rho$, where $r_{chal} = \sum_{j \in A} r_{zj}$

Fig. 2. The PB-PMDP Scheme by Barsoum et al. [18, 38]

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. In a strong encryption model, if only 1 bit changes in the input data the output cipher will be completely changed (diffusion feature). Using this feature the PB-PMDP scheme generates differentiable data copies. The data owner generates *n* distinct copies $\tilde{\mathbb{F}} = \{\widetilde{F}_i\}_{i\le i\le n}$. 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 $\tilde{F}$ into *m* blocks and each block is fragmented into *s* sectors. Thus, $\tilde{F}_i = \{\tilde{b}_{ijk}\}$, where $i:1\rightarrow n, j:1\rightarrow m$, *k*: 1 $\rightarrow s$, 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. In the PB-PMDP scheme (to efficiently reduce the storage and communication overheads), the authors proposed that the owner aggregates the tags σ_{ij}: $\sigma_j = \prod_{i=1}^{n}\sigma_{ij} \in \mathbb{G}_1$. Thus, the PB-PMDP scheme proposed in [18, 38] requires cloud servers to store only *m* tags for the files copies $\tilde{\mathbb{F}}$ (not *mn* tags). The owner sends $\{\tilde{\mathbb{F}}, \Phi, ID_F\}$ to the remote servers ($\Phi = \{\sigma_j\}_{1\le j\le m}$).

Barsoum *et al*. [18, 38] performed fine tuning on their scheme to be able to identify data copies that have been corrupted (if any).There is no aggregation for the block tags, *i.e.*, $\Phi = \{\sigma_{ij}\}, i:1\rightarrow n$ and $j:1\rightarrow m$. The CSP computes a response $\mu = \{\mu_{ik}\}(i:1\rightarrow n, k:1\rightarrow s)$, 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). If the verification fails, the verifier requires the CSP to respond by $\sigma = \{\sigma_i\}_{1\le i\le n}$, where $\sigma_i = \prod_{(j,r_j)\in Q}\sigma_{ij}^{r_j} \in \mathbb{G}_1$. Thus, the verifier has two lists $\sigma\text{List} = \{\sigma_1\} 1\le i\le n$ and $\mu\text{List} = \{\mu_{ik}\}(i:1\rightarrow n, k:1\rightarrow s)$ 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 to enable the data owner to detect any cheating from the CSP side (*i.e.*, if the CSP stores a fewer number of data copies or if the copies are not up-to-date).A map-version table (a linked list of nodes of three integers) is a core component of the MB-PMDDP scheme.

The map-version table (MVT) is not outsourced with the file copies (*i.e.*, it is locally stored by the verifier). The MVT is used during the verification process to validate the completeness and consistency of all file copies. The MVT is implemented as a linked list of nodes, where each node contains 3 integers: *SN* (serial number), *BN* (block number), and *BV* (block version). 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 $F = \{b_j\}_{1\le j\le 8}$. Initially (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 <4, 9, 1> has been inserted in the MVT after SN_3 due to inserting a new data block after position 3 in the file copies $\tilde{\mathbb{F}}$. In the table entry <4, 9, 1>, the first parameter indicates the physical index of the new block, the second parameter is the logical block index (computed as MAX (*BN*) + 1), and the third parameter is the block version. Figure 4d shows the changes in the MVT due to deleting b_2 (the entry at index 2 will be deleted).

In [21], for a file $F = \{b_1, b_2, \ldots b_m\}$, the data owner generates *n* distinct copies $\tilde{\mathbb{F}} = \{\tilde{F}_1, \tilde{F}_2, \ldots, \tilde{F}_n\}$, 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 $\{\tilde{b}_{ij1}, \tilde{b}_{ij2}, \ldots, \tilde{b}_{ijs}\}$. Thus, the file copy $\tilde{F}_i = \{\tilde{b}_{ijk}\}(i:1\rightarrow n, j:1\rightarrow m, \text{and } k\,1\rightarrow s)$, where

each sector $\tilde{b}_{ijk} \in \mathbb{Z}_p$ for some large prime p.

