ISSN: 2320-1363

Performance of Cloud Data Integrity proofs in Cloud Storage System

D.Nirmala*1, P.Ramesh Babu*2

M.Tech (CSE) Student Department of CSE, Priyadarshini Institute of Technology & Science, Chintalapudi, Guntur(Dist), Ap, India.

Associated Professor, Department of CSE, Priyadarshini Institute of Technology & Science, Chintalapudi, Guntur(Dist), Ap, India

Abstract

In cloud computing data is moved to a remotely located cloud server. Cloud faithfully stores the data and return back to the owner whenever needed. But there is no guarantee that data stores in the cloud is secured and not altered by the cloud or Third Parity Auditor (TPA). Apart from reducing the storage costs cloud data outsourcing to the cloud also helps in reducing the maintenance. Cloud storage moves the user's data to large data centers, which are remotely located on which user does not have any control. However in this we have unique feature of the cloud data poses many new security challenges which helps to be clearly understood and resolved. As the cloud data is physically there is no accessible to the user, the cloud should provide a way for the user to find if the integrity of his data is managed or is compromised. In this article we are provide a scheme which gives a proof of data integrity in the cloud which the user can employ to check the correctness of his data in the cloud. This research ensures that the storage at the client side is minimal which will be useful for thin clients.

Keywords: Data Integrity, Cryptography, TPA, Cloud Storage.

1. INTRODUCTION

Cloud computing has been envisioned as the next generation architecture of the IT enterprise due to its long list of unprecedented advantages in IT: on demand self service, ubiquitous network access, location independent resource pooling, rapid resource elasticity usage based pricing and transference of risk. Cloud storage is visualized pool where data and applications are stored which are hosted by the third party. Company who desire to store their data in the cloud buy or lease storage capacity from them and use it for their storage needs. Some of the cloud storage benefits are reduce costs, provide more flexibility, reduce IT management of hardware and data, reduce IT management of hardware and data reduce management of web applications through automated updates, and provide greater storage capacity. In spite these benefits, "cloud" lack in some of the issues like data integrity, data loss, unauthorized access, privacy etc.

ISSN: 2320-1363

Data integrity is very important among the other cloud storage issues. Because data integrity ensured that data is of high quality, correct, consistent and accessible. After moving the data to the cloud, owner hopes that their data and applications are secured manner. But that hope may fail sometimes the owner's data may be altered or deleted. In that scenario it is important to verify if one's data has been tampered with or deleted. From the data owners' perspective, including both individuals and IT enterprises, storing data remotely in a cloud in a flexible on-demand manner brings appealing benefits: relief of the burden of storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, personnel maintenance, and so on. While cloud computing makes these advantages more appealing than ever, it also brings new and challenging security threats to the outsourced data.

2. RELATED WORK

The simplest Proof of derivability (POR) scheme can be made using a keyed hash function hk(F). In this scheme the verifier, before archiving the data file F in the cloud storage, pre-computes the cryptographic hash of F using hk(F) and stores this hash as well as the secret key K. To check if the integrity of the file F is lost the verifier releases the secret key K to the cloud archive and asks it to compute and return the value of hk(F). By storing multiple hash values for different keys the verifier can check for the integrity of the file F for multiple times, each one being an independent proof. Though this scheme is very simple and easily implementable the main drawback of this scheme are the high resource costs it requires for the implementation. At the verifier side this involves storing as many keys as the number of checks it want to perform as well as the hash value of the data file F with each hash key. Also, computing hash value for even a moderately large data files can be computationally burdensome for some clients. As the archive side, each invocation of the protocol requires the archive to process the entire file F. Also the archive needs to access only a small portion of the file F unlike in the key-has scheme which required the archive to process the entire file F for each protocol verification. This small portion of the file F is in fact independent of the length of F. The schematic view of this approach is shown in Fig. 2.1. In this scheme special blocks are hidden among other blocks in the data file F.

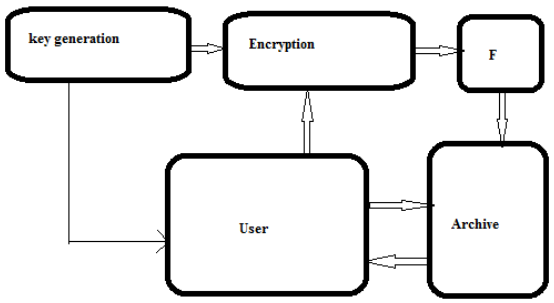

FIG.2.1 schematic view of a POR

The above architecture describes that; user (cloud client) likes to store a file (F) in the cloud server (archive). Before storing the file to the cloud owner needs to encrypt the

ISSN: 2320-1363

file in order to prevent from the unauthorized access.

3. Proposed Work

The present a scheme this does not involve the encryption of the whole data. We encrypt only few bits of data per data block thus reducing the computational overhead on the clients. A data file F with 6 data blocks, the client storage overhead is also minimized as it does not store any data with it. Hence our scheme suits well for thin clients. In our data integrity protocol the verifier needs to store only a single cryptographic key irrespective of the size of data file F and two functions which generate a random sequence. The verifier does not store any data with it. The verifier before storing the file at the archive preprocesses the file and appends some Meta data to the archive. At the time of verification the verifier uses this Meta data to verify the integrity of the data. It is important to note that our proof of data integrity protocol just checks the integrity of data, if the data has been illegally modified or deleted. It does not prevent the archive from modifying the data. In order to prevent such modifications or deletions other schemes like redundant storing etc, can be implemented. This is not a scope of discussion in this paper.

