IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 12, Issue 4 (Jul. - Aug. 2013), PP 01-05
www.iosrjournals.org

Enabling Integrity for the Compressed Files in Cloud Server

S.K. Prashanth[1] Dr N. Sambasiva Rao[2] K. SUNITHA[3] V. TEJASWINI[4]
[1]*Associate Professor in CSE, VCE, Hyderabad, India,*
[2]*Professor in CSE, VCE, Hyderabad, India,*
[3]*M.Tech (C.S.E), VCE, Hyderabad, India,*
[4]*M.Tech (C.S.E), VCE, Hyderabad, India,*

***Abstract*:** *Cloud computing has been envisioned as next-generation architecture of IT enterprise. Cloud computing is the internet based model which delivers computing services on demand. Cloud storage moves the users data to large data centers, which are remotely located and on which user does not have any control. Apart having more benefits, this unique feature of the cloud also poses many new security challenges which need to be clearly understood and resolved. The important concerns that need to be addressed in cloud computing is to assure the integrity of data i.e. correctness of his data in the cloud. In this paper we provide a scheme which gives a proof of data integrity in the cloud which the customer can employ to check the correctness of his data in the cloud. This proof can be done by both the cloud and the customer and can be incorporated in the Service level agreement (SLA). Here we are encrypting the hash values and providing the more security. This scheme ensures that the storage at the client side is minimal which will be beneficial for thin clients. Here the computational time is minimized by using integrity and encryption methodology on the data in cloud . It should be noted that this scheme applies only to archive storage data.*

Key words : *Cloud Computing, Data Integrity, Data Encryption.*

Submitted date 21 June 2013 **Accepted Date: 26 June 2013**

I. Introduction

Cloud computing is used to share the resources globally with less cost. we can also called it as 'ON DEMAND'. Cloud computing offers three types of services i,e., Infrastructure as a service(IaaS) , Platform as a service(PaaS) and Software as a service(SaaS). The Cloud Computing simply means "Internet Computing." The Internet is commonly visualized as clouds, hence the term "cloud computing" for computation done through the Internet. With Cloud Computing users can access database resources via the Internet from anywhere, for as long as they need, without worrying about the maintenance. Cloud computing is a new computing paradigm that is built on virtualization, parallel and distributed computing and service oriented architecture. Cloud computing is defined as a model for enabling omnipresent, convenient and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

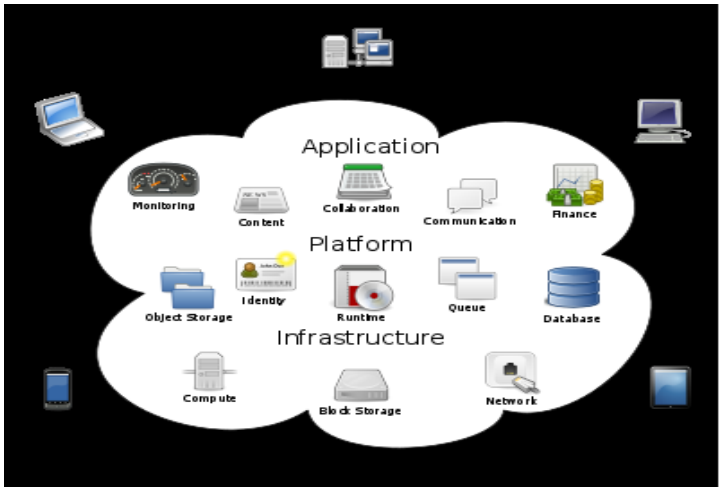

Fig 1: Cloud Computing

Cloud storage is a service of cloud computing which allows data owner to move data from their local computing systems to the cloud. By moving their data in the cloud, data owners can avoid the initial investment

of expensive infrastructure setup, large equipments and daily maintenance cost. The data owners only need to pay how much they actually use. Another reason is that data owners can rely on the cloud to provide more reliable services so that they can access data from anywhere and at any time. Cloud storage moves the owners data to large remote data centers, on which data owner does not have any control. However, this unique feature of the cloud poses many new security challenge issues like data security, privacy, data availability and data integrity in cloud computing due to its internet-based data storage and management.

Here we are concern with data integrity which is defined as the accuracy and consistency of stored data, in absence of any modification to the data between two updates of a file or record. Cloud services should ensure the data integrity and provide a trust to the user privacy.

