Data Security and Integrity Verification in Multi-Cloud Storage Using Cooperative Provable Data Possession Technique

G. Mahammad Ghouse
M.Tech (CSE) Intel Engineering College,
Anantapur, AP, INDIA

K. Janaradhan M.Tech
Associate Professor, Department of CSE,
Intel Engineering College,
Anantapur, AP, INDIA

Abstract

Cloud computing has become a faster profit growth point in recent years by providing a comparably low-cost, scalable, position-independent platform for data outsourcing.

Multi clouds like hybrid clouds offer the cost and scale benefits of public clouds while also offering the security and control of private clouds. Benefits of hybrid clouds includes cost saving by improving resource allocation to the projects optimizing the infrastructure spreading during different stages of application lifecycle and business agility by offering the controls present in public and private clouds providing drastic improvements in the overall organizational agility.

In this paper, we address the construction of an efficient PDP scheme to cooperatively store and maintain the data in multi-cloud storage using PDP technique.

1. Introduction

Storage outsourcing in clouds has become a new profit growth point by providing a comparably low-cost, scalable, location-independent platform for managing clients' data. However, security is critical for such convenient storage services due to the following reasons: the cloud infrastructures are much more powerful and reliable than personal computing devices but they are still facing all kinds of internal and external threats; for the benefits of their own business, there exist various motivations for cloud service providers to behave unfaithfully towards the cloud users; and, furthermore, the dispute occasionally suffers from a lack of trust on CSPs. Consequently, the behaviors of CSPs may not be known by the cloud users, even if this dispute may result from the users' own improper operations. Therefore, it is crucial for a CSP to offer an efficient verification on the integrity and availability of the stored data to enable the credibility of cloud services. We expect that the size of outsourced data cannot be too small to influence the verification efficiency. All outsourced data would require additional storages for the verification parameters which must be stored in a Trusted Third Party (TTP). Thus, from a practical standpoint, the outsourced data can be either a large file, a database, or a set of files in an application system including software's, scripts, Web pages, snapshots, and so on. Especially, it is critical to check the integrity of application software's in public clouds even if sensitive data are stored in private clouds. For instance, an attacker can modify application software's or scripts, or load a trojan into a snapshot of virtual machine (VM) to compromise the applications in a cloud.

The traditional cryptographic technologies for data integrity and availability, based on Hash functions and signature schemes, cannot support the outsourced data without a local copy of data. It is evidently impractical for a cloud storage service to download the whole data for data validation due to the expensiveness of communication, especially, for large-size files. Recently, several PDP schemes are proposed to address this issue. In fact, PDP is essentially an interactive proof between a CSP and a client because the client makes a false/true decision for data possession without downloading data.

Existing PDP schemes mainly focus on integrity verification issues at untrusted stores in public clouds, but these schemes are not suitable for a hybrid cloud environment since they were originally constructed based on a two-party interactive proof system. For a hybrid cloud, these schemes can only be used in a trivial way: clients must invoke them repeatedly to check the integrity of data stored in each single cloud. This means that clients must know the exact position of each data block in outsourced data. Moreover, this process will consume higher communication bandwidth and computation costs at client sides. Thus, it is of utmost necessary to construct an efficient verification scheme with collaborative features for hybrid clouds. In response to practical requirements for outsourced storages in hybrid clouds, the concerns to improve the performance of PDP services are mainly from three aspects:

- How to design a more efficient PDP model for hybrid clouds to reduce the storage and network

overheads and enhance the transparency of verification activities;

- How to provide an efficient sampling policy to help provide a more cost-effective verification service; and

- How to optimize the parameters of PDP scheme to minimize the computation overheads of verification services in hybrid clouds.

Solving these problems will help improve the quality of PDP services, which can not only timely detect anomalies, but also take up less resource, or rationally allocate resources. Hence, a new PDP scheme is desirable to accommodate these application requirements from hybrid clouds.

