Provable Data Possession & Dynamic Audit Services for Integrity Verification in Clouds Environments

Qumarunnisa Begum  K.V Ramu  G Charles Babu  Dr Subhash Chandra  SP. Santhosh
M.Tech Student  Asst. Prof  HOD  Professor  M.Tech Student
HMITS  HMITS  HMITS  HMITS  HMITS

Abstract—Provable data possession (PDP) is a technique for ensuring the integrity of data in storage outsourcing. In this paper, we address the construction of an efficient PDP scheme for distributed cloud storage to support the scalability of service and data migration, in which we consider the existence of multiple cloud service providers to cooperatively store and maintain the clients’ data. We present a cooperative PDP (CPDP) scheme based on homomorphic verifiable response and hash index hierarchy. We prove the security of our scheme based on multi-prover zero-knowledge proof system, which can satisfy completeness, knowledge soundness, and zero-knowledge properties. In addition, we propose a dynamic audit service for verifying the integrity of untrusted and outsourced storage. Our audit service, constructed based on the techniques, fragment structure, random sampling and index-hash table, can support provable updates to outsourced data, and timely abnormal detection.

Index: Storage Security, Provable Data Possession, Dynamic Audit, Integrity Verification.

1. Introduction
Cloud computing provides a scalability environment for growing amounts of data and processes that work on various applications and services by means of on-demand self-service. One of the strength of cloud computing is that data are being centralized and outsourced in clouds. This kind of outsourced storage in clouds has become a new profit growth point by providing a comparably low-cost, scalable, location independent platform for managing clients’ data. The cloud storage service (CSS) relieves the burden for storage management and maintenance. However, if such an important service is vulnerable to attacks or failures, it would bring irretrievable losses to the clients since their data or archives are stored in an uncertain storage pool outside the enterprises. These security risks come from the following reasons: the cloud infrastructures are much more powerful and reliable than personal computing devices. However, they are still facing all kinds of internal and external threats; for the benefits of their possession, there exist various motivations for cloud service providers (CSP) to behave unfaithfully towards the cloud users; furthermore, the dispute occasionally suffers from a lack of trust on CSP. Consequently, their behaviors may not be known by the cloud users, even if this dispute may result from the users’ own improper operations. Therefore, it is necessary for cloud service providers to offer an efficient audit service to check the integrity and availability of the stored data [10]. Security audit is an important solution enabling tracking and analysis of any activities including data accesses, security breaches, application activities, and so on. Data security tracking is crucial for all organizations that must be able to comply with a range of federal laws including the Sarbanes-Oxley Act, Basel II, HIPAA and other regulations. Furthermore, compared to the common audit, the audit service for cloud storages should provide clients with a more efficient proof of the integrity of stored data. In this paper, we introduce a dynamic audit service for integrity verification of untrusted and outsourced storages. Our audit system, based on a novel audit system architecture, can support dynamic data operations and timely abnormal detection with the help of several effective techniques, such as fragment structure, random sampling, and index-hash table. Furthermore, we propose an efficient approach based on probabilistic query and periodic verification for...
improving the performance of audit services. A proof-of-concept prototype is also implemented to evaluate the feasibility and viability of our proposed approaches. Our experimental results not only validate the effectiveness of our approaches, but also show our system has a lower computation cost, as well as a shorter extra storage for integrity verification.

Provable data possession (PDP) [2] (or proofs of retrievability (POR) [3]) is such a probabilistic proof technique for a storage provider to prove the integrity and ownership of clients’ data without downloading data. The proof-checking without downloading makes it especially important for large-size files and folders (typically including many clients’ files) to check whether these data have been tampered with or deleted without downloading the latest version of data. Thus, it is able to replace traditional hash and signature functions in storage outsourcing. Various PDP schemes have been recently proposed, such as Scalable PDP [4] and Dynamic PDP [5]. However, these schemes mainly focus on PDP issues at untrusted servers in a single cloud storage provider and are not suitable for a multi-cloud environment. To provide a low-cost, scalable, location independent platform for managing clients’ data, current cloud storage systems adopt several new distributed file systems, for example, Apache Hadoop Distributed File System (HDFS), Google File System (GFS), Amazon S3 File System, Cloud Store etc. These file systems share some similar features: a single metadata server provides centralized management by a global namespace; files are split into blocks or chunks and stored on block servers; and the systems are comprised of interconnected clusters of block servers. Those features enable cloud service providers to store and process large amounts of data. However, it is crucial to offer an efficient verification on the integrity and availability of stored data for detecting faults and automatic recovery. Moreover, this verification is necessary to provide reliability by automatically maintaining multiple copies of data and automatically redeploying processing logic in the event of failures.