A tag σ_{ij} is generated and associated with each block $\tilde{b}_{ij}: \sigma_{ij} = \left(H\left(ID_F \| BN_j \| BV_j\right) \cdot \prod_{k=1}^{S} u_k^{\tilde{b}_{ijk}}\right)^x \in \mathbb{G}_1 \, (i: 1 \to n, \, j: 1 \to m, \text{and } k1 \to s)$. 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 $\tilde{F}$ as $\sigma_j = \prod_{i=1}^{n} \sigma_{ij} \in \mathbb{G}_1$. 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. Figure 5 shows the procedure of the verification process of the MB-PMDDP scheme [21].

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 $\{y = g^x \in \mathbb{G}_2, (u_1, u_2, \ldots, u_s) \in_R \mathbb{G}_1\}$

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

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

Data Owner

– Generate n distinct copies $\tilde{\mathbb{F}} = \{\tilde{F}_1, \tilde{F}_2, \ldots, \tilde{F}_n\}$, 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 $\{\tilde{\mathbb{F}}, \Phi, ID_F\}$ 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 $\mu = \{\mu_{ik}\}(i: 1 \to n, k: 1 \to s)$

← σ, μ $\quad r_j \cdot \tilde{b}_{ijk} \in \mathbb{Z}_p$ over $(j,r_j) \in Q$

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

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

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 structure in which the hash values of the data blocks are stored in the leaf nodes. Figure 6a shows a hash tree for a file $F = \{b_j\}_{(1 \le j \le 8)}$, a cryptographic hash function h is used to compute the hash values for the data blocks. 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 blocks $\{b_j\}_{(1 \le j \le 8)}$ are outsourced to be stored on remote sites and h_R is locally stored. To validate the integrity of the blocks b_2 and b_6, the server sends

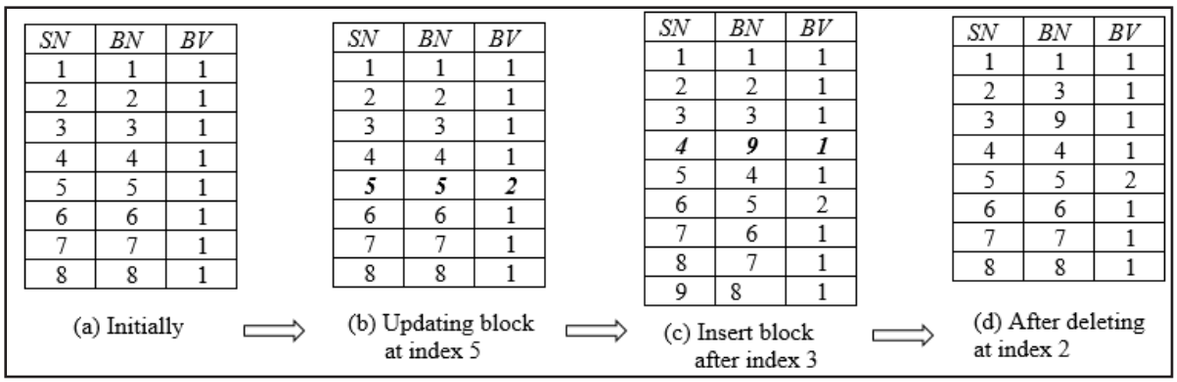

Fig. 4. Changes in the MVT Due to Different Dynamic Operations on Copies of a File F = $\{b_j\}_{(1 \le j \le 8)}$