2. Related Work

The traditional cryptographic technologies for data integrity and availability, based on Hash functions and signature schemes, cannot work on the outsourced data without a local copy of data. In addition, it is not a practical solution for data validation by downloading them due to the expensive communications, especially for large-size files. Moreover, the ability to audit the correctness of the data in a cloud environment can be formidable and expensive for the cloud users. Therefore, it is crucial to realize public auditability for CSS, so that data owners may re- sort to a third party auditor (TPA), who has expertise and capabilities that a common user does not have, for periodically auditing the outsourced data. This audit service is significantly important for digital forensics and credibility in clouds. To implement public auditability, the notions of proof of retrievability (POR) and provable data possession (PDP) have been proposed by some researchers. Their approach was based on a probabilistic proof technique for a storage provider to prove that clients' data remain intact. For ease of use, some POR/PDP schemes work on a publicly verifiable way, so that anyone can use the verification protocol to prove the availability of the stored data. Hence, this provides us an effective approach to accommodate the requirements from public auditability. POR/PDP schemes evolved around an untrusted storage offer a publicly accessible re-mote interface to check the tremendous amount of data. There exist some solutions for audit services on outsourced data. For example, Xie et al. proposed an efficient method on content comparability for outsourced database, but it wasn't suited for irregular data. Wang et al. also provided a similar architecture for public audit services. To support their architecture, a public audit scheme was proposed with privacy-preserving property. However, lack of rigorous performance analysis for constructed audit system greatly affect the practical application of this scheme. For instance, in this scheme an outsourced file is directly split into n blocks, and then each block generates a verification tag. In order to maintain security, the length of block must be equal to the size of cryptosystem, that is, 160-bit=20-Bytes. This means that 1M-Bytes file is split into 50,000 blocks and generates 50,000 tags, and the storage of tags is at least 1M-Bytes. It is clearly inefficient to build an audit system based on this scheme. To address such a problem, a fragment technique is introduced in this paper to improve performance and reduce extra storage. Another major concern is the security issue of dynamic data operations for public audit services. In clouds, one of the core design principles is to provide dynamic scalability for various applications. This means that remotely stored data might be not only accessed but also dynamically updated by the clients, for instance, through block operations such as modification, deletion and insertion. However, these operations may raise security issues in most of existing schemes, e.g., the forgery of the verification metadata (called as tags) generated by data owners and the leakage of the user's secret key. Hence, it is crucial to develop a more efficient and secure mechanism for dynamic audit services, in which possible adversary advantage through dynamic data operations should be prohibits. Note that this paper only addresses the problems of integrity checking and auditing. Other security services, such as user authentication and data encryption, are orthogonal to and compatible with audit services.

3. PDP Structure and Techniques

In this section, we study the verification framework for hybrid cloud storage and of CPDP and also studied two basic techniques for constructing our CPDP scheme: hash index hierarchy (HIH) and homomorphic verifiable response (HVR).

3.1 Multi-cloud Framework

According to our study it was clear that existing PDP schemes offer a publicly accessible remote interface for checking and managing the tremendous amount of data and majority of existing PDP schemes are incapable to satisfy the inherent requirements from hybrid clouds in terms of communication and computation costs. To address this problem, we consider hybrid cloud storage service as illustrated in Figure 1. In this

architecture three different entities exists for data storage.

1. Clients can access and manipulate data which is stored in hybrid clouds with certain permissions.
2. Cloud Service Providers (CSPs) works together to provide data storage services with sufficient storages and computation resources; and
3. Trusted Third Party (TTP) who is trusted to store verification parameters and offer public query services.

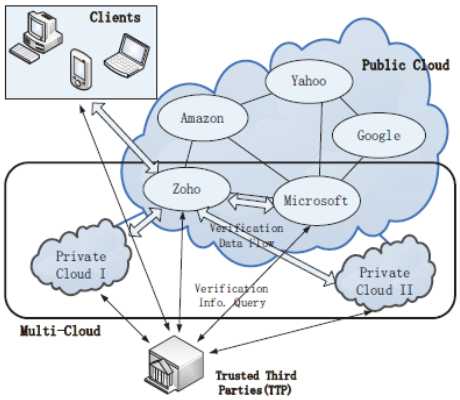

Figure 1: Data Integrity Verification Architecture

In this architecture, we identified the existence of multiple CSPs which cooperatively store and maintain the clients' data. However, a cooperative PDP is used to verify the integrity and availability of their stored data in all CSPs. Firstly we verify, a client (data owner) uses the secret key to pre-process a file which consists of a collection of n blocks and generates a set of public verification information which is stored in TTP followed by transmitting the file with some verification tags to CSPs, leading to the deletion of local copy; Secondly, by using a verification protocol, the clients can issue a challenge to check the integrity and availability of outsourced data.