Although outsourcing data into the cloud is economically attractive for the costand complexity of long-term large-scale data storage, its lacking of offering strong guarantee of data integrity. Hence, the system must have some sort of mechanism to ensure the integrity of data. The current cloud security model is based on the assumption that the user or customer should trust the provider. This is typically governed by a Service Level Agreement (SLA) that in general defines mutual provider and user expectations and obligations.

In this paper we deal with the problem of implementing a protocol for obtaining a proof of data possession in the cloud sometimes referred to as Proof of Retrievability (POR). This problem tries to obtain and verify a proof that the data that is stored by a user at remote data storage in the cloud (called cloud storage archives) is not modified and there by the integrity of the data is assured. Such kinds of proofs are very much helpful in peer-to-peer storage systems, network file systems, long term archives and database systems. Such verification systems avoid the cloud storage archives from misrepresenting or modifying the data stored at it without the consent of the data owner by using frequent checks on the storage archives. Such checks must allow the data owner to efficiently, frequently, quickly and securely verify that the cloud archive is not cheating the owner. Cheating, In this context means that the storage archive might delete some of the data or may change some of the data. It must be noted that the storage server might not be malicious; instead, it might be simply unreliable and lose or inadvertently corrupt the hosted data. Although the data integrity schemes that are to be developed need to be equally applicable for malicious as well as unreliable cloud storage servers. Any such proofs of data possession schemes do not, by itself protect the data from corruption by the archive. It just allows detection of tampering or deletion of a remotely located file at an unreliable cloud storage server.

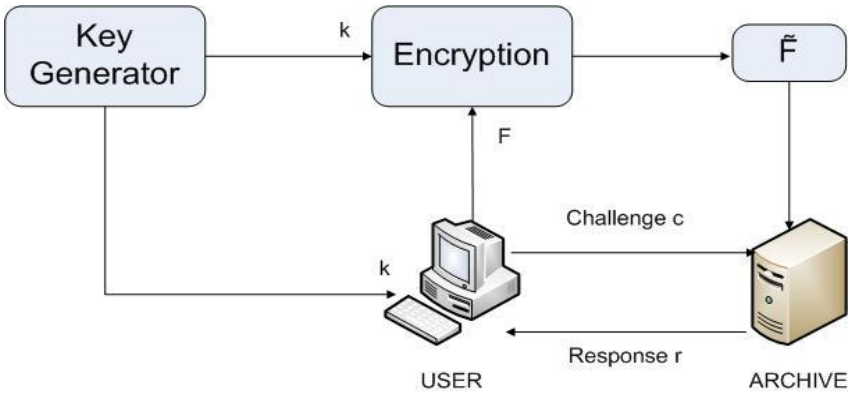

Fig 2: Schematic view of a proof of retrievability based on inserting random sentinels in the data file F

While developing proofs for data possession at untrusted cloud storage servers we are often require high resource costs for its implementation. Accessing the entire file can be expensive in I/O costs which are of large data sizes and are stored at remote server. The problem is further complicated by the fact that the owner of the data may be a small device like PDA or a mobile phone, which have limited CPU power, battery power and communication bandwidth. Hence a data integrity proof that has to be developed needs to take the above limitations into consideration and it has some of the advantages of proposed scheme. The proposed scheme should be able to produce a proof without the need for the server to access the entire file or the client retrieving the entire file from the server. Also the scheme should minimize the local computation at the client as well as the bandwidth consumed at the client side. It also reduces the chance of losing data by hardware failures.

II. Related Work

We describe about provable data possession that provides background for related work and for the specific description of our schemes. A Provable Data Possesion (PDP) protocol checks that an outsourced storage retains a file [2], which consists of a collection of n blocks. The data owner pre-processes the file, generating a piece of metadata that is stored locally, transmits the file to the server, and may delete its local copy of data file. The server stores the file and responds to challenges issued by the client.

As part of pre-processing, the client may modify the file to be stored at the server and also expand the file or include additional metadata to be stored at the server. Before deleting its local copy of the file, the client may execute a data possession challenge to make sure the server has successfully stored the file or not. Here client encrypt a file prior to out-sourcing the storage of data. This encryption is an orthogonal issue in our purpose, the file may consist of encrypted data and our metadata does not include encryption keys. At a later time, the client challenges the server to establish that the server has retained the file. The client requests the server, then the server compute a function of the stored file, which it sends back to the client. By using its local metadata the client verifies the response.