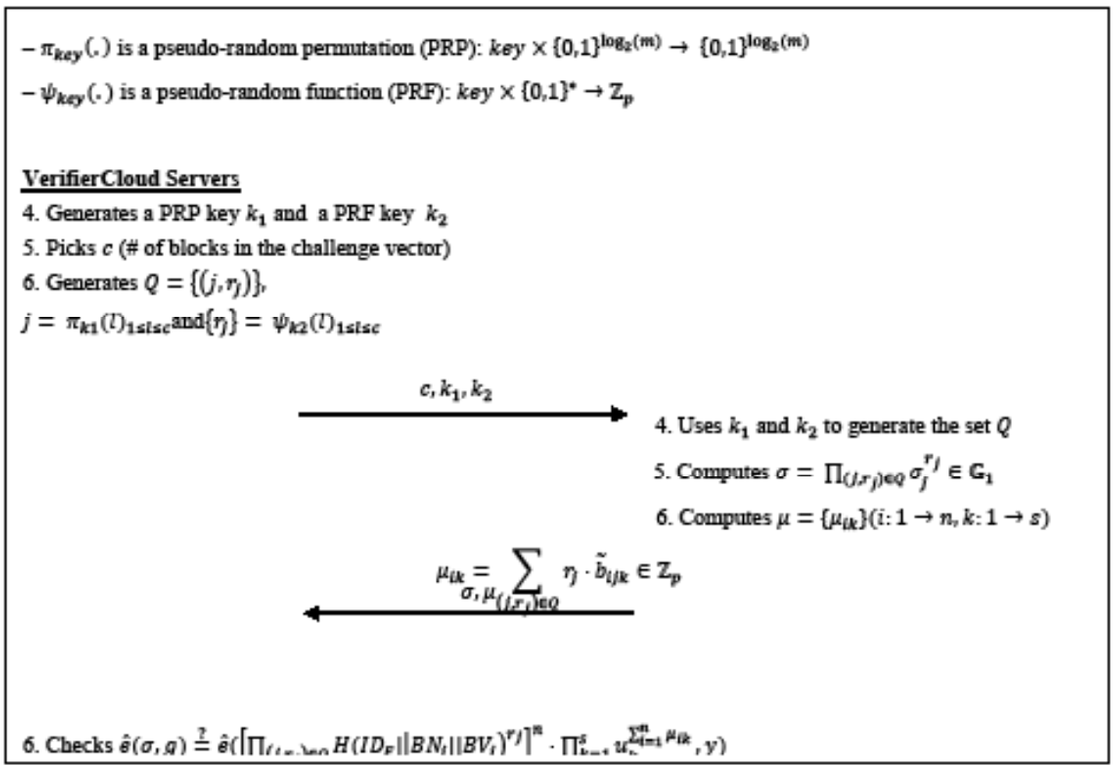

Fig. 5. The Verification Procedure of the MB-PMDDP scheme [21]

, a cryptographic hash function h is used to compute the hash values for the data blocks.

In the first level, the hash. At upper levels, and so on. The root is the hash of the root node that is used to validate the integrity of all data blocks. The blocks are outsourced to be stored on remote sites and is locally stored. To validate the integrity of the blocks and, the server sends back, where the set is used to generate the root of the hash tree. The verifier regenerate the root hash value using the received blocks and the authentication paths. The verifier computes and the verifier compares the authentic root value (), which is stored locally with the value that is computed .

The MHT is commonly used to authenticate the values of the data blocks. In case of dynamic data, both the *values* and the *positions* of the outsourced data blocks are used to validate the integrity. 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, $h(intrenal\ node) = h(LN||RN)$, where LN is the left node and LN is the right node. Note for example that $h_A = h(h_1||h_2) \neq h(h_2||h_1)$. Moreover, there is a *specific order* for the nodes in the authentication path.

For the TB-PMDDP scheme [21], two levels of MHTs are created. In the bottom level a MHT for each data copy is generated and the root of each tree is used to construct a MHT in the upper level, 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 needs to store only the value $\mathcal{M} = h(ID_F||h_{DR})$, where the first parameter is the file ID, and h_{DR} is the root value for the directory MHT.

Probabilistic Encryption-based Scheme

Mukundan*et al*. [24] proposed a scheme (which we will refer to as DMR-PDP) that handle dynamic data replicated on multiple sites. Their scheme is based on utilizing Paillieren cryption 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 $F_i = \left\{(1+N)^{b_j}(k_i r_{i,j})^N\right\}_{j=1}^{m}$, 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 $\sigma_j = (h(F).u^{b_j.N})^d$, 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:$E(\Delta b_j) = (1+\Delta b_j N)r^N$, and generates a new tag for the modified block b_j' , as $\sigma_j' = \left(h(F).u^{b_j'.N}\right)^d$. Then, the data owner sends to the CSP $E(\Delta b_j), \sigma_j'$, 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.