II. Cooperative PDP

In order to prove the integrity of data stored in a multi-cloud environment, we define a framework for CPDP based on interactive proof system (IPS) and multi-prover zero-knowledge proof system (MPZKPS), as follows:

Definition 1 (Cooperative-PDP): A cooperative provable data possession \(\mathcal{S} = (\text{KeyGen}, \text{TagGen}, \text{Proof})\) is a collection of two algorithms (KeyGen, TagGen) and an interactive proof system Proof, as follows:

- Key(\(\kappa\)): takes a security parameter \(\kappa\) as input, and returns a secret key \(sk\) or a public-secret keypair \((pk, sk)\);
- 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 \((\zeta, \psi, \sigma)\), where \(\zeta\) is the secret in tags, \(\psi = (u, H)\) is a set of verification parameters \(u\) and an index hierarchy \(H\) 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 \(Pk\);
- \(Pr(\mathcal{P}, V)\): is a protocol of proof of data possession between CSPs \((\mathcal{P} = \{Pk\})\) and a verifier \((V)\), that is,

\[
\langle \Sigma Pk \in (F(k), \sigma(k)) \leftrightarrow V \rangle (pk, \psi) =
\]

\[
1\ F = \{(k)\} \text{ is intact}
\]

\[
0\ F = \{(k)\} \text{ is changed ,}
\]

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 \(\psi\) are the common input between \(P\) and \(V\). At the end of the protocol run, \(V\) returns a bit \(\{0\mid 1\}\) denoting false and true. Where, \(\Sigma Pk \in \mathcal{P}\) 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., \(\land Pk \in \mathcal{P}\) \((F(k), \sigma(k)) \leftrightarrow V\) \((pk, \psi)\), where \(\land\) denotes the logical AND operations among the boolean outputs of all protocols \((Pk, V)\) 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 [13]. To solve this problem, we extend above definition
Cooperative PDP Scheme

In this section, we propose a CPDP scheme for multicloud system based on the aforementioned structure and techniques. This scheme is constructed on collision-resistant hash, bilinear map group, aggregation algorithm, and homomorphic responses.

3.1 Notations and Preliminaries
Let \( \mathbb{H} = \{ H_k \} \) be a family of hash functions : \( \{ 0, 1 \}^n \rightarrow \{ 0, 1 \}^* \) index by \( k \in \mathbb{K} \). We say that algorithm \( \mathcal{A} \) has advantage \( \epsilon \) in breaking collision resistance of \( \mathbb{H} \) if \( \Pr[\mathcal{A}(k) = (m0,m1) : m0 \neq m1, Hk(m0) = Hk(m1)] \geq \epsilon \), where the probability is over the random choices of \( k \in \mathbb{K} \) and the random bits of \( \mathcal{A} \). So that, we have the following definition.

**Definition 3 (Collision-Resistant Hash):** A hash family \( \mathbb{H} \) is \( (t, \epsilon) \)-collision-resistant if no \( t \)-time adversary has advantage at least \( \epsilon \) in breaking collision resistance of \( \mathbb{H} \). We set up our system using bilinear pairings proposed by Boneh and Franklin [14]. Let \( \mathbb{G} \) and \( \mathbb{G}^T \) be two multiplicative groups using elliptic curve conventions with a large prime order \( p \). The function \( e \) is a computable bilinear map \( e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}^T \) with the following properties: for any \( G,H \in \mathbb{G} \) and all \( a, b \in \mathbb{Z}_p \), we have 1) Bilinearity: \( e(\langle a \rangle G, \langle b \rangle H) = e(G, H)ab \); 2) Non-degeneracy: \( e(G, H) \neq 1 \) unless \( G = H = 1 \); and 3) Computation: \( e(G, H) \) is efficiently computable. **Definition 4 (Bilinear Map Group System):** A bilinear map group system is a tuple \( \mathcal{S} = \langle \mathbb{P}, e \rangle \) composed of the objects.

