IJDCST @November Issue- V-1, I-7, SW-60
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Scalable & Reliable Proofs in Multi Cloud Storage System

B.V.N KUMAR[1], G.TATAYYA NAIDU[2]
Dept of Computer Science and Engineering
Prasiddha College of Engineering & Technology

Abstract: A multi cloud is a cloud computing environment in which an organization provides and manages some internal resources and the others provided externally. However, this new environment could bring irretrievable losses to the clients due to lack of integrity verification mechanism for distributed data outsourcing. In this paper, we address the construction of a collaborative integrity verification mechanism in hybrid clouds to support the scalable service and data migration, in which we consider the existence of multiple cloud service providers to collaboratively store and maintain the clients' data. We propose a collaborative provable data possession scheme adopting the techniques of homomorphic verifiable responses and hash index hierarchy. In addition, we articulate the performance optimization mechanisms for our scheme and prove the security of our scheme based on multi-prover zero-knowledge proof system, which can satisfy the properties of completeness, knowledge soundness, and zero-knowledge. Our experiments also show that our proposed solution only incurs a small constant amount of communications overhead.

I.INTRODUCTION

Cloud Computing

It is the use of computing resources (hardware and software) which are available in a remote location and accessible over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation.

Cloud storage

A online network storage where data is stored and accessible to multiple clients. Cloud storage is generally deployed in the following configurations: public cloud, private cloud, community cloud, or some combination of the three also known as hybrid cloud.

Cloud Services

Software as a service (SaaS)

The software-as-a-service (SaaS) service-model involves the cloud provider installing and maintaining software in the cloud and users running the software from their cloud clients over the Internet (or Intranet). The users' client machines require no installation of any application-specific software - cloud applications run on the server (in the cloud). SaaS is scalable, and system administration may load the applications on several servers. In the past, each customer would purchase and

load their own copy of the application to each of their own servers, but with SaaS the customer can access the application without installing the software locally. SaaS typically involves a monthly or annual fee.

Development as a service (DaaS)

Development as a service is web based, community shared development tools. This is the equivalent to locally installed development tools in the traditional (non-cloud computing) delivery of development tools.

Platform as a service (PaaS)

It is cloud computing service which provides the users with application platforms and databases as a service. This is equivalent to middleware in the traditional (non-cloud computing) delivery of application platforms and databases.

Infrastructure as a service (IaaS)

It is taking the physical hardware and going completely virtual (e.g. all servers, networks, storage, and system management all existing in the cloud). This is the equivalent to infrastructure and hardware in the traditional (non-cloud computing) method running in the cloud. In other words, businesses pay a fee (monthly or annually) to run virtual servers, networks, storage from the cloud. This will mitigate the need for a data center, heating, cooling, and maintaining hardware at the local level.

Cloud computing architecture

It refers to the components and subcomponents required for cloud computing. These components typically consist of a front end platform (fat client, thin client, mobile device), back end platforms (servers, storage), a cloud based delivery, and a network (Internet, Intranet, Intercloud). Combined, these components make up cloud computing architecture.

Provable Data Possession

The provable data possession (PDP) library allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it or storing a copy himself. It accomplishes this by generating probabilistic proofs of possession by sampling random sets of blocks from the server.

PDP uses homomorphic verifiable tags to minimize the amount of server computation, network traffic and block accesses while achieving a strong guarantee of data possession.

Multi Cloud

Cloud computing environment is constructed based on open architectures and interfaces, it has the capability to incorporate multiple internal and/or external cloud services together to provide high interoperability. We call such a distributed cloud environment as a multi-Cloud (or hybrid cloud). Often, by using virtual infrastructure management (VIM) [1], a multi-cloud allows clients to easily access his/her resources remotely through interfaces such as Web services provided by Amazon.

There exist various tools and technologies for multicloud, such as Platform VM Orchestrator, VMware
vSphere, and Ovirt. These tools help cloud providers construct a distributed cloud storage platform (DCSP) for managing clients' data. However, if such an important platform is vulnerable to security attacks, it would bring irretrievable losses to the clients. For example, the confidential data in an enterprise may be illegally accessed through a remote interface provided by a multi-cloud, or relevant data and archives may be lost or tampered with when they are stored into an uncertain storage pool outside the enterprise. Therefore, it is indispensable for cloud service providers (CSPs) to provide security techniques for managing their storage services.