4. 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 MR-PDP and PB-PMDP models consider data that are subject to infrequent changes (static). In this analysis, we consider a 128-bit security level for the comparison. Thus, the used elliptic curve will be over $\mathbb{F}_p$ (p is of is of size 256 bits), h is of size 256 bits, and 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 block, respectively. We use the symbols c and $|F|$ to indicate the number of blocks in the challenge vector and the size of one data copy, respectively. For a 128-bit security level, the PRP and PRF need keys of size 128 bits. Table 1 shows a comparison among the presented schemes from

different perspectives: setup, storage, communication, computation, and dynamic operations costs.

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	$E_K + nm\ R + nm\mathcal{A}_{\mathbb{Z}}$	nE_K	nmE_K	nmE_K	nmE_K
	Tag Generation	$2m\mathcal{E}_{\mathbb{Z}_N} + m\mathcal{M}_{\mathbb{Z}} + m\mathcal{H}_{QR_N}$	$(s+1)nm\mathcal{E}_{\mathbb{G}} + nm\mathcal{H}_{\mathbb{G}} + (ns+n-1)m\mathcal{M}_{\mathbb{G}}$	$(s+1)nm\mathcal{E}_{\mathbb{G}} + nm\mathcal{H}_{\mathbb{G}} + (sn+n-1)m\mathcal{M}_{\mathbb{G}}$	$(s+1)nm\mathcal{E}_{\mathbb{G}} + nm(sn+n-1)m\lambda$	$nm(\mathcal{H}_{\mathbb{G}} + \mathcal{M}_{\mathbb{G}}) + nm(2\mathcal{E}_{\mathbb{G}} + \mathcal{M}_{\mathbb{Z}_p})$
	Metadata Generation	—	—	—	$nm\mathcal{H}_{\mathbb{G}} + (2m+1)$	—
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	—	—	64 m 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	$(2n+c+1)\mathcal{E}_{\mathbb{Z}_N} c\mathcal{H}_{QR_N} + \mathcal{D}_{\mathbb{Z}} + (cn+c+n-1)\mathcal{M}_{\mathbb{Z}} + (c-1)n\mathcal{A}_{\mathbb{Z}}$	$2\mathcal{P} + (c+s+1)\mathcal{E}_{\mathbb{G}} + c\mathcal{H}_{\mathbb{G}} + (c+s-1)\mathcal{M}_{\mathbb{G}} + (n-1)s\mathcal{A}_{\mathbb{Z}_p}$	$2\mathcal{P} + (c+s+1)\mathcal{E}_{\mathbb{G}} + c\mathcal{H}_{\mathbb{G}} + (c+s-1)\mathcal{M}_{\mathbb{G}} + (n-1)s\mathcal{A}_{\mathbb{Z}_p}$	$2\mathcal{P} + (c+s)\mathcal{E}_{\mathbb{G}} + cn\mathcal{H}_{\mathbb{G}} + (cn+s-1)\mathcal{M}_{\mathbb{G}} + (n-1)s\mathcal{A}_{\mathbb{Z}_p} + (\text{clog}_2(m)+2)nh_{SHA}$‡	$(2\mathcal{M}_{\mathbb{Z}_p} + \mathcal{E}_{\mathbb{G}})n + n\mathcal{A}_{\mathbb{Z}_p} + D_K + 3\mathcal{E}_{\mathbb{G}} + \mathcal{H}_{\mathbb{G}} + \mathcal{M}_{\mathbb{Z}_p}$
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 + 256 bits using hashing.*

‡*is the upper bound of the authentication path length when >1.*

5. Proof of Retrievability

Proof of retrievability (POR)-like PDP-is a technique to audit data over remote sites. But, POR has the advantage that it can help to *reconstruct* the outsourced data file using the responses sent from the remote servers. The fact that POR schemes *encode* the data file before outsourcing to remote sites allows more robustness. Thus, if it is required to find out if any changes occurred to minor parts of the outsourced data, then encoding (before sending the files to remote sites) should be applied.

Replication and encoding schemes are the most two common techniques used to achieve data redundancy. The storage cost to make copies of a file of size bits is bits. Using erasure code schemes, the file is fragmented into blocks, encoded into blocks () [26], and stored on different servers (one code block per server). Thus, the storage cost is bits. Any out of the 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. It is not worthy for CPSs to lose reputation (while gaining almost zero extra dollars) by deleting minor portions of the outsourced data copies. Using erasure codes to achieve redundancy has less storage cost; however, duplicating data file across multiple servers achieves scalability, which is a crucial need for cloud customers. The coding-based schemes have two sorts of

overheads, the first one is in the computations required to apply the encoding schemes before outsourcing. The second overhead is in time required to access a subset of servers to recover the original file.

