International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 9, November- 2012

Data Integrity Check In Cloud Using Dispersal Code

Mr.S.P.Patil[1]
Assistant Professor
ADCET, Ashta
Maharashtra

Mr.R.S.Nejkar[2]
Assistant Professor
ADCET, ashta
Maharashtra

Ms.D.V.Patil[3]
Assistant Professor
ADCET, ashta
Maharashtra

ABSTRACT

Cloud computing is an emerging field and is envisioned as the next-generation architecture of IT Enterprise. A distributed cryptographic system is introduced that allows a set of servers to prove to a client that a stored file is intact and retrievable. It cryptographically verifies and reactively reallocates file shares. This work studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, a third party auditor (TPA), on behalf of the cloud client is allowed to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of client in the auditing his data stored in the cloud is indeed intact or not. Extensive security and performance analysis show that the proposed scheme is highly efficient and provably secure.

Keywords— Cloud Computing, Proof of data possession (PDP), Proof of retrievability (POR), Public verifiability, Third party in cloud

I. Introduction

Cloud storage represents a family of on-line services for archiving, backup and even primary storage of files. Example: Amazon S3, Google. Cloud-storage providers offer clean and simple file-system interfaces to the user by hiding the complexities of hardware management. The cryptographic community has proposed tools called proofs of retrievability (PORs) and proofs of data possession (PDPs) for security assurance. A POR is a challenge response protocol that enables client to verify that a file is retrievable or not from cloud service provider. The benefit of a POR over simple transmission of file is efficiency. The response can be highly compact, and the client can complete the proof using a small fraction of file. As a standalone tool for testing file retrievability against a single server, though, a POR is of limited value. Detecting that a file is corrupted is not helpful if the file is irretrievable and the client has no recourse.

Public verifiability is useful in environments where file is distributed across multiple systems. In this, file is stored in redundant form across multiple servers. A client can test the availability of file on individual servers via a POR. If it detects corruption within a given server, it can appeal to the other servers for file recovery.

In a distributed file system, a file is spread across servers with redundancy through an erasure code. This supports file recovery in server errors or failures. It can help a client to check the integrity of file by retrieving fragments of file from individual servers and cross-checking their consistency. The system manages file integrity and availability across a collection of servers. It allows use of PORs as tool by which storage resources can be tested and reallocated when failures are detected.

The cloud computing is envisioned as a promising service platform for the Internet. The new data storage paradigm in “Cloud” focuses many challenging design issues. These design issues have profound influence on the security and performance of the overall system. The data integrity verification at untrusted servers is one of the major concerns with cloud data storage. For example, the storage service provider, which experiences Byzantine failures occasionally, may decide to hide the data errors from the clients for their own benefits. The service data files which belong to an ordinary client for saving provider might neglect to keep or deliberately delete rarely accessed money and storage space. It may be more serious. Consider the large size of the outsourced electronic data and the client’s constrained resource capability, the core of the problem can be generalized as how can the client find an efficient way to perform periodical integrity verifications without the local copy of data files.

II. Proposed System:

Many schemes are proposed under different systems and security models to solve this problem. The great efforts are made to design solutions that meet various requirements: high scheme efficiency, stateless verification, unbounded use of queries and retrievability of data, etc. Considering the role of the verifier in the model, all the schemes presented before fall into two categories: **private verifiability and public verifiability**. Although schemes with private verifiability can achieve higher scheme efficiency, public verifiability allows anyone, not just the client (data owner), to challenge the cloud server for correctness of data storage while keeping no private information. Then, clients are able to delegate the evaluation of the service performance to an independent third party auditor (TPA), without devotion of their computation resources. In the cloud, the clients themselves are unreliable or cannot afford the overhead of performing frequent integrity checks. Thus, for practical use, it seems more rational to equip the verification protocol with public verifiability, which is expected to play a more important role in achieving economies of scale for Cloud Computing. That is, the outsourced data themselves should

not be required by the verifier for the verification purpose. In the context of public verification, the importance of blocklessness goes even further because a TPA should not be allowed to possess the original data files for the obvious security concern. In this paper we present a framework and an efficient construction for seamless integration of these two components in our protocol design. Our contribution can be summarized as follows:

1. We propose a general formal PoR model with public verifiability for cloud data storage, in which both blockless and stateless verification are achieved simultaneously.

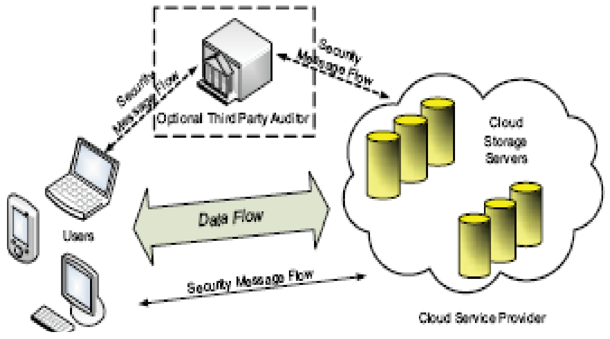

Fig.1. System Architecture

A. Need of public verifiability in cloud

To ensure the correctness of users' data in cloud, an effective and flexible distributed scheme with explicit dynamic data support is proposed. By utilizing the homomorphism token with distributed verification of erasure coded data, the scheme achieves the integration of storage correctness insurance and data error localization. By detailed security and performance analysis, the scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack and even server colluding attacks.