II.MOTIVATION

A hybrid cloud is a cloud computing environment in which an organization provides and manages some internal resources as well as external resources. For example, as shown in Figure 1, an organization, Hybrid Cloud I, uses a public cloud service such as Amazon's EC2 for general computing purposes while storing customers' data within its own data center in a private cloud. As cloud computing has been rapidly adopted, the hybrid model will be more prevalent for a number of reasons [1]: to provide clients with the same features found in commercial public clouds; to provide a uniform and homogeneous view of virtualized resources; to support configurable resource allocation policies to meet the organization's specific goals (high availability, server consolidation to minimize power usage, and so on); and to meet an organization's changing resource needs.

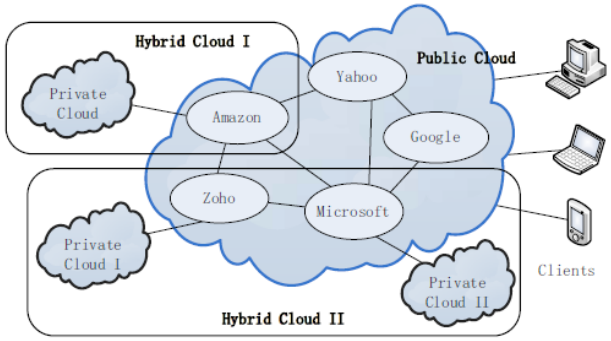

Fig 1: Types of cloud computing: private cloud, public cloud and hybrid cloud.

In hybrid clouds, one of core design principles is dynamic scalability, which guarantees cloud storage services to handle growing amounts of application data in a flexible manner. By employing virtualization, such as VIM, hybrid clouds can effectively provide dynamic scalability of service and data migration. For example, a client might integrate the data from multiple private or public providers into a backup or archive file (see Hybrid Cloud II in Figure 1); or a service might capture the data from other services in private clouds, but the application scripts, intermediate data and results are executed and stored in public clouds [6], [7]. Since this new collaborative paradigm still faces a variety of security concerns, it is necessary to develop a new mechanism for ensuring the security of outsourced data in hybrid clouds.

III.FRAME WORK & MAIN TECHNIQUE

We present our verification framework for hybrid clouds and a formal definition of collaborative PDP. In order to construct such a PDP, we propose three main techniques: fragment structure, hash index hierarchy, and homomorphic verifiable response.

These techniques lay the foundation of our CPDP scheme.

Verification Framework for Hybrid Clouds

Although PDP schemes revolved around public clouds offer a publicly accessible remote interface to check and manage the tremendous amount of data, the majority of existing PDP schemes are incapable of satisfying inherent requirements of hybrid clouds in terms of bandwidth and time. To address this issue, we consider a hybrid cloud architecture as illustrated in Figure 2. In this architecture, we consider a data storage service involving three different entities: Granted clients, who have a large amount of data to be stored in hybrid clouds

and have the permissions to access and manipulate the stored data; Cloud service providers (CSPs), who work together to provide data storage services and have enough storage spaces and computation resources; and Trusted third parties (TTPs), who are trusted to store the verification parameters and offer the query services for these parameters.

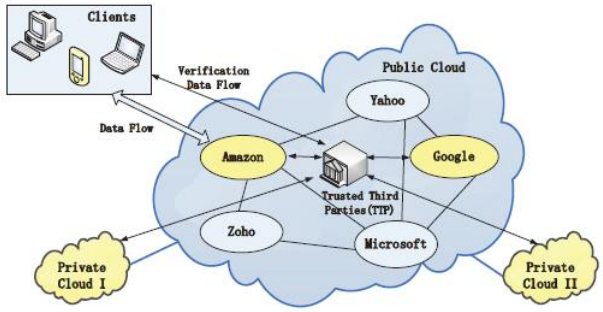

Fig: 2: Verification architecture for data integrity in hybrid clouds

To support such an architecture, a cloud storage provider also needs to add corresponding modules to implement collaborative PDP services. For example, OpenNebula [1] is an open-source virtual infrastructure manager integrated with multiple virtual machine managers, transfer managers, and external cloud providers. In Figure 3, we describe such a cloud computing platform based on OpenNebula architecture, in which a service module of collaborative PDP is added into the cloud computing management platform (CCMP). This module is able to response the PDP requests of TTP through cloud interfaces. In addition, a hash index hierarchy (HIH), which is described in details in Section III-C, is used to provide a uniform and homogeneous view of virtualized resources in virtualization components. For the sake of clarity, we use yellow color to indicate the changes from original OpenNebula architecture.

