International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869 (O) 2454-4698 (P) Volume-8, Issue-6, June 2018

Data Security in Cloud Storage Using Encryption

Nisha Gautam, Mr. Rahul Singh

Abstract— *Abstract*— **Cloud storage has been conceived as the de-facto solution to the mounting storage costs of IT Enterprises. High costs of data storage devices as well as the rapid rate at which data are being generated it proves costly for enterprises or individual users to frequently update their hardware. Apart from reduction in storage cost data outsourcing to the cloud also helps in thinning out the maintenance. Cloud storage move the user's data to large data centers, which are remotely located, on which user does not have any control. Nevertheless, this unique feature of the cloud poses many new security challenges which need to be clearly understood and answered. We provide a system which passes a verification of information protection in the cloud which the client can use to determine the correctness of his information in the swarm. This security key can be checked upon by both the swarm and the customer and can be incorporated in the Service level agreement (SLA).**

Index Terms— **Cloud Storage, Data Security, Encryption, Integrity**

I. INTRODUCTION

Recently, Data outsourcing to cloud storage servers is growing trend among many households and users owing to its economic advantages. This basically implies that the owner (client) of the data moves its data to a third party cloud storage server, which is supposed to - presumably for a fee - faithfully store the data with it and offer it back to the owner whenever required.

Information origination is far outpacing data storage it proves expensive for small firms to quickly update their hardware whenever additional information is drawn. Besides, maintaining the storages can be a tough task. Storage outsourcing of data to cloud storage helps such firms by bringing down the costs of storage, maintenance and staff office. It can also insure a dependable store of important data by maintaining multiple copies of the data, thereby reducing the probability of missing data by hardware failures. Stored client data in the cloud despite its benefits have many interesting security concerns which demand to be highly investigated for making it a reliable solution to the situation of avoiding local storage of information [7].

The verification systems prevent the cloud storage archives from misreport or modifying the data stored on it without the concurrence of the data owner by using repeated checks on the storage archives. Such checks must take into explanation the information owner to efficiently, frequently, rapidly and securely verify that the cloud archive is not cheating the owner. Cheating, in t Such correct must take into explanation the data owner to efficiently, rapidly and securely verify that the cloud archive is not cheating the owner. Cheating, in this context, implies that the storage archive might delete some of the data or may modify some of the information. Explaining his context, implies that the storage archive might delete some of the data or may modify some of the data. The outsourcing of this data eliminates the worries associated with the induction of the complex underlying hardware, saves increasing high cost in data center management and alleviates the responsibilities of its upkeep. Thus, a big number of Organizations and individuals are taking up these storage services by identifying their information in their cloud storage. All the same, there are security concerns connected with cloud storage. [3].

Recently, information security has been considered as one of the primary obstacles that hinder the growth of cloud storage service. A study [4] surveyed more than 500 CTO and IT managers in 17 countries, showed that despite the possible benefits of cloud storage, systems and individuals do not desire the existing cloud. Storage service providers because the fear of the security threats associated with them. The cloud system in general can be carved up into various types according to the users and range of cloud [3].

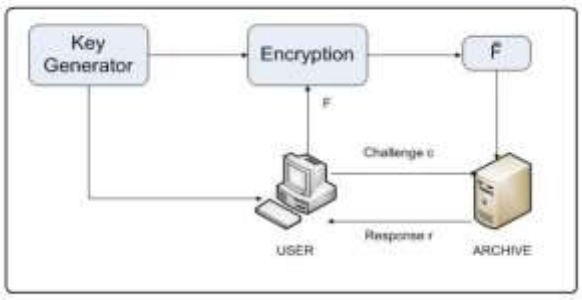

Figure 1: Schematic view of a proof of retrievability based on inserting random sentinels in the data file F[3]

II. LITERATURE SURVEY

Literature survey is the most important measure in the software evolution process. Before developing the tool, it is necessary to determine the time factor, economy n company strength. In one case these matters are satisfied, ten next steps is to see which operating system and language can be used for getting the puppet. At one time the programmers set about building the tool the programmers need lot of external documentation. This funding can be obtained from senior programmers, from books or from sites. Before building the system the above consideration is taken into account in preparing the proposed scheme.

One of the most difficult problems of cloud service solicitation is to advise users to trust the security of cloud service and upload their sensitive data. A though cloud service providers can claim that their services are well-protected by elaborate encryption mechanisms, traditional cloud systems still cannot persuade the users that even if the cloud servers are compromised, the data are still securely protected.[1].