But the system can further be improved in which public verifiability is enforced. Public verifiability allows TPA to audit the cloud data storage without demanding users' time, feasibility or resources. Ivy stores file as log, one log per user. Each participant can read data by consulting all logs. But if you want to modify it then it must be appended to your own log means automatically modify our own log too, although we don't want it. We can also use GFS, PAST or OceanStore file share.

III. THE PROPOSED SCHEME FOR PUBLIC VERIFIABILITY IN CLOUD

In the proposed work, a file is spread across multiple servers with redundancy through a dispersal code as shown in Fig.2. This supports file recovery in server errors or failures. It can help a client to check the integrity of file by retrieving fragments of file from individual servers and cross-checking their consistency. The system manages file integrity and availability across a collection of servers. It allows use of PORs as tool by which storage resources can be tested and reallocated when failures are detected.

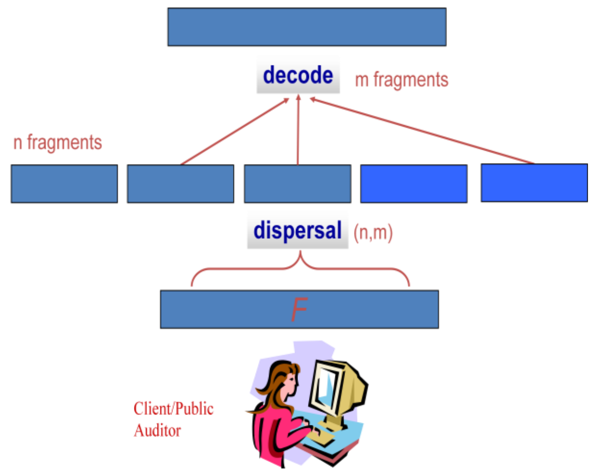

Fig.2. File Distribution using dispersal code

Whenever user wants to store a file on cloud, our system will disperse the entire file in n fragments and stored on n servers instead of replicating entire file on n servers. We can distribute it using an error-correcting code referred as dispersal code. To overcome the problem of corruption attacks we will use **Message Authentication Code (MAC),** computed with secret key known to the client. To make it highly available, we will replicate these data and codebase on different cloud environments using appropriate mechanism. Our system will periodically checks the intactness and retrievability of file by aggregating responses from servers and cross checking it with codeword present in the codebase.

The proposed system will have the following modules:

Module 1: Devising vector using Dispersal code

A file is distributed using the **dispersal code**. Each file block is individually distributed across the n servers under the dispersal code as shown in fig.3.

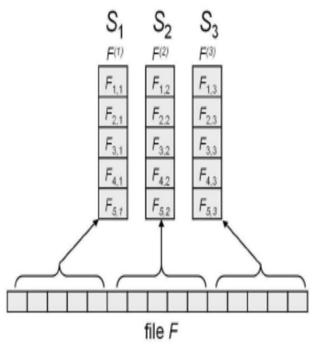

Fig. 3.a Dispersal of File

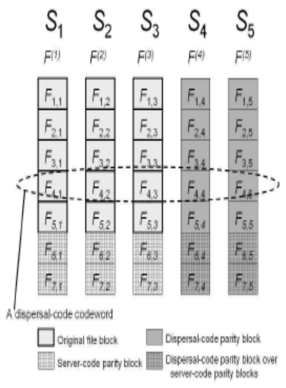
S_1 S_2 S_3 S_4 S_5
$F^{(1)}$ $F^{(2)}$ $F^{(3)}$ $F^{(4)}$ $F^{(5)}$
A dispersal-code codeword
Original file block
Dispersal-code parity block
Server-code parity block
Dispersal-code parity block over server-code parity blocks

Fig.3.b Encoding of file

To increase the lifetime of a file the **Message Authentication Code (MAC)** is embedded into the dispersal code. MAC may be constructed as the straightforward composition of a **Universal Hash Function (UHF)** with a **pseudorandom function (PRF).**

Module 2: Vector replication at each point in the cloud

The vector generated in the Module 1 is replicated in the codebase present the different cloud data servers.