In this architecture, we consider the existence of multiple CSPs to collaboratively store and maintain the clients' data. Moreover, a collaborative PDP is used to verify the integrity and availability of stored data in CSPs. The verification procedure is described as follows: Firstly, the client (data owner) uses the secret key to pre-process the file, which consists of a collection of n blocks, generates a set of public verification information that is stored in TTP, transmits the file and some verification tags to CSPs, and may delete its local copy; At a later time, by using a verification protocol for collaborative

PDP, the clients can issue a challenge for one CSP to check the integrity and availability of outsourced data in terms of public verification information stored in TTP.

Definition of Collaborative PDP

In order to prove the integrity of data stored in hybrid clouds, we define a framework for collaborative PDP based on interactive proof system (IPS) :

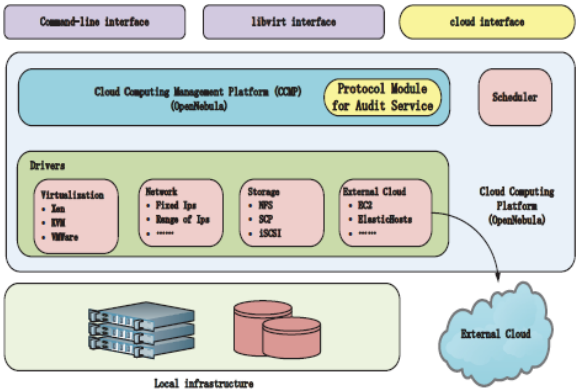

Fig. 3. Cloud computing platform for CPDP service based on OpenNebula architecture.

Definition 1 (Collaborative-PDP)**.** A collaborative provable data possession scheme S′ is a collection of two algorithms and an interactive proof system,

S′ =(K,T ,P):

KeyGen**(**1^k**):** takes a security parameter k as input, and returns a secret key sk or a public-secret keypair (pk, sk);

TagGen**(**sk, F,P**):** takes as inputs a secret key sk, a file F, and a set of cloud storage providers P = {Pk}, and returns the triples (Ç, Ψ,ø), where Ç is the secret of tags, Ψ = (u,H) is a set of verification parameters u and an index hierarchy H for F, ø = { ø (k) Pk∈ P denotes a set of all tags, ø (k) is the tags of the fraction F(k) of F in Pk;

Proof**(**P, V **):** is a protocol of proof of data possession between the CSPs (P = {Pk}) and a verifier (V), that is, (k∈ P Pk(F(k), ø (k)), V) (pk, Ψ), where each Pk takes as input a file F(k) and a set of tags ø (k), and a

public key pk and a set of public parameters Ψ is 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, Pk∈P denotes the collaborative computing in Pk ∈ P.

To realize the CPDP, a trivial way is to check the data stored in each cloud one by one. However, it would cause significant cost growth in terms of communication and computation overheads. It is obviously unreasonable to adopt such a primitive approach that diminishes the advantages of cloud storage: scaling arbitrarily up and down on-demand [12]. For the sake of brevity, we list some used signals in Table I.

Hash Index Hierarchy for Collaborative PDP

As a virtualization approach, we introduce a simple indexhash table to record the changes of file blocks, as well as generate the Hash value of block in the verification process. The structure of our index-hash table is similar to that of

TABLE I
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.

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 must assure that all records in this kind of table differ from one another to prevent the forgery of data blocks and tags. In practical applications, it should been constructed into the virtualization infrastructure of cloud-based storage service [1].
Three layers can be
described as follows:

· First-Layer (*Express Layer*): offers an abstract representation of the stored resources;

· Second-Layer (*Service Layer*): immediately offers and manages cloud storage services;

· Third-Layer (*Storage Layer*): practicably realizes data storage on many physical devices;

Homomorphic Verifiable Response for Collaborative PDP

A homomorphism is a map $f : P \rightarrow Q$ between two groups such that $f(g_1 \oplus g_2) = f(g_1) \otimes f(g_2)$ for all $g_1, g_2 \in P$, where $\oplus$ denotes the operation in P and $\otimes$ denotes the operation in Q. This notation has been used to define Homomorphic Verifiable Tags (HVTs) in [2]: Given two values $\__i$ and $\__j$ for two message m_i and m_j , anyone can combine them into a value $\_'$ corresponding to the sum of the message $m_i + m_j$. When provable data possession is considered as a challengeresponse protocol, we extend this notation to the concept of a Homomorphic Verifiable Responses (HVRs), which is used to integrate multiple responses from the different CSPs in collaborative PDP scheme, as follows:

4The index record is used to support dynamic data operations.