Similar to PDP, Juels and Kaliski introduce the notion of proof of retrievability (POR) [3], which allows a server to influence a client that it can retrieve a file that was previously stored at the server. The POR scheme uses sentinels (disguised blocks) hidden among regular file blocks in order to detect data modification by the server. Although compare with PDP, the POR scheme can only be applied to encrypted files and can handle only a limited number of queries. The PDP schemes can be applied to large public databases other than encrypted ones and put no restriction on the number of challenges that can be executed.

Sravan Kumar R and Ashutosh Saxena [4] have worked to facilitate the client in getting a "proof of integrity of the data " which he wishes to store in the cloud storage servers with bare minimum costs and efforts. In this data integrity protocol the verifier needs to store only a single cryptographic key - irrespective of the size of the data file F. The verifier does not store any data with him. The verifier before storing the file at the archive, preprocesses the file and adds some meta data (data about data) to the file and stores at the archive. At the time of verification the verifier uses this meta data to verify the integrity of the data. By appending meta data to the file, bandwidth of the file increases and it over burden for the verifier. Also computing hash value for even moderately large data files can be computationally burdensome for clients like PDA (personal digital assist), mobile phones etc.

The scheme we present here does not include appending metadata, before storing the file archive, preprocesses the file and compress it, that provides the minimal storage at client side which is more beneficial to users like PDA and mobile phones. Hence our scheme suits well for thin clients which having limited storage space.

III. OUR Contribution

In this scheme we are encrypting only the few bits of data per data not whole data thus reducing the computational overhead on the clients. The client storage overhead is also minimized as it does not store any data. Hence our scheme suits well for thin and thick clients. In our protocol the verifier needs to store only a single key - irrespective of the size of the data file F. The verifier does not store any data he only stores hash values with him to verify the data.

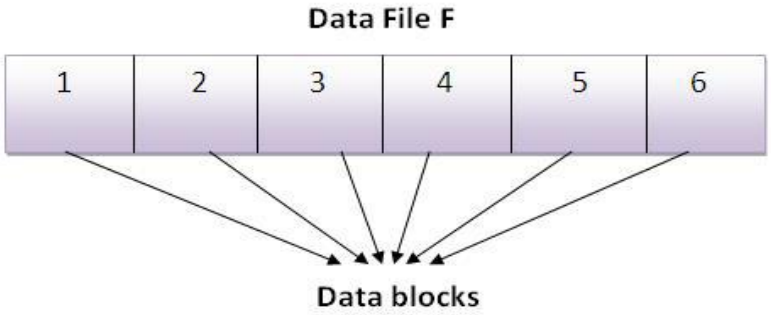

Fig 3: A data file F with 6 data blocks

The verifier before storing the file he encrypts the data file and stores at the archive. At the time of verification the verifier uses this data to verify the integrity using hash values. If the hash values matches then the data is said to be integrity or else the data is illegally modified or deleted. In order to prevent such modifications or deletions other schemes like redundant storing etc, can be implemented which is not a scope of discussion in this paper.

IV. A Data Integrity Proof In Cloud Based On Selecting Data Blocks

The client before storing its data file F in the cloud he should process it and create suitable data which is used for verification of data integrity at the cloud storage. Data integrity can be check by asking queries to the cloud server for suitable replies based on which it concludes whether the data stored in the server is in proper way or not i.e without any modifications or deletion of data.

A. Setup phase

Let the verifier V wishes to the store the file F. Let this file F consist of n file blocks and we encrypts the file. Let each of the n data blocks have m bits in them. A typical data file F which the client wishes to store in the cloud server is shown in Figure 3 . The initial setup phase is as follows:

1) Generation of data file: Let g be a function defined as follows

$$g(i, j) \rightarrow \{1..m\},\ i \in \{1..n\},\ j \in \{1..k\} \qquad (1)$$

Where k is the number of bits per data block. The function g generates for each data block a set of k bit positions within the m bits that are in the data block. Hence g(i, j) gives j^{th} bit in the i^{th} data block. The value of k is the choice of the verifier and is a secret which is known only to him. Therefore for each block we get a set of k bits and we are finding a polynomial hashing for each block so total for all the n blocks we get $n*k$ bits. Figure shows a data block of the file F with random bits selected using the function g.

2) Encrypting the data file: Each of the data from the data blocks m_i is encrypted by using a RSA algorithm to give a new modified data file. In order to not to get same repeated hash values we are performing shift order operation and thus providing more security to the data file. Let h be a hash function which generates a k bit integer αi for each i. This function is a secret and is known only to the verifier V.

$$h : i \rightarrow \alpha i,\ \alpha i \in \{0..2^n\} \qquad (2)$$

$$Mi = \alpha i \qquad (3)$$

The encryption method is used to provide still stronger protection for data file.