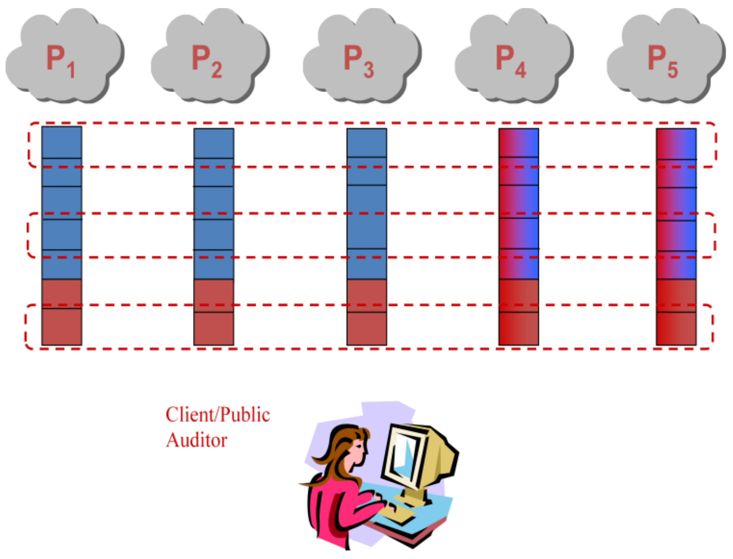

Fig. 4 Replication of vector stripes on different storage server

Module 3: Assembly and verification of data at a point in the cloud

As in the module 2 the file is replicated on different servers. Whenever the request comes from client to check intactness and retrievability of file. We will assemble a stripe from servers and crosscheck with the codeword present in the codebase. Verify data integrity by throwing challenges to some or all of the servers. The stripe generated during response is crosschecked with codeword present in codebase.

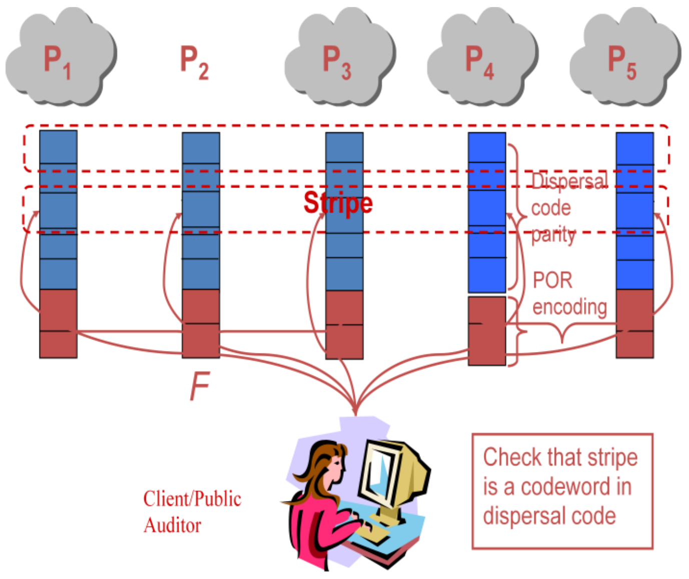

Fig.5 Checking the integrity of the vector at a point in cloud

The proposed system will implement a distributed file system in the cloud environment using dispersal code.

During fragmentation, vector information is generated and replicated on all servers. The client will request for checking integrity of data. It throws challenge using secrete key to some or all servers. In a response a server will generate a stripe. The stripe is crosschecked with codeword present in the codebase. If it matches, then the data is intact and retrievable.

IV. CONCLUSIONS

In this paper, we first distribute file across different servers and then publicly verify it on demand from anyone in cloud. Then we proposed a public verifiability scheme for third party in cloud with the help of error correcting and MAC Code.

V. REFERENCES

[1] Cong Wang, Qian Wang, Kui Ren and Wenjing Lou "Ensuring Data Storage Security in Cloud Computing", IEEE 2009

[2] Kun-Yi Cheng and Chun-Hsin Wu,"Peeraid: A Resilient Path-aware Storage System for Open Clouds" 2009 IEEE.

[3] Wassim Itani, Ayman Kayssi and Ali Chehab, "Privacy-Aware Data Storage and Processing in Cloud Computing Architectures", 2009 8th IEEE International Conference on Dependable Atomic and Secure Computing

[4] Hovav Shacham and Brent Waters, "Compacts Proofs of Retrievability for Large Files,"
Proc. of Asiacrypt 2008, Springer-Verlag, 2008.

[5] Abhishek Verma, Shivram Venkatraman,Matthew Caesar and Roy Campbell," Efficient Metadata Management for cloud computing applications"

[6] J. S. Plank and Y. Ding, "Note: Correction to the 1997 Tutorial on Reed-Solomon Coding," University of Tennessee, Tech. Rep. CS-03- 504, 2003.

7]

http://www.schneier.com/blog/archives/2009/07/homomorphic_enc.html

[8] Michael Miller, "Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online", book.

[9] Anthony T. Velte, Toby J. Velte, Robert Elsenpeter, "Cloud Computing: A Practical Approach", book.

[10] http://en.wikipedia.org/wiki/paillier_crptoststem