**KeyGen(1k):** Let \( \mathcal{S} = \langle \mathbb{P}, \mathbb{G}, \mathbb{G}^T \rangle \), \( e \) be a bilinear map group system with randomly selected generators \( g, h \in \mathbb{G} \), where \( \mathbb{G}, \mathbb{G}^T \) are two bilinear groups of a large prime order \( p \), \( |p| = 0(\kappa) \). Makes a hash function \( \cdot \) public. For a CSP, chooses a random number \( s \in R \mathbb{Z}_p \) and computes \( S = gs \in \mathbb{G} \). Thus, \( skp = s \) and \( pkp = (g, h) \). For a user, chooses two random numbers \( \alpha, \beta \in R \mathbb{Z}_p \) and sets \( sku = (\alpha, \beta) \) and \( pku = (g, h, H1 = ha, H2 = h\beta) \).

**TagGen(sk, F, P):** Splits \( F \) into \( n \times s \) sectors \( \{ m_i,j \} i \in [1,n] j \in [1,s] \in \mathbb{Z}^{n \times s} \) \( p \). Chooses \( s \) random \( r_1, \ldots, r_s \in \mathbb{Z}_p \) as the secret of this file and computes \( u_i = g r_i \in \mathbb{G} \) for \( i \in [1,s] \). Constructs the index table \( \chi = \{ \chi_i : i = 1 \} \) and fills out the record \( \chi_i \) a in \( \chi \) for \( i \in [1,n] \), then calculates the tag for each block \( m_i \) as

\[
\begin{align*}
(1) & \rightarrow H \sum_{i=1}^{n} r_i (F_n), \\
(2) & k \rightarrow H(\chi), \\
(3) & i \rightarrow H(\chi_i), \\
(3) & c \rightarrow H(\chi_i) \cdot (\prod_{j=1}^{s} u_{m_i,j}),
\end{align*}
\]

where \( F_n \) is the file name and \( c_k \) is the CSP name of \( Pk \in P \). And then stores \( \psi = (u, (1), \chi) \) into TTP, and \( o_k = \{ c_i \} \forall j = k \) to \( Pk \in P \), where \( u = (u_1, \ldots, u_s) \). Finally, the data owner saves the secret \( \zeta = (r_1, \ldots, r_s) \).

**Proof(P, V):** This is a 5-move protocol among the Provers \( (P = \{ P \} i \in [1,c]) \), an organizer \( (O) \), and a Verifier \( (V) \) with the common input \( (pk, \psi) \), which is stored in TTP, as follows:

1) **Commitment\((O \rightarrow V)\):** The organizer chooses a random \( \gamma \in R \mathbb{Z}_p \) and sends \( H' = H\gamma 1 \) to the verifier;

2) **Challenge1**\((O \leftarrow V)\): the verifier chooses a set of challenge index-coefficient pairs \( Q = \{ i, \)
3) Challenge2(\mathcal{P} \leftarrow O): the organizer forwards \( Qk = \{(i, \; vi)\}_{i \in I} \) and sends \( Q \) to the organizer, where \( I \) is a set of random indexes in \([1, \; n]\) and \( vi \) is a random integer in \( \mathbb{Z} \ast p \);

4) Response1(\mathcal{P} \rightarrow O): \( Pk \) chooses a random \( rk \in \mathbb{Z}p \) and \( s \) random \( \lambda j, k \in \mathbb{Z}p \) for \( j \in [1, \; s] \), and calculates a response \( \sigma' k = \Pi(i,vi)\in\mathbb{Q}kvi \cdot \mu j, k = \lambda j, k + \Sigma(i,vi)\in\mathbb{Q}kvi \cdot mi, j, \; \pi j, k = e(\lambda j, kj, H2), \; \) where \( \mu k = [\mu j, k]_{j[1,s]} \) and \( \pi k = \Pi sj=1 \; \pi j, k \). Let \( \eta k \leftarrow grk \in \mathbb{G} \), each \( Pk \) sends \( \theta k = (\pi k, \; \sigma' k, \; \mu k, \; \eta k) \) to the organizer;

5) Response2(0 \rightarrow V): After receiving all responses from \( \{Pi\}i\in[1,c] \), the organizer aggregates \( \{\theta k\} Pk\in\mathcal{P} \) into a final response \( \theta = (\sigma' k, \; \mu k, \; \eta k) \) to the verifier.