Schwartz *et al.* [27] proposed a scheme to validate data integrity on multiple remote servers. In [27] keyed algebraic encoding and symmetric encryption can be used to detect any data damage. In [27], the communication cost is questionable (linear complexity). In addition, the security of their model is not proven and remains in question [7].

Juels*et al*. [28] is from the earliest POR models. In [28], the data file is first encrypted then masked with special blocks (guards). The guard blocks are concealed inside the original file to find out any misbehavior from the server side. During the verification phase, the verifier asks for randomly selected sentinels and checks whether they are modified or not. The guard blocks will be affected by any misbehavior/damage done by the server. The model in [28] does not allow for unlimited number of verifications (this is specified by the number of guard blocks). The number of guard blocks limits the number of verifications for with each verification one of the guard blocks will be known to the server and cannot be reused.

Shacham *et al.* [19] introduced a POR scheme that allows for unlimited number of verifications. In their scheme, they proposed to generate homomorphic authenticators, and thus the server can combine the data blocks and their tags and send a short response to the verifier. Shacham *et al.* [19] constructed two homomorphic authenticators using pseudorandom function and the digital signature of [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. The proposed scheme in [30] can guarantee the possession and retrievability of data. Wang *et al*. [31] proposed a distributed storage system that enables auditing remote data with recued computations. The scheme in [31] utilizes both homomorphic authenticators and erasure coding. Other erasure coding-based schemes include [32-34]. The schemes based on erasure coding suffer from high communication costs. To reduce the communication overhead, some proposals use network coding to develop POR schemes [35-37].

6. 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. In this work we also examine the basics behind the design of various PMDP protocols and underline some of the limitations. The paper also considered PMDP protocols for *static*and *dynamic* data. PMDP models for dynamic data support operations like modification, insertion, deletion, and append. The models for dynamic data provide an evidence to the verifier that all outsourced copies are complete and up-to-date (*i.e.*, the CSP did not ignore any of the update requests sent from 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.

Moreover, the paper addressed the proof of retrievability techniques. POR schemes *encode* the data file before outsourcing to remote sites to allow more robustness.

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. To protect against some great damages in one location, data need to be replicated across different areas

(*e.g.*, different states/courtiers). In addition, this will help to speed up the data access for different users in different countries all over the world. For this type of work, we need a group of researchers from both academia and industry to put the proof-of-concept models and to build the data centers.

- **Self-organized data replication over cloud servers.** Overloading the data centers with so many access requests from different locations can be a main reason for a failure of such data centers. The servers may not be available if the number of access requests increased in an unexpected manner.

  One of the future directions is to design storage schemes that can be automatically managed (self-managed). These schemes should be able to adapt themselves by reserving (or releasing) storage space on the cloud for data copies. To specify what should be the optimal number of data copies (and which servers to be located on) according to the current circumstances, an optimization framework needs to be designed. This optimization framework should speed up the response to the data retrieval requests and minimize the storage overhead on the cloud infrastructure.

- **User authentication for cloud computing systems.** In cloud computing environments, clients do not need powerful devices at their sides for they are using the computing power of the cloud servers, and thus users tend to use devices like PDA or cell phones. Data and applications will be available on the cloud, and the CSP needs to authenticate the users to allow them to access and use the cloud resources. In this model of computation, the monitoring is done in a centralized fashion and software piracy is a difficult task. But, for this computing paradigm, a secure authentication scheme is needed to protect against unauthorized use of cloud resources.

  Many traditional systems use passwords as the basic way of authentication, but this is not an effective method especially when we are dealing with sensitive outsourced data. Many passwords (which are not very strong) can be compromised using dictionary attacks. One of the guidelines for users is to use long passwords, but unfortunately it is a challenging task to memorize such long passwords and users prefer to use short and meaningful passwords, which will be subject to dictionary attacks.

  One of the techniques that can address the authentication problem in the cloud is implicit authentication. Information needs to be gathered about the user behavior for a long period. This sort of information will be used to construct a user model using learning algorithms. For a user to be authenticated, his current actions will be compared against the developed model to decide if that user is an eligible one.