3) File compressing: Here we are compressing both the data file(i.e making it into a ZIP file or Archive file) and the hash values. By compressing the data file we need only less storage space thus reducing the storage space also.

4) Placing file in CSP (Cloud Service Provider): The compressed data file is placed in the cloud server and the hash values are kept with client.

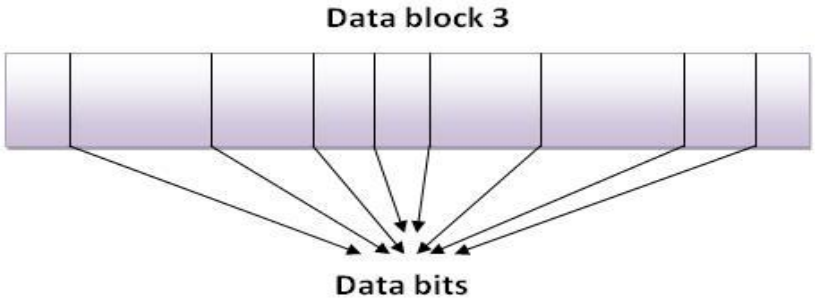

Fig 4: A data block of the file F with random bits selected in it

Fig 5: The encrypted file $\tilde{F}$

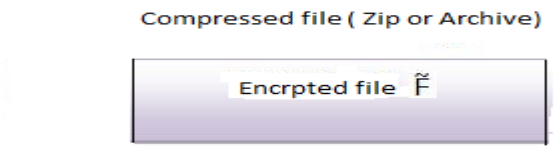

Fig 6: Compressed data file which will be stored on the cloud

B. Verification phase

Let the verifier V wants to verify the integrity of the file F. He throws a challenge to the cloud server and asks it to respond. The challenge and the response are compared and verifier accepts or rejects the integrity proof.

Suppose if the verifier wishes to check the integrity of n^{th} block. The verifier challenges the cloud server by specifying the block number i and a bit number j by using the function g which is only known to the verifier. The verifier specifies the position at which the data file which corresponds to the block i is appended.

This data file will be a k-bit number so he sends a request of k-bit number.. Hence the cloud storage server is required to send a response of k+1 bits for verification by the client. The server decompress the file and compute the hash value, then further it will decrypted by using the private key number αi and the corresponding bit in this decrypted data file is compared with the bit that is sent by the cloud. Any mismatch occurs between the two would mean that there is loss of the integrity of the clients data file at the cloud storage.

V. Conclusion

Here we have proved the integrity of the data by encrypting with RSA algorithm in order to provide security. The hash values are find by using polynomial hashing technique. In order to remove the repeated hash values we are performing shift order operaton in order to provide more security and compressing the data file which reduces the storage space. This scheme provides more security as well as integrity of data.

Our scheme was developed to reduce the computational and storage overhead of the client at cloud storage server. We have also minimized the size of the proof of data integrity so as to reduce the network bandwidth consumption. We only store two functions at the client side, the bit generator function g, and the function h which is used for encrypting the data. Hence the storage at the client is very much minimal compared to all other schemes [3]. Hence this proof provides more benefits to the both thin user like PDA's, mobile phones. The operation of encryption of data takes large computational power. But In our scheme we are eliminating it to a fraction of the whole data thereby saving the time and money of the client.

The network bandwidth is also minimized as the proof size is comparatively very less (k+1 bits for one proof). Our scheme is applied only to static storage of data, dynamic operations not be performed. Hence developing on this will be a future challenge.

References

[1] E. Mykletun, M. Narasimha, and G. Tsudik, "*Authentication and integrity in outsourced databases*," Trans. Storage, vol. 2, no. 2, pp. 107–138, 2006.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "*Provable data possession at untrusted stores*," in CCS '07: Proceedings of the 14th ACM conference on Computer and communications security. New York, NY, USA: ACM, 2007, pp. 598–609.

[3] Juels and B. S. Kaliski, Jr., "*Pors: proofs of retrievability for large files*," in CCS '07: Proceedings of the 14th ACM conference on Computer and communications security. New York, NY, USA: ACM, 2007, pp.584–597.

[4] Sravan Kumar R, Ashutosh Saxena, "*Data Integrity Proofs in Cloud Storage*" 978-1-4244-8953-4/11/$26.00 © 2011 IEEE.

[5] Ku. Swati G. Anantwar, Prof. Karuna G. Bagde, "*Data Integrity and Security in Cloud*," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 1, Issue 7, September 2012