Definition 2 (Homomorphic Verifiable Response)**.** *A response is called homomorphic verifiable response in PDP protocol,if given two responses* Θ_i *and* Θ_j *for two challenges* Q_i *and* Q_j *from two CSPs, there exists an efficient algorithm to combine them into a response* Θ *corresponding to the sum of the challenges* $Q_i \cup Q_j$.

Homomorphic verifiable response is the key technique of collaborative PDP because it not only reduces the communication bandwidth, but also conceals the location of outsourced data in hybrid clouds.

. Verification Process in Collaborative PDP

According to the above-mentioned architecture, four different network entities can be identified as follows: the verifier (V), trusted third party (TTP), the organizer (O), and some cloud service providers (CSPs) in hybrid cloud $P = \{P_i\}_{i\in[1,c]}$. The organizer is an entity that directly contacts with the verifier. Moreover it can initiate and organize the verification process. Often, the organizer is an independent server or a certain CSP in P. In our scheme, the verification is performed by a 5-move interactive proof protocol showed in Figure 4 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 P_i in P according to the exact position of each data block;

4) each P_i returns its response of challenge to the organizer;

5) the organizer synthesizes a final response from these 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.

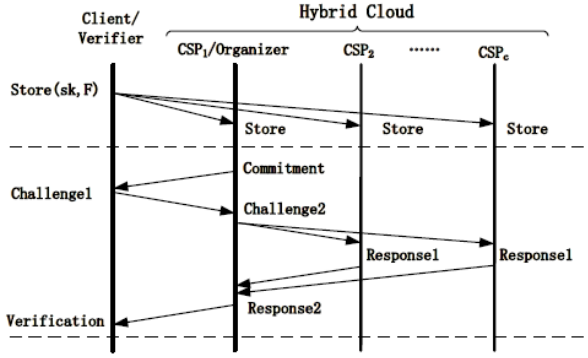

Fig. 4. Flowchart of verification process in our CPDP scheme.

IV.CONCLUSION

In this paper, we addressed the construction of collaborative integrity verification mechanism for distributed data outsourcing in multi clouds. Based on the techniques of homo morphic verifiable responses and hash index hierarchy, we proposed a collaborative provable data possession scheme to support dynamic scalability on multiple storage servers. Our performance analysis indicated that our proposed solution only incurs a small constant amount of communications overhead.

Reference

[1] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. T. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14–22, 2009.

[2] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, "Provable data possession at untrusted stores," in *ACM Conference on Computer and Communications Security*, 2007, pp. 598–609.

[3] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th international conference on Security and privacy in communication netowrks, SecureComm*, 2008, pp. 1–10.

[4] C. C. Erway, A. K¨upc¸ ¨u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *ACM Conference on Computer and Communications Security*, 2009, pp. 213–222.

[5] L. Fortnow, J. Rompel, and M. Sipser, "On the power of multi-prover interactive protocols," in *Theoretical Computer Science*, 1988, pp. 156– 161.

[6] S. Y. Ko, I. Hoque, B. Cho, and I. Gupta, "On availability of intermediate data in cloud computations," in *Proc. 12th Usenix Workshop on Hot Topics in Operating Systems (HotOS XII)*, 2009, pp. 1–10.

[7] S. Pallickara, J. Ekanayake, and G. Fox, "Granules: A lightweight, streaming runtime for cloud computing with support, for map-reduce," in *CLUSTER*, 2009, pp. 1–10.

[8] H.-C. Hsiao, Y.-H. Lin, A. Studer, C. Studer, K.-H. Wang, H. Kikuchi, A. Perrig, H.-M. Sun, and B.-Y. Yang, "A study of user-friendly hash comparison schemes," in *ACSAC*, 2009, pp. 105–114.

[9] A. R. Yumerefendi and J. S. Chase, "Strong accountability for network storage," *TOS*, vol. 3, no. 3, 2007.

[10] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *ESORICS*, 2009, pp. 355–370.