Improving Security and Integrity of Data Storage in Cloud Computing By Using Homomorphic Authentication technique

Vipul Patel
Department of Computer science and Engineering
Suresh gyan vihar university, Jaipur, Rajasthan, India

Rahul Kumar
Department of Computer science and Engineering
Suresh gyan vihar university, Jaipur, Rajasthan, India

Ankur Raj
Department of Computer science and Engineering
Suresh gyan vihar university, Jaipur, Rajasthan, India

Abstract- as we seen that, today's generation of IT has widely use the cloud computing services, that means in future cloud computing will important part of IT enterprise. In which the application software and databases are stored at centralized large data storage. Now in cloud data center, management of data & services may not be fully trustworthy. This new phenomenon brings many security challenges. This paper focus on the integrity and security of data storage in cloud computing. The data integrity verification is done by introducing third party auditor (TPA) who has privileges to check the integrity of dynamic data in cloud on behalf of cloud client. Which is significant in economies achievement for cloud computing because TPA can eliminate the direct involvement of cloud client for auditing the data to be stored? Cloud client can get notification from TPA when the data integrity is lost. These systems also support data dynamics via the data operation such as modification, insertion, deletion. There is many work has been done but there is lacks the support of either public auditability or dynamic data operations. This paper achieves both public auditability and data dynamics. This work shows that proposed systems are highly efficient and secure.

Keywords –cloud storage server, data dynamic, public auditability, cloud computing.

I. INTRODUCTION

Cloud computing is the next level in the evolution of internet, the means through which everything to be as services whenever and wherever you need. According to U.S. National Institute of Standards and Technology (NIST): Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.

In cloud computing, everything is delivered as a Service (XaaS). Thus, today there are three main service models such as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

A. Software as a Service

Through cloud computing, cheaper and powerful processor, come together and form "software as a service" service model architecture and put them at centralized data storage server because of this network bandwidth increases and network connection should be reliable.

B. Platform as a Service

PaaS offers a high-level integrated environment to build, test, and deploy custom applications. A client (developer) have the flexibility to build (develop, test and deploy) applications on the provider's platform (API, storage and infrastructure).

C. *Infrastructure as a Service*

It provides software, hardware and equipment's to deliver software application infrastructure with a resource usage based pricing model.

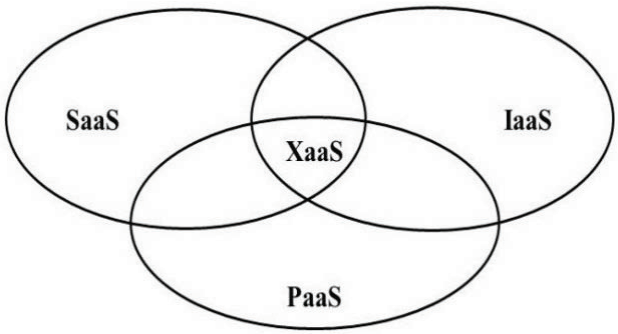

Figure. 1. Cloud service model

This has many challenging design issues which demand great knowledge affecting on the security and performance of the overall system. One of the biggest concerns with cloud data storage is that integrity verification of cloud data at untrusted server.

In order to solve the problem of data integrity verification, there are many security models & scheme has been proposed. In these works, great efforts are made to build the solution that meet requirement like high efficiency, retrievability of data. Due to, involvement of verifier i.e. TPA on behalf of the cloud client, scheme is fall into two categories: private auditability and public auditability before they presented. Although, private auditability achieve higher scheme efficiency and public auditability allow anyone, not only data owner but also publicly, to challenge the cloud server for correctness of the data storage while keeping no private information. Then, clients are given all the privilege to an independent third party auditor (TPA) without devotion of their computing resource. In cloud, clients are may not able to performing frequent integrity checks. So that, there is need to develop verification protocol which has public auditability.

Another major design issue to supporting dynamic data operation for cloud data storage which will provide not only access but also update the data stored at remote location by client. So here we focus on the public auditability and data dynamics.

The rest of the paper is organized as follows. Related works are explained in section II. Design goals and methodology and details are presented in section III and IV, respectively. Concluding remarks are given in section V.

II. LITERATURE REVIEW

Now adays, a large amount of growing interest is pursued in the area of verification of remotely stored data in [1], [2], [3], [4], [5], [6], [7], [8], [9], [11], [12] and [13]. Ateniese et al. [2] are the first for considering Public auditability in their respective model of "provable data possession" to ensure possession of files on storages that is not trusted. In their model, they used Homomorphic tags that are RSA-based for auditing data which is out sourced. Hence public auditability has been achieved. Moreover, Ateniese et al. have not considered the context that involves storage of dynamic data, and extending their scheme from static data storage to dynamic storage in a direct manner can result in design as well as security problems. In [12], Wang et al. have taken into account dynamic data storage in a scenario which is distributed, and their challenge-response protocol is able to determine both the data correctness as well as location of possible errors.