- **Outsourcing computation to untrusted cloud servers.** It is not only data that can be outsourced to the cloud but also computations. Currently, many applications that require heavy computations (e.g., big data analysis) are outsourced to the cloud to benefit from the computational power of the cloud servers. The users need to make sure that their computations are *correctly* and *completely* performed, and the CSP is not cheating by executing just parts of them. The CSP may be economically motivated to save its computation resources by ignoring some of the heavy computations or just execute parts of them.

  Validating the completeness and correctness of the outsourced computation is a key requirement for cloud users. Moreover, it is crucial for the verification procedure to be much lighter (from the computation perspective) than performing the actual computation on the local side. The domain of verifiable computations is a hot research area. To complete the image, it will be interesting to address the mutual trust issue when computations are outsourced. There should be schemes to detect a dishonest party (a user or a CSP) who is trying to falsely accuse the other side. Thus, if the CSP does not completely and correctly perform the computations, this should be detected, and also a dishonest client should not be able to falsely claim that the computations are malformed.

References

1. Mell, P., & Grance, T. (2009). *Draft NIST working definition of cloud computing*. Retrieved from http://csrc.nist.gov/groups/SNS/cloudcomputing/index.html.
2. Wang, W., Li, Z., Owens, R., & Bhargava, B. (2009). *Secure and efficient access to outsourced data.* In Proceedings of the 2009 ACM workshop on Cloud computing security, (pp. 55-66).

3. Xie, M., Wang, H., Yin, J., & Meng, X. (2007). *Integrity auditing of outsourced data.* In VLDB '07 Proceedings of the 33rd International Conference on Very Large Databases, (pp. 782-793).
4. Gohring, N. (2008). Amazon's S3 down for several hours. Retrieved from http://www.pcworld.com/businesscenter/article/142549/amazons s3 down for severalhours.html.
5. Krebs, B. (2009). *Payment processor breach may be largest ever.* Retrieved from http://voices washingtonpost.com/securityfix/2009/01/payment processor breach may b.html
6. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., & Song, D. (2007). *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. Zeng, K. (2008). *Publicly verifiable remote data integrity.* In Proceedings of the 10th International Conference on Information and Communications Security, ser. ICICS '08. Berlin, Heidelberg: Springer-Verlag, (pp. 419-434).
8. Deswarte, Y., Quisquater, J. J., & Saıdane, A. (2003). *Remote integrity checking*. In 6th Working Conference on Integrity and Internal Control in Information Systems (IICIS), (pp. 1-11).
9. Filho, D. L. G., & Barreto, P. S. L. M. (2006). Demonstrating data possession and uncheatable data transfer. Cryptology e-Print Archive, Report 2006/150.
10. Sebe, F., Domingo-Ferrer, J., Martinez-Balleste, A., Deswarte, Y., & Quisquater, J. J. (2008). *Efficient remote data possession checking in critical information infrastructures*. IEEE Transactions on Knowledge and Data Engineering, 20(8), 1034-1038.
11. Golle, P., Jarecki, S., & Mironov, I. (2003). *Cryptographic primitives enforcing communication and storage complexity*. In FC'02: Proceedings of the 6th International Conference on Financial Cryptography, Berlin, Heidelberg, (pp. 120-135).
12. Shah, M. A., Baker, M., Mogul, J. C., & Swaminathan, R. (2007). *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, (pp. 1-6).
13. Shah, M. A., Swaminathan, R., & Baker, M. (2008). *Privacy-preserving audit and extraction of digital contents*. Cryptology e-Print Archive, Report 2008/186.
14. Mykletun, E., Narasimha, M., & Tsudik, G. (2006). Authentication and integrity in outsourced databases. *ACM Transactions on Computational Logic*, 2(2), 122-128.
15. Amazon Simple Storage Service (Amazon S3). Retrieved from http://aws.amazon.com/s3/.
16. Barsoum, A. F., & Hasan, M. A. (2010). *Provable possession and replication of data over cloud servers.* Centre for Applied Cryptographic Research (CACR), University of Waterloo, Report 2010/32. Retrieved from http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-32.pdf.
17. Curtmola, R., Khan, O., Burns, R., & Ateniese, G. (2008). *MR-PDP: Multiple-replica provable data possession*. In 28th IEEE International Conference on Distributed Computing Systems, ICDCS, (pp. 411-420).
18. Barsoum, A. F., & Hasan, M. A. (2012). *Integrity verification of multiple data copies over untrusted cloud servers*. In 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CC Grid 2012. IEEE Computer Society, (pp. 829-834).
19. Shacham, H., & Waters, B. (2008). *Compact proofs of retrievability*. In Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, (pp. 90-107).
20. Ferrara, A. L., Green, M., Hohenberger, S., & Pedersen, M. (2009). *Practical short signature batch verification.* In The Cryptographer's Track at RSA Conference, (pp. 309-324).
21. Barsoum, A., & Hasan, M. (2015). *Provable multicopy dynamic data possession in cloud computing systems*. IEEE Transactions on Information Forensics and Security, March, 10(3), 485-497.
22. Merkle, R. C. (1980). *Protocols for public key cryptosystems*. IEEE Symposium on Security and Privacy, (pp. 122).
23. Martel, C., Nuckolls, G., Devanbu, P., Gertz, M., Kwong, A., & Stubblebine, S. G. (2001). A general model for authenticated data structures," *Algorithmica*, 39(1), 21-41.
24. Mukundan, R., Madria, S., Linderman, M., & Rome, N. (2012). *Replicated data integrity verification in cloud*. IEEE Data Engineering Bulletin, 35(4), 55-64.
25. Barreto, P. S. L. M., & Naehrig, M. (2006). *IEEE P1363.3 submission: Pairing-friendly elliptic curves*

