Volume 2, Issue 10, October 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Novel Approach to Data Integrity Proofs in Cloud Storage

Neha T* *Department of CSE, CMRIT, Hyderabad India*

P.S Murthy *Department of CSE, CMRIT, Hyderabad India*

Abstract— Next generation computing is basically a development of the Cloud computing. It has been envisioned as the de-facto standard to mounting storage costs of IT Enterprises. Cost of data storage devices increases at a rapid rate which raises the burden of enterprises or individual users to frequently update their hardware and the maintenance of data gives a big headache to the enterprises. Cloud computing is the main gift to the enterprises by reducing the storage cost and the maintenance of the data without giving any hurdle to the enterprises. Cloud storage moves the user's data to large data centres, which are remotely located, on which user does not have any control. However, this unique feature of the cloud poses many new security challenges which need to be clearly understood and resolved. One of the important concerns that need to be addressed is to assure the integrity of the data owner. Since the physical availability of the data is restricted only at the cloud server the data integrity is to be ensured by the provider. This paper focuses on a novel approach to data integrity in the cloud which the client can utilize to check the correctness of his data in the cloud. Service level agreement (SLA) is made between the client and cloud service provider to mount the services.

Keywords— Cloud Computing, Data Integrity,

I. INTRODUCTION

Cloud computing gave a path to data outsourcing. Data outsourcing to cloud storage servers is raising trend among many firms and users owing to its monetary advantages. This is in essence means that the client nothing but owner of the data moves its data to a third party cloud storage server which authentically stocks up the data with it and offer it back to the client whenever required. An enterprise deals with voluminous amount of data. Generation of data is very easy but the storage of data needs the hardware to be updated frequently in order to give space to the large volume of data. In addition to data storage data maintenance also poses huge problem. Storage outsourcing of data to cloud storage helps such firms by reducing the costs of storage, maintenance and personnel. It can also assure a reliable storage of important data by keeping multiple copies of the data thereby reducing the chance of losing data by hardware failures. Cloud computing provides solution to both these problems in an effective manner. It's a pay-per-use model in which the Provider is customized by means of a set of Service Level Agreements (SLAs) [1]. SLA offers guarantees to the provider and client Organizations and individuals can benefit from mass computing and storage centres, provided by large companies with stable and strong cloud architectures. Storing of user data in the cloud has its own advantages and leads to many interesting security questions which need to be inspected extensively for making it a reliable solution to the problem of avoiding local storage of data. Many problems like data authentication and integrity arises when outsourcing the data. To ensure data security the cloud provider stores the data in an encrypted form.

This paper focuses on the problem of implementing a protocol for obtaining a proof of data possession in the cloud sometimes referred to as Proof of retrievability (POR). These types of proofs are very helpful in peer-to-peer storage systems, network file systems, long term archives, web-service object stores, and database systems. POR ensures that 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. Frequent checks are made to clarify the fact that the stored data in the provider side is not get modified without the consent of the data owner. 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. But 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. To ensure file robustness other kind of techniques like data redundancy across multiple systems can be maintained.

While developing proofs for data possession at untrusted cloud storage servers we are often limited by the resources at the cloud server as well as at the client. Given that the data sizes are large and are stored at remote servers, accessing the entire file can be expensive in I/O costs to the storage server. Also transmitting the file across the network to the client can consume heavy bandwidths. Since growth in storage capacity has far outpaced the growth in data access as well as network bandwidth, accessing and transmitting the entire archive even occasionally greatly limits the scalability of the network resources. Furthermore, the I/O to establish the data proof interferes with the on-demand bandwidth of the server used for normal storage and retrieving purpose. The problem is further complicated by the fact that the owner of the data may be a small device, like a PDA (personal digital assist) 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. The 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.

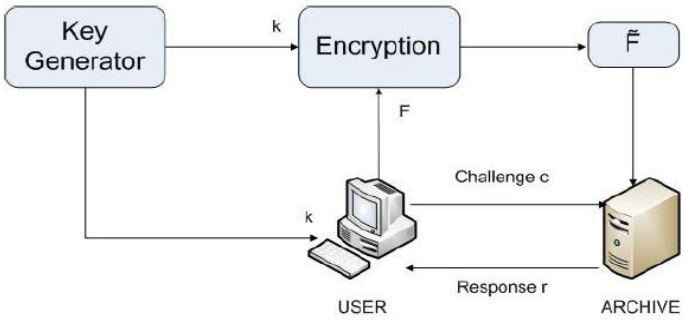

Fig 1 Proof of retrievability based on random sentinels in the file.

II. RELATED WORK

In cloud computing data storage need a lot of attention and care by the provider and owner. A protocol or mechanism is required to ensure that the data is retrieved only by an authenticated owner. Proof of retrievability is one such mechanism. Proof of retrievability (POR) scheme can be made using a keyed hash function Hash (F). In this scheme the verifier, before archiving the data file F in the cloud storage, pre-computes the cryptographic key Hash(F) and stores this hash as well as the secret key K. Though this scheme is very simple and easily implementable the main drawback of this scheme are the high resource costs 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 (PDAs, mobile phones, etc). As the archive side, each invocation of the protocol requires the archive to process the entire file F. This can be computationally burdensome for the archive even for a lightweight operation like hashing. Furthermore, it requires that each proof requires the archiever to read the entire file F - a significant overhead for an archive whose intended load is only an occasional read per file, were every file to be tested frequently[2].