Verification: Now the verifier can check whether the response was correctly formed by checking that \( \Sigma \in \mathbb{G} \), each \( Pk \) sends \( \theta k = (\pi k, \; \sigma' k, \; \mu k, \; \eta k) \) to the verifier;

In our scheme, the manager first runs algorithm KeyGen to obtain the public/private key pairs for CSPs and users. Then, the clients generate the tags of outsourced data by using TagGen. Anytime, the protocol Proof is performed by a 5-move interactive proof protocol between a verifier and more than one CSP, in which CSPs need not to interact with each other during the verification process, but an organizer is used to organize and manage all CSPs. This protocol can be described as follows: 1) the organizer initiates the protocol and sends a commitment to the verifier; 2) the verifier returns a challenge set of random index-coefficient pairs \( Q \) to the organizer; 3) the organizer relays them into each \( Pi \) in \( \mathcal{P} \) according to the exact position of each data block; 4) each \( Pi \) returns its response of challenge to the organizer; and 5) the organizer synthesizes a final response from received responses and sends it to the verifier. The above process would guarantee that the verifier accesses files without knowing on which CSPs or in what geographical locations their files reside. In contrast to a single CSP environment, our scheme differs from the common PDP scheme in two aspects:

1) Tag aggregation algorithm: In stage of commitment, the organizer generates a random \( \gamma \in \mathbb{R} \) and returns its commitment \( H' \) to the verifier. This assures that the verifier and CSPs do not obtain the value of \( \gamma \). Therefore, our approach guarantees only the organizer can compute the final \( \sigma' \) by using \( \gamma \) and \( \sigma' k \) received from CSPs. After \( \sigma' \) is computed, we need to transfer it to the organizer in stage of “Response1”. In order to ensure the security of transmission of data tags, our scheme employs a new method, similar to the ElGamal encryption, to encrypt the combination of tags \( \Pi(i,vi)\in\mathbb{Q}kvi \cdot \eta k \), i.e., that, for \( sk = s \in \mathbb{Z}p \) and \( Pk = (g, \; S = gs) \in \mathbb{G}2 \), the cipher of message \( m \) is \( C = (C1 = gr, \; C2 = m \cdot Sr) \) and its decryption is performed by \( m = C2.C^{-s} \). Thus, we hold the equation

\[
\sigma' = (\Pi Pk \in \mathbb{R} \; \Pi(i,vi)\in\mathbb{Q}kvi) = (\Pi Pk \in \mathcal{P} S k . \Pi(i,vi)\in\mathbb{Q}kvi) \cdot e(\lambda j, kj, H2)
\]

2) Homomorphic responses: Because of the homomorphic property, the responses computed from CSPs in a multi-cloud can be combined into a single final response as follows: given a set of \( \theta k = (\pi k, \; \sigma' k, \; \mu k, \; \eta k) \) received from \( Pk \), let \( \lambda j = \Sigma Pk \in \mathcal{P} \lambda j, \) the organizer can compute

\[
\mu j = \Sigma Pk \in \mathcal{P} \cdot \mu j = \Sigma Pk (\lambda j, k
\]