We never consider whether the CSP guarantees the security of the stored data, or assume that data owner has the ability to collect the evidence of the CSP's fault after finding the errors. In order to achieve, a TTP server is constructed as a core trust base on the cloud for the sake of security. We identified the assumption that the TTP is reliable and independent to setup and maintain the CPDP cryptosystem, generate and store data owner's public key and to store the public parameters used to execute the verification protocol in the CPDP scheme. We also identified that the TTP is not directly involved in the CPDP scheme to reduce the cryptosystem complexity.

3.2 Cooperative PDP

The integrity of data stored in a multi-cloud environment can be defined by a framework for CPDP based on interactive proof system (IPS) and multi-prover zero-knowledge proof system (MP-ZKPS), as follows:

Definition 1 (Cooperative-PDP): A cooperative provable data possession $\mathcal{S} = (KeyGen, TagGen, Proof)$ is a collection of two algorithms $(KeyGen, TagGen)$ and an interactive proof system $Proof$, as follows: $KeyGen(1\kappa)$: takes a security parameter κ as input, and returns a secret key sk or a public-secret key- pair (pk, sk); $TagGen$ $(sk, F, \mathcal{P})$: takes as inputs a secret key sk,a file F, and a set of cloud storage providers $\mathcal{P} = \{Pk\}$, and returns the triples (ζ, ψ, σ), whereζ is the secret in tags, $\psi = (u, \mathcal{H})$ is a set of verification parameters u and an index hierarchy $\mathcal{H}$ for F, $\sigma = \{\sigma(k)\}Pk\in\mathcal{P}$ denotes a set of all tags, $\sigma(k)$ is the tag of the fraction $F(k)$ of F in Pk;

$Proof(\mathcal{P}, V)$: is a protocol of proof of data possession between CSPs ($\mathcal{P} = \{Pk\}$) and a verifier (V), that is,

$$\langle \sum Pk\in\mathcal{P}\ Pk(F(k),\sigma(k)) \longleftrightarrow V\rangle(pk,\psi) = \begin{cases} 1 & F = \{F(k)\} \text{ is intact} \\ 0 & F = \{F(k)\} \text{ is changed} \end{cases},$$

where each Pk takes as input a file $F(k)$ and a set of tags $\sigma(k)$, and a public key pk and a set of public parameters ψ are the common input between P and V . At the end of the protocol run, V returns a bit$\{0|1\}$ denoting false and true. Where,$\sum Pk\in\mathcal{P}$ denotes cooperative computing in $Pk \in\mathcal{P}$. A trivial way to realize the CPDP is to check the data stored in each cloud one by one, i.e.,

$$\wedge P\ k\in\mathcal{P}\ \langle Pk(F(k),\sigma(k)) \longleftrightarrow V\ \rangle(pk,\psi),$$

Where $\wedge$ denotes the logical AND operations among the Boolean outputs of all protocols $\langle Pk, V\rangle$ for all $Pk \in\mathcal{P}$. However, it would cause significant communication and computation overheads for the verifier, as well as a loss of location-transparent. Such a primitive approach obviously diminishes the advantages of cloud storage: scaling arbitrarily up and down on-demand. To solve this problem, we extend above definition by adding an organizer(O), which is one of CSPs that directly contacts with the verifier, as follows:

$$\langle \sum Pk\in\mathcal{P}\ Pk(F(k),\sigma(k)) \longleftrightarrow O \longleftrightarrow V\rangle(pk,\psi),$$

where the action of organizer is to initiate and organize the verification process. This definition is consistent with aforementioned architecture, e.g., a client (or an authorized application) is considered

as V, the CSPs are as $\mathcal{P} = \{Pi\}$ $i\in[1,c]$, and the Zoho cloud is as the organizer in Figure 1. Often, the organizer is an independent server or a certain CSP in $\mathcal{P}$. The advantage of this new multi-prover proof system is that it does not make any difference for the clients between multi-prover verification process and single- prover verification process in the way of collaboration. Also, this kind of transparent verification is able to conceal the details of data storage to reduce the burden on clients. For the sake of clarity, we list some used signals in Table 1.