of prime order with embedding degree 12, New Jersey: IEEE Standards Association.

26. Aguilera, M. K., Janakiraman, R., & Xu, L. (2005). *Using erasure codes efficiently for storage in a distributed system.* In Proceedings of the 2005 International Conference on Dependable Systems and Networks, (pp. 336-345).
27. Schwarz, T. S. J., & Miller, E. L. (2006). *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.
28. Juels, A., & Kaliski, B. S. (2007). *PORs: Proofs of Retrievability for large file*. In Proceedings of the 14th ACM Conference on Computer and Communications Security, (pp. 584-597).
29. Boneh, D., Lynn, B., & Shacham, H. (2001). *Short signatures from the Weil pairing*. In Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security, London, UK, (pp. 514-532).
30. Bowers, K. D., Juels, A., & Oprea, A. (2009). *HAIL: A High-Availability and Integrity Layer for Cloud Storage.* Proceedings of the 16th ACM conference on Computer and communications security. New York, NY, USA, (pp. 187-198).
31. Wang, C., Wang, Q., Ren, K., Cao, N., & Lou, W. (2012). *Toward secure and dependable storage services in cloud computing*. IEEE Transactions on Services Computing, April, 5(2), 220-232.
32. Curtmola, R., Khan, O., & Burns, R. (2008). *Robust remote data checking.* In Proceedings of the 4th ACM international workshop on Storage security and survivability. New York, NY, USA, (pp. 63-68).
33. Bowers, K. D., Juels, A., & Oprea, A. (2009). *Proofs of retrievability: Theory and implementation*. In Proceedings of the 2009 ACM workshop on Cloud computing security. New York, NY, USA, (pp. 43-54).
34. Dodis, Y., Vadhan, S., & Wichs, D. (2009). *Proofs of retrievability via hardness amplification.* In Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography. Berlin, Heidelberg: Springer-Verlag, (pp. 109-127).
35. Chen, B., Curtmola, R., Ateniese, G., & Burns, R. (2010). *Remote data checking for network coding-based distributed storage systems*. In Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop, New York, NY, USA, (pp. 31-42).
36. Anh, L., & Markopoulou, A. (2012). *Nc-audit: Auditing for network coding storage*. In International Symposium on Network Coding (NetCod), (pp. 155-160).
37. Chen, H. C. H., & Lee, P. P. C. (2014). Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation. *IEEE Transactions on Parallel Distributed Syst*ems, February, 25(2), 407-416.
38. Barsoum, A. F., & Hasan, M. A. (2014). Verifying outsourced replicated data in cloud computing storage systems. *International Journal of Computer Applications*, August, 99(7), 1-13.