\[
\Sigma(i,vi)\in\mathbb{Q}kvi \cdot mi, j
\]
It is obvious that the final response $\theta$ received by the verifiers from multiple CSPs is same as that in one simple CSP. This means that our CPDP scheme is able to provide a transparent verification for the verifiers. Two response algorithms, Response1 and Response2, comprise an HVR: Given two responses $\theta_i$ and $\theta_j$ for two challenges $Q_i$ and $Q_j$ from two CSPs, i.e., $\theta_i = \text{Response1}(Q_i, \{mk\}_{k \in E_i}, \{sk\}_{k \in E_i})$, there exists an efficient algorithm to combine them into a final response $\theta$ corresponding to the sum of the challenges $Q_i \cup Q_j$, that is, $\theta = \text{Response1}(Q_i \cup Q_j, \{mk\}_{k \in E_i} \cup \{sk\}_{k \in E_i}) = \text{Response2}(\theta_i, \theta_j)$. For multiple CSPs, the above equation can be extended to $\theta = \text{Response2}(\theta_k \mid k \in \mathcal{E})$. More importantly, the HVR is a pair of values $\theta = (\pi, \sigma, \mu)$, which has a constant-size even for different challenges.

### III Dynamic Audit Service

We introduce an audit system architecture for outsourced data in clouds as shown in Figure 1. In this architecture, we consider a data storage service involving four entities: data owner (DO), who has a large amount of data to be stored in the cloud; cloud service provider (CSP), who provides data storage service and has enough storage space and computation resources; third party auditor (TPA), who has capabilities to manage or monitor the outsourced data under the delegation of data owner; and authorized applications (AA), who have the right to access and manipulate stored data. Finally, application users can enjoy various cloud application services via these authorized applications. We assume the TPA is reliable and independent through the following audit functions:

1. **Tag Generation**: the client (data owner) uses the secret key $sk$ to pre-process a file, which consists of a collection of $n$ blocks, generates a set of public verification parameters (PVP) and index-hash table (IHT) that are stored in TPA, transmits the file and some verification tags to CSP, and may delete its local copy.

2. **Periodic Sampling Audit**: by using an interactive proof protocol of retrievability, TPA (or other applications) issues a “Random Sampling” challenge to audit the integrity and availability of outsourced data in terms of the verification information (involves PVP and IHT) stored in TPA.

3. **Audit for Dynamic Operations**: An authorized applications, who hold data owner’s secret key $sk$, can manipulate the outsourced data and update the associated index-hash table (IHT) stored in TPA.

The privacy of $sk$ and the checking algorithm ensure that the storage server cannot cheat the authorized applications and forge the valid audit records. In general, the authorized applications should be cloud application services inside clouds for various application purposes, but they must be specifically authorized by data owners for manipulating the outsourced data. Since the acceptable operations require that the authorized applications must present authentication information for TPA, any unauthorized modifications for data will be detected in audit processes or verification processes. Based on this kind of strong authorization-verification mechanism, we neither assume that CSP is trust to guarantee the security of stored data, nor assume
that a data owner has the capability to collect the evidence of CSP’s faults after errors have been found.

The ultimate goal of this audit infrastructure is to enhance the credibility of cloud storage services, but not to increase data owner’s burden and overheads. For this purpose, TPA should be constructed in clouds and maintained by a cloud storage provider (CSP). In order to ensure the trust and security, TPA must be secure enough to resist malicious attacks, and it also should be strictly controlled to prevent unauthorized access even for internal members in clouds. A more practical way is that TPA in clouds should be mandated by a trusted third party (TTP). This mechanism not only improves the performance of audit services, but also provides the data owner with a maximum access transparency. This means that data owners are entitled to utilize the audit service without further costs besides storing a secret-key and some secret information.

Our experiments clearly demonstrated 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. Furthermore, we optimized the probabilistic query and periodic verification to improve the audit performance. 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. We also proposed Integrity Verification and innovative approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. Our approach allows the data owner to not only audit his content but also enforce strong back-end protection if needed. Moreover, one of the main features of our work is that it enables the data owner to audit even those copies of its data that were made without his knowledge.

**IV Conclusions**

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. Furthermore, we optimized the probabilistic query and periodic verification to improve the audit performance. 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. We also proposed Integrity Verification and innovative approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. Our approach allows the data owner to not only audit his content but also enforce strong back-end protection if needed. Moreover, one of the main features of our work is that it enables the data owner to audit even those copies of its data that were made without his knowledge.

**References**


SP. Santosh has received his master degree in Computer Networks and Information Security from JNTU Hyderabad, India. Has Published many papers in various journals. His research interests include Public Relations, Network Security, Cloud Security, Information Security and Ethical Hacking.