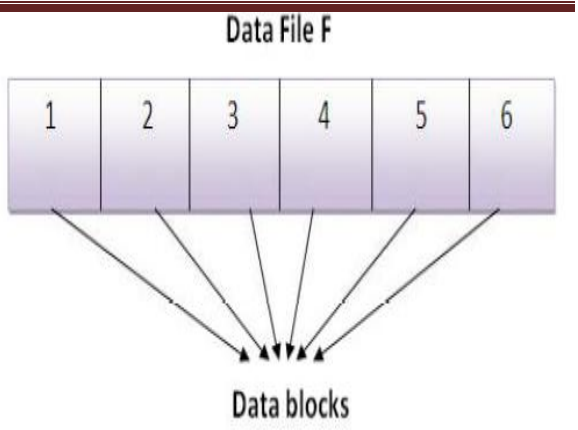

Fig.3.1. A data file F with 6 data blocks

4. Data integrity proof in cloud based on selecting random bits in data blocks

The client before storing its data file F at the client should process it and create suitable mets data which is used in the later stage of verification the data integrity at the cloud storage. When checking for data integrity the client queries the cloud storage for suitable replies based on which it concludes the integrity of its data stored in the client.

4.1 setup phase

Let the verifier V wishes to the store the file F with the archive. Let this file F consist of n file blocks. We initially preprocess the file and create metadata to be appended to 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 is shown in Figure 3.1. The initial setup phase can be described in the following steps.

ISSN: 2320-1363

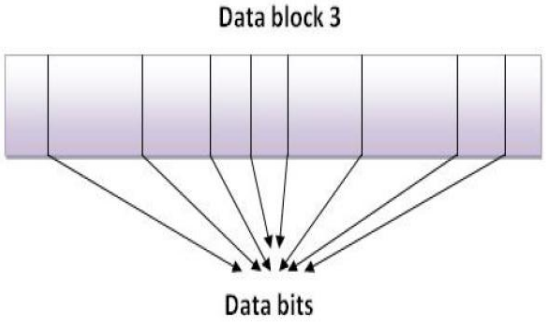

Figure4.1.1 A data block of the file F with random bits selected in it

- ✓ generation of meta data: let g be a function defined as follows

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

Where k is the number of bits per data block which we wish to read as Meta data. 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 the j^{th} bit in the i^{th} data block. The value of k is in the choice of the verifier and is a secret known only to him. Therefore for each data block we get a set of k bits and in total for all the n blocks we get n*k bits. Let m_i represent the k bits of Meta data for the i^{th} block. Figure4.1. shows a data block of the file F with random bits selected using the function g.

- ✓ Encrypting the meta data

Each of the Meta data from the data blocks m_i is encrypted by using a suitable algorithm to give a new modified Meta data M_i. Without loss of generality we show this process by using a simple XOR operation. Let h be a function which generates a k bit integer $_i$ for each i. This function is a secret and is known only to the verifier V.

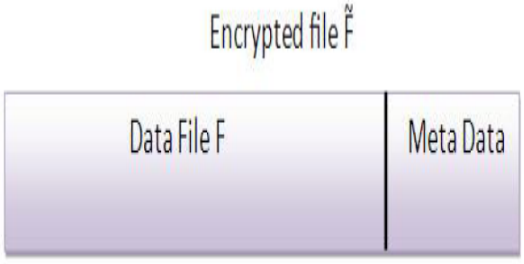

Figure 4.1.2 The encrypted file F□ which will be stored in the cloud.

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

For the Meta data (m_i) of each data block the number $_i$ is added to get a new k bit number M_i.

$M_i = m_i + \ i$ (3)

In this way we get a set of n new Meta data bit blocks. The encryption method can be improvised to provide still stronger protection for verifier's data.

- ✓ Appending of Meta data

All the Meta data bit blocks that are generated using the above procedure are to be concatenated together. This concatenated Meta data should be appended to the file F before storing it at the cloud server. The file F along with the appended Meta data F□ is archived with the cloud. Figure 4.2.1 shows the encrypted file F□ after appending the Meta data to the data file F.

4.2 verification phase

Let the verifier V wants to verify the integrity of the file F. It throws a challenge to the archive and asks it to respond. The challenge and the response are compared and the verifier accepts or rejects the integrity proof. Suppose the verifier wishes to check the integrity of n^{th} block. The verifier challenges the cloud storage server

ISSN: 2320-1363

by specifying the block number i and a bit number j generated by using the function g which only the verifier knows. The verifier also specifies the position at which the Meta data corresponding the block i is appended. This Meta data will be a k-bit number. Hence the cloud storage server is required to send k+1 bits for verification by the client. The Meta data sent by the cloud is decrypted by using the number i and the corresponding bit in this decrypted Meta data is compared with the bit that is sent by the cloud. Any mismatch between the two would mean a loss of the integrity of the client's data at the cloud storage.

5. Conclusion

The next generation of cloud storage provides a new architecture to address the storage, management and analysis of fast growing machine generated data. This paper briefly explaining about the cloud storage, advantages along with its characteristics. Our scheme was developed to reduce the computational and storage overhead of the cloud storage server. We also minimized the size of the proof of data integrity so as to reduce the network bandwidth consumption. At the client we only store two functions, 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 that were developed. In our scheme the encrypting process is very much limited to only a fraction of the whole data thereby saving on the computational time of the client. Many of the schemes proposed earlier require the archive to perform tasks that need a lot of computational power to generate the proof of data integrity. Many of the schemes proposed earlier require the archive to perform tasks that need a lot of computational power to generate the proof of data integrity. But in our scheme the archive just need to fetch and send few bits of data to the client. And also evaluate the performance of the cloud storage performance.

References

[1] R. Sravan kumar and Saxena ,"Data integrity proofs in cloud storage" in IEEE 2011.

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

[3] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy. Washington, DC, USA: IEEE Computer Society, 2000, p. 44.

[4] A. 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.

[5] 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.

ISSN: 2320-1363