Juels and Kaliski [2] proposed a model of "proof of retrievability", where spot-checking as well as error correcting codes have been used for ensuring both "possession" as well as data files "retrievability" on archive service systems. Though the existing systems strive for providing verification of integrity for various systems of data storage, the issue that involves supporting public auditability as well as data dynamics is not fully addressed.

III. DESIGN GOALS

Our design goals can be summarized as the following:

1) Public auditability for storage correctness assurance: To allow anyone, not just the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand
2) Dynamic data operation support: This will allow the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. The design should be as efficient as possible so as to ensure the seamless integration of public audit ability and dynamic data operation support.

IV. METHODOLOGY AND DETAILS

A. Problem statement

I. System architecture

Fig. 2 shows the network architecture of cloud data storage. Client, cloud storage server and third party auditor these are three entities in network architecture. Client can anyone who is uses cloud services, he may be individual or organization. Client has large data to be stored in cloud and relieve the burden of data maintenance and computation. Cloud storage server (CSS) is entity who has large storage space and computation resources for client which is maintained by cloud service provider (CSP). Third party auditor is trusted and has capabilities of auditing the client's data on demand.

In this phenomenon, client do not have burden of computation and storage because, after putting data on remote servers client deleted that data from it local storage space. But it's important for clients to ensure that their data being correctly stored and maintained. For ensuring, client should be equipped with certain means so client can periodically verify the correctness of the remote data even without existence of local copies. In case that client does not have much time, they deliver their task to a trusted TPA.

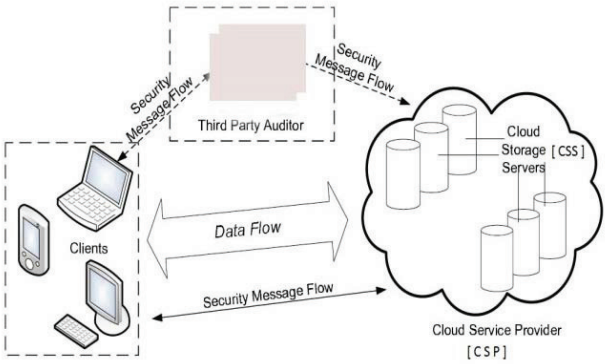

Figure. 2. Cloud data storage architecture

In this work, we only put the problem of verification scheme with public auditability: any TPA in possession of the public key can act as a verifier. We assume that TPA is unbiased while the server is untrusted. For application purposes, the clients may interact with the cloud servers via CSP to access or retrieve their pre-stored data. More importantly, in practical scenarios, the client may frequently perform block-level operations on the data files. The most general forms of these operations we consider in this paper are modification, insertion, and deletion.

II. Security model

The security model defined in [4], we say that the checking scheme is secure if there exists no polynomial-time algorithm that can cheat the verifier with non-negligible probability; also, there exists a polynomial-time extractor that can recover the original data files by carrying out multiple challenges responses. The client or TPA can periodically challenge the storage server to ensure the correctness of the cloud data, and the original files can be recovered by interacting with the server. The authors in [4] also define the correctness and soundness of their

scheme: the scheme is correct if the verification algorithm accepts when interacting with the valid prover (*e.g.*, the server returns a valid response) and it is sound if any cheating server that convinces the client it is storing the data file is actually storing that file.

Our security model has delicate but very important variation in verification process from the PDP or PoR model [2]-[4].Moreover those schemes do not support block insertion and dynamic data operations. The main reason is that signature construction needs information on file index. Re-computing with new indexes once a block is inserted has to be done which causes mover overhead. To overcome this problem, the index information is removed in the computation of signatures. The tag used H(mi) for block "mi" instead of using H(name||i) as used in [3] and h(v||i) [2]. Therefore it doesn't affect other up on individual data operation on any file. In the proposed scheme the client is not given any privilege to calculate H(mi) without data. This is known as block less verification. In order to achieve, the server takes responsibility of computing H(mi) and give it back to prover. The security risk in this approach is that the adversaries are given more opportunities to cheat the prover by generating "mi" or H(mi). The design goals of the system are storage correctness assurance through public auditability, support for dynamic data operations, and block less verification.

B. Proposed Solution

By keeping design goals in mind, here we put the new scheme which ensures the security of cloud data. The protocol developed supports public auditability with dynamic data operations. The proposed system enables public auditability without retrieving block of data from file for this we uses Homomorphic authentication technique that was used in previous model. There is the unforgivable metadata generator computed from individual data blocks. In the proposed work two authenticators such as BLS signature [3] based authenticator. The security mechanism is further described here. The procedure of protocol is divided into

1) Setup

In this phase KeyGen() method is invoked to generate public key and private key. SigGen() is meant for pre-processing and Homomorphic authenticators and along with meta data. The SigGen() method takes two arguments namely secret key and file. The file content is divided into blocks. Then signature is computed for each block. Each block's hash code is taken and two nodes' hash is merged into one in order to generate the next node. This process continues for all leaf nodes until tree node is found. The root element is then taken by client and signs it and send to cloud storage server.