Ari Juels and Burton S. Kaliski Jr proposed a scheme called Proof of retrievability for large files using ”sentinels”[3]. In this scheme, unlike in the key-hash approach scheme, only a single key can be used irrespective of the size of the file or the number of files whose retrievability it wants to verify. 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[3]. 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. 1

In this scheme special blocks (called sentinels) are hidden among other blocks in the data file F. In the setup phase, the verifier randomly embeds these sentinels among the data blocks. During the verification phase, to check the integrity of the data file F, the verifier challenges the archiever (cloud archive) by specifying the positions of a collection of sentinels and asking the archiever to return the associated sentinel values. If the archiever has modified or deleted a substantial portion of F, then with high probability it will also have suppressed a number of sentinels. It is therefore unlikely to respond correctly to the verifier. To make the sentinels indistinguishable from the data blocks, the whole modified file is encrypted and stored at the archive. The use of encryption here renders the sentinels indistinguishable from other file blocks. This scheme is best suited for storing encrypted files. As this scheme involves the encryption of the file F using a secret key it becomes computationally cumbersome especially when the data to be encrypted is large. Hence, this scheme proves disadvantages to small users with limited computational power (PDAs, mobile phones etc.). There will also be a storage overhead at the server, partly due to the newly inserted sentinels and partly due to the error correcting codes that are inserted. Also the client needs to store all the sentinels with it, which may be a storage overhead to thin clients (PDAs, low power devices etc.)

III. NOVEL APPROACH TO DATA INTEGRITY PROOF

In this paper a novel approach is proposed to achieve data integrity proof which does not involve the encryption of the whole data. Only few bits of data are encrypted for each data block and considerably reduce the computing cost. 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 the 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 file and stores at 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 i.e. if the data has been illegally modified or deleted.

The client before sending its file F to store at the provider side, process it and create suitable meta data which is used in the later stage of verification. 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 cloud.

A. Set up phase

Before storing the file F Data owner gives to verifier V. The verifier V wishes to the store the file F with the archive. The file F consist of n file blocks. First and foremost file preprocessing is to be done and meta data is to created in the setup phase. The created meta data is appended to the file itself which is shown in Fig. 2.

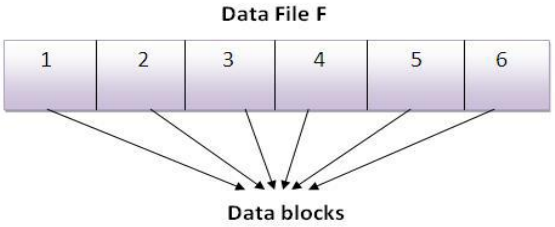

Fig 2. A Data file F with 6 data blocks

The initial setup phase can be described in the following steps

1) Generation of meta-data: Let g be a function defined as

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

Where k is the number of bits per data block for which the meta data is to be read.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 verifier. Therefore for each data block a set of k bits is received and for all the n blocks n^k bits are received.

2) Encrypting the meta data: Each of the meta data from the data blocks m_i is encrypted by using a suitable algorithm and the encrypted data is called as M_i. A secret function which randomly generates a unique h value for given i is known only to the verifier.

$$h: I \rightarrow \alpha_i, \alpha_i \in \{0..2^n\} \quad (2)$$

For the meta data (m_i) form the data block α_i is added to get a new k bit number M_i

$$M_i = m_i + \alpha_i \quad (3)$$

3) 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 e F is archived with the cloud. Figure 4 shows the encrypted file e F after appending the meta data to the data file F.

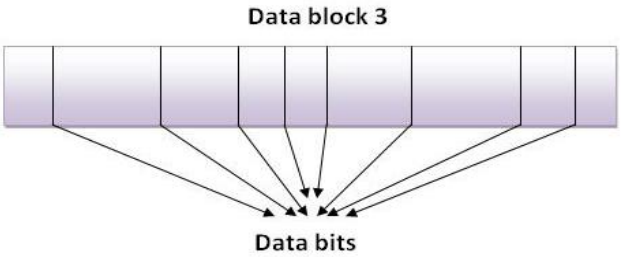

Fig 3. Data block of File f with random bits selected

B. Verification phase

Integrity of the file F is verified by the verifier V. V throws a challenge to the archive and asks it to respond.The challenge and the response are compared and the V accepts or rejects the integrity proof. Assume the verifier V wants to check the integrity of n^{th} block. The verifier challenges the cloud storage server 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 clients data at the cloud storage.

IV. CONCLUSION

A novel approach is proposed to facilitate the client in getting a proof of integrity of the data which is to be stored in the cloud storage servers. The proposed scheme is developed to reduce the computational and storage overhead of the client as well as to minimize the computational overhead of the cloud storage server. In addition to that the size for the proof of data integrity is reduced to minimize the network bandwidth consumption. Overhead of the client is reduced by

storing only 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 previous schemes [4]. Mainly the proposed scheme is advantageous to light weight devices like PDAs and mobile phones. The operation of encryption of data generally consumes a large computational power. 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.

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] 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.

[3] 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.

[4] 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. Pp. 598-609.