Table 1: The Signal and its Explanation

Sig.	Repression
n	the number of blocks in a file;
s	the number of sectors in each block;
t	the number of index coefficient pairs in a query;
c	the number of clouds to store a file;
F	the file with $n \times s$ sectors, i.e., $F = \{m_{i,j}\}_{j\in[1,s]}^{i\in[1,n]}$;
σ	the set of tags, i.e., $\sigma = \{\sigma_i\}_{i\in[1,n]}$;
Q	the set of index-coefficient pairs, i.e., $Q = \{(i, v_i)\}$;
θ	the response for the challenge Q.

3.3.Structural Representation of Cooperative PDP Scheme

An architecture representing the support of distributed cloud storage we use cooperative PDP scheme as shown in Figure 2. This architecture shows a structural representation which resembles a natural representation of file storage. This structure $\mathcal{H}$ consists of three layers to represent relationships among all blocks for stored resources. They are described as Express Layer offers an abstract representation of the stored resources, Service Layer offering and manages cloud storage services and Storage Layer: realizes data storage on many physical devices.

In storage layer, a common fragment structure provides probabilistic verification of data integrity for outsourced storage. The fragment structure is a data structure that maintains a set of block-tag pairs, allowing searches, checks and updates in $O(1)$ time. An instance of this structure is as outsourced file F splits into n blocks $\{m1, m2, \cdots, mn\}$, and each block mi is split into s sectors $\{mi, 1, mi, 2, \cdots, mi, s\}$. The fragment structure consists of n block-tag pair $(mi, \sigma\ i)$, where σi is a signature tag of block mi generated by a set of secrets $\tau = (\tau 1, \tau 2, \cdots, \tau s)$.

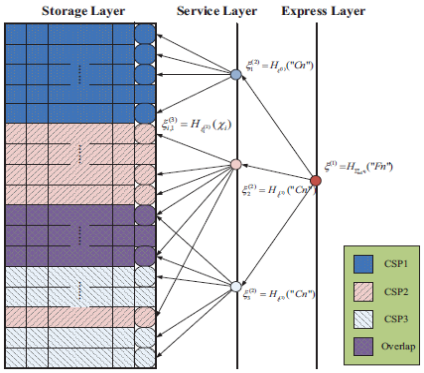

Figure 2: Index-hash hierarchy of CPDP model

Data integrity can be checked with a fragment structure by implementing probabilistic verification as a given random chosen challenge (or query) $Q = \{(i,\ v\ i)\}i\in RI$, where I is a subset of the block indices and vi is a random coefficient. An efficient algorithm exists to produce a constant-size response $(\mu 1,\ \mu 2, \cdots, \mu s, \sigma\ ')$, where μi comes from all $\{mk, i, vk\}k\in I$ and σ' is from all $\{\sigma k, vk\}k\in I$.

Given a collision-resistant hash function $Hk(\cdot)$, we make use of this architecture to construct a Hash Index Hierarchy $\mathcal{H}$ (viewed as a random oracle), which is used to replace the common hash function in prior PDP schemes, as Express layer with a given s random $\{\tau i\}s\ i$=1 and the file name Fn, sets $\xi(1) = H \sum s\ i$=1 $\tau i(Fn)$ and makes it public for verification but makes $\{\tau i\}s\ i$=1 secret; A Service layer given the $\xi(1)$ and the cloud name Ck, sets$\xi(2)\ k = H\xi(1)(Ck)$ and a Storage layer which gives the $\xi(2)$, a block number i, and its index record χi ="$Bi||Vi||Ri$", sets$\xi(3)\ i,k = H\xi(2)\ k\ (\chi i)$, where$Bi$ is the sequence number of a block, Vi is the updated version number, and Ri is a random integer to avoid collision. As a virtualization approach, we introduce a simple index-hash table $\chi = \{\chi i\}$ to record the changes of file blocks as well as to generate the hash value of each block in the verification process. The structure of χ is similar to the structure of file block allocation table in file systems. The index-hash table consists of serial number, block number, version number, random integer, and so on. Different from the common index table, we assure that all records in our index table differ from one another to prevent forgery of data blocks and tags. By using this structure, especially the index records $\{\chi i\}$, our CPDP scheme can also support dynamic data operations. The proposed structure can be readily incorporated into MAC-based, ECC or RSA schemes. These schemes, built from collision-resistance signatures and the random oracle model, have the shortest query and response with public verifiability. They share several common characters for the implementation of the CPDP framework in the multiple clouds:

A file is splits into $n\times s$ sectors corresponds each as a tag, so that the storage of signature tags can be reduced by increasing s, a verifier to verify the integrity of file in random sampling approach of large files. These schemes rely on homomorphic properties to aggregate data and tags into a constant size response, which minimizes the overhead of network communication and the hierarchy structure

to provide a virtualization approach to conceal the storage details of multiple CSPs.

3.4 Impact of RSA on Cloud Performance

In this section we carried out an experimental approach to determine how RSA impacts the performance of the cloud environment.

3.4.1. RSA Implementation in Multi Cloud

User data is encrypted first and then it is stored in the Cloud. When required, user places a request for the data for the Cloud provider, Cloud provider authenticates the user and delivers the data. RSA is a block cipher, in which every message is mapped to an integer. RSA consists of Public-Key and Private-Key. In our Cloud environment, Pubic-Key is known to all, whereas Private-Key is known only to the user who originally owns the data. Thus, encryption is done by the Cloud service provider and decryption is done by the Cloud user or consumer. Once the data is encrypted with the Public-Key, it can be decrypted with the corresponding Private-Key only. RSA algorithm involves three steps:

1. Key Generation
2. Encryption
3. Decryption

Key Generation: Before the data is encrypted, Key generation should be done. This process is done between the Cloud service provider and the user.

Steps:

1. Choose two distinct prime numbers a and b. For security purposes, the integers a and b should be chosen at random and should be of similar bit length.
2. Compute n = a * b.
3. Compute Euler's totient function, Ø(n) = (a-1) * (b-1).
4. Chose an integer e, such that 1 < e < Ø(n) and greatest common divisor of e , Ø(n) is 1. Now e is released as Public-Key exponent.
5. Now determine d as follows: d = e-1(mod Ø(n)) i.e., d is multiplicate inverse of e mod Ø(n).
6. d is kept as Private-Key component, so that d * e = 1 mod Ø(n).
7. The Public-Key consists of modulus n and the public exponent e i.e, (e, n). 8. The Private-Key consists of modulus n and the private exponent d, which must be kept secret i.e, (d, n).

Encryption: Encryption is the process of converting original plain text (data) into cipher text (data).

Steps:

1. Cloud service provider should give or transmit the Public- Key (n, e) to the user who want to store the data with him or her.
2. User data is now mapped to an integer by using an agreed upon reversible protocol, known as padding scheme.
3. Data is encrypted and the resultant cipher text(data) C is C = me (mod n).
4. This cipher text or encrypted data is now stored with the Cloud service provider.

Decryption: Decryption is the process of converting the cipher text(data) to the original plain text(data).

Steps:

1. The cloud user requests the Cloud service provider for the data.
2. Cloud service provider verify the authenticity of the user and gives the encrypted data i.e, C.
3. The Cloud user then decrypts the data by computing, m = Cd (mod n).
4. Once m is obtained, the user can get back the original data by reversing the padding scheme.

3.4.2 Experimental Results

In this section, we are taking some sample data end implementing RSA algorithm over it.

Key Generation:

1. We have chosen two distinct prime numbers a=61 and b=53.
2. Compute n=a*b, thus n=61*53 = 3233.
3. Compute Euler's totient function, Ø(n)=(a-1)*(b-1), Thus Ø(n)=(61-1)*(53-1) = 60*52 = 3120.
4. Chose any integer e, such that 1 < e < 3120 that is coprime to 3120. Here, we chose e=17.
5. Compute d, d = e-1(mod Ø(n)), thus d=17-1(mod 3120) = 2753.
6. Thus the Public-Key is (e, n) = (17, 3233) and the Private- Key is (d, n) = (2753, 3233).