2) Default integration verification

The content of outsourced data can be verified by either client or TPA. This is done by challenging server by giving some file and block randomly. Up on the challenge, the cloud storage server computes the root hash code for the given file and blocks and then returns the computed root hash code and originally stored hash code along with signature. Then the TPA or client uses public key and private key in order to decrypt the content and compare the root hash code with the root hash code returned by clients. This procedure is specified in the following Table 1.

Table -1. Algorithm for data integrity verification

1	Start/begin
2	TPA generates random set
3	CSS computes root hash code based on the filename input
4	CSS computes the originally stored value
5	TPA decrypts the given content and compares with generated root hash
6	After verification, the TPA can determine whether the integrity is breached
7	Stop

3) Dynamic data operation with integrity assurance

Data modifications are the frequent operations on cloud storage. It is a process of replacing specified blocks with new ones. The data modification operation can't affect the logic structure of client's data. Another operation is known as data insertion. Data Insertion is a process of inserting new record in to existing data. The new blocks are inserted into specified locations or blocks in the data file F. The procedure for modification and insertion is given in Table 2.

Table -2. Algorithm for updating & deleting data at CSS

1	Start/begin
2	Client generates new Hash for tree then sends it to CSS
3	CSS updates F and computes new R'
4	Client computes R
5	Client verifies signature. If it fails output is FALSE
6	Compute new R and verify the update
7	If verify succeed, CSS update R'
8	Stop

V.CONCLUSION

The main design consideration is to achieve auditability and data dynamics. The solution is BLS based and it can also be done with RSA based signatures. BLS solution is 160 bits whereas RSA is of 1024 bits. Shortest query and response is possible with BLS. RSA also supports variable sized blocks. MHT (Merkle Hash Tree) has to be used to achieve the solution. The other design consideration is data dynamics. To achieve data dynamics PDP and PoR schemes can be extended. However, they have security problems.

For ensuring security of cloud data storage, it is difficult for enabling a TPA for evaluating the quality of service from an objective and independent point of view. Public auditability is able to allow clients for delegating the tasks of integrity verification to TPA while they are independently not reliable or cannot commit required resources of computation performing verifications in a continuous manner. One more important concern is the procedure for construction of verification protocols which can be able to accommodate data files that are dynamic. In this paper, the problem of employing simultaneous public auditability and data dynamics for remote data integrity check in Cloud Computing is explored. The construction is designed for meeting these two main goals but efficiency is set as the main goal. For achieving data dynamics that are effective, the existing proof of storage models is enhanced through manipulation of the construction of classic Merkle Hash Tree for authentication of block tag. Huge security as well as performance analysis proves that the proposed scheme is efficient and secure to a greater extent.

REFERENCES

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song, "Provable Data Possession at UntrustedStores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS'07), pp. 598-609, 2007.

[2] A. Juels and B.S. Kaliski Jr., "Pors: Proofs of Retrievability forLarge Files," Proc. 14th ACM Conf. Computer and Comm. Security(CCS '07), pp. 584-597, 2007.

[3] H. Shacham and B. Waters, "Compact Proofs of Retrievability,"Proc. 14th Int'l Conf. Theory and Application of Cryptology andInformation Security: Advances in Cryptology (ASIACRYPT '08),pp. 90-107, 2008.

[4] K.D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability:Theory and Implementation," Report 2008/175, Cryptology ePrintArchive, 2008.

[5] M. Naor and G.N. Rothblum, "The Complexity of Online MemoryChecking," Proc. 46th Ann. IEEE Symp. Foundations of ComputerScience (FOCS '05), pp. 573-584, 2005.

[6] E.-C. Chang and J. Xu, "Remote Integrity Check with DishonestStorage Server," Proc. 13th European Symp. Research in ComputerSecurity (ESORICS '08), pp. 223-237, 2008.

[7] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-PreservingAudit and Extraction of Digital Contents," Report 2008/186,Cryptology ePrint Archive, 2008.

[8] A. Oprea, M.K. Reiter, and K. Yang, “Space-Efficient Block Storage Integrity,” Proc. 12th Ann. Network and Distributed System Security Symp. (NDSS ’05), 2005.
[9] T. Schwarz and E.L. Miller, “Store, Forget, and Check: UsingAlgebraic Signatures to Check Remotely Administered Storage,”Proc. 26th IEEE Int’l Conf. Distributed Computing Systems (ICDCS’06), p. 12, 2006.
[10] Q. Wang, K. Ren, W. Lou, and Y. Zhang, “Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance,” Proc.IEEE INFOCOM, pp. 954-962, Apr. 2009.
[11] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, “Scalableand Efficient Provable Data Possession,” Proc. Fourth Int’l Conf.Security and Privacy in Comm. Networks (SecureComm ’08), pp. 1-10,2008.
[12] C. Wang, Q. Wang, K. Ren, and W. Lou, “Ensuring Data StorageSecurity in Cloud Computing,” Proc. 17th Int’l Workshop Quality of Service (IWQoS ’09), 2009.
[13] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia,“Dynamic Provable Data Possession,” Proc. 16th ACM Conf.Computer and Comm. Security (CCS ’09), 2009.