This Private-Key is kept secret and it is known only to the user.

Encryption:

1. The Public-Key (17, 3233) is given by the Cloud service provider to the users who wish to store the data.
2. Let us consider that the user mapped the data to an integer m=65.

3. Data is encrypted now by the Cloud service provider by using the corresponding Public-Key

which is shared by both the Cloud service provider and the user. C = 6517(mod 3233) = 2790.

4. This encrypted data i.e, cipher text is now stored by the Cloud service provider.

Decryption:

1. When the user requests for the data, Cloud service provider will authenticate the user and delivers the encrypted data (If the user is valid).
2. The cloud user then decrypts the data by computing, m = Cd(mod n) = 27902753(mod 3233) = 65.
3. Once the m value is obtained, user will get back the original data.

First, from our experiments we found that the performance of CPDP scheme, especially for large files, is affected by the bilinear mapping operations due to its high complexity. To solve this problem, **RSA based constructions** may be a better choice, and quite efficient digital signature schemes whose security is based on the RSA assumption. This is the strongest type of security for a digital signature scheme that one can expect, and a signature scheme that is secure in this sense can be safely deployed in the widest possible range of applications.

But this is still a challenging task because the existing RSA based schemes have too many restrictions on the performance and security.

3.5. Performance Evaluation

In this section, to detect abnormality in a low-overhead and timely manner, we analyze and optimize the performance of CPDP scheme based on the above scheme from two aspects: evaluation of probabilistic queries and optimization of length of blocks.

- Performance Analysis for CPDP Scheme
- Probabilistic Verification
- Parameter Optimization
- CPDP for Integrity Audit Services

4. Future Enhancements

We would extend our work to explore more effective CPDP constructions. From a practical point of view, we still need to address some issues regarding the integration of CPDP scheme with existing systems. Matching of index hash hierarchy with HDFS's two-layer name space, index structure with cluster network model and how to dynamically update the CPDP parameters according to HDFS's requirement specifications are still need to be addressed. Finally, it is still need to face a challenging problem for the generation of tags with the length irrelevant to the size of data blocks.

5. Conclusion

In this paper, we presented the construction of an efficient PDP scheme for distributed cloud storage.

Based on homomorphic verifiable response and hash index hierarchy, we have proposed a cooperative PDP scheme to support dynamic scalability on multiple storage servers. We also showed that our scheme provided all security properties required by zero- knowledge interactive proof system, so that it can resist various attacks even if it is deployed as a public audit service in clouds. Our experiments clearly demonstrated that our approaches only introduce a small amount of computation and communication overheads. Therefore, our solution can be treated as a new candidate for data integrity verification in outsourcing data storage systems. As part of future work, we would extend our work to explore more effective CPDP constructions.

6. References

[1] Yan Zhu, Hongxin Hu, Gail-Joon Ahn, Yujing Han, Shimin Chen – "Collaborative Integrity Verification in Hybrid Clouds", Arizona State University Proceedings, p.p.1-10.

[2] Yan Zhu, Hongxin Hu, Gail-Joon Ahn, Senior Member, IEEE, Mengyang Yu – "Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage", IEEE Transactions on Parallel and Distributed Systems, p.p 1-14.

[3] Krishna Kumar Singh, Rajkumar Gaura, Sudhir Kumar Singh – "Cooperative provable data retention for integrity authentication in multi-cloud Storage", International Journal of Scientific & Engineering Research, Volume 4, Issue 1, January-2013.

[4] Yan Zhu, Huaixi Wang, Zexing Hu1, Gail-Joon Ahn, Hongxin Hu, Stephen S. Yau – "Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds", SAC'11 March 21-25, 2011, TaiChung, Taiwan, p.p.1-8.

[5] Yan Zhu and Shanbiao Wang – "Secure Collaborative Integrity Verification For Hybrid Cloud Environments", International Journal of Cooperative Information Systems Vol. 21, No. 3 (2012) 165–197.

[6] Parsi Kalpana, Sudha Singaraju – "Data Security in Cloud Computing using RSA Algorithm", International Journal of Research in Computer and Communication technology, IJRCCT, ISSN 2278-5841, Vol 1, Issue 4, September 2012, pp.no 145-146.