Nisha Gautam, M.Tech Scholar, Department of Computer Science & Engineering, Kanpur Institute of Technology, Kanpur, India.
Rahul Singh Assistant Professor , Department of Computer Science & Engineering, Kanpur Institute of Technology, Kanpur, India.

III. PROPOSED SYSTEM

The simplest Proof of retrivability (POR) scheme can be established using a keyed hash function hook (F). In this scheme the verifier, before filing away the data file F in the cloud storage, pre-computes the cryptographic hash of F uses hook (F) and stores this hash as well as the secret key K. To check if the unity of the file F is lost the verifier releases the secret key K to the cloud archive and requires it to calculate and deliver the value of hk (F).

By storing multiple hash values for different keys the verifier can check for the security of the file F in multiple times, each one being an independent validation. Though this system is really bare and easily implementable the main drawback of this scheme are higher resource costs it requires for the carrying out. At the verifier side this involves storing as many keys as then number of checks it wants to do as well as the hash value of the data file F with each hash key. Also, computing the hash value for even a logically large data files can be computationally demanding for some clients (PDAs, etc.). As the archive side, each invocation of the protocol involves 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 wants that each proof requires the proved to read the entire file F - a significant overhead for an archive whose Intended load is only an occasional read per file, where every file to be examined often [4].

Ari Juels and Burton S. Kaliski Jr proposed a scheme called Proof of retrievability for large files using ”sentinels”[5]. In this system, unlike in the key-hash approach scheme, merely a single key can be applied regardless of the size of the file or the number of files whose retrievability it wants to affirm. Besides the archive needs to access only a minuscule share of the file F unlike in the key-has a scheme which required the archive to process the entire file F for each protocol verification. This little part of the file F is in fact independent of the length of F. The schematic view of this approach is presented in Figure 1. In this scheme special blocks (called sentinels) are covered among other blocks in the data file F. In the setup phase, the verifier randomly embeds these sentinels among the data blocks. In the setup phase, the verifier randomly embeds these sentinels among the data blocks. During the verification phase, to suss out the integrity of the data file F, the verifier challenges the prover (cloud archive) by fixing the attitudes of a collection of sentinels and as concerns the prover to return the associated sentinel values. If the prover has modified or deleted a real portion of F, then with high probability it will also have inhibited a number of scouts. It is consequently unlikely to react correctly to the verifier. To get to the sentinels indistinguishable from the data blocks, the whole modified file is encrypted and stored in the archive. The purpose of encryption here renders the sentinels indistinguishable system affects the encoding 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 telephones and so on). There will likewise be stored overhead on the server, partly due to the newly inserted sentinels and partly due to the error correcting codes that are entered. Besides the node demands to stash away all the sentinels with it, which may be stored overhead to thin clients (PDAs, low power devices etc.)

We introduce a system which does not affect the encryption of the whole information. We encrypt only little pieces of data per data block, thus reducing the computational overhead on the nodes.

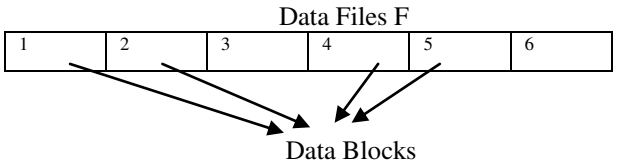

Figure 2: A data file F with 6 data blocks

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 security protocol the verifier needs to store only a single cryptographic key - no matter of the size of the data file F- and two social occasions which bring forth a random succession. 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 information to the file and stores in the archive. At the time of verification the verifier uses this metadata to verify the integrity of the information. It is important to observe that our proof of data security protocol just checks the security of data, i.e. if the information has been illegally modified or deleted. It does not prevent the archive from modifying the information. In parliamentary law to prevent such modifications or deletions other schemes like redundant storing, etc., can be carried out which is not a scope of discourse in this newspaper.

IV. DATA A SECURITY PROOF IN CLOUD BASED ON SELECTING RANDOM BITS IN DATA BLOCK

The client before storing its data file F at the client should treat it and create suitable Meta data which is practiced in the after stage of verify the data security in the cloud storage. When fitting for data security the client queries the cloud storage for suitable replies based on which it resolves the security of its data stored in the client.

A. Setup phase

Let the verifier V wishes to the store the file F with the archive. Let this file F consist of n file block. We initially preprocess the file and create metadata to be added to the file. Allow each of the n data blocks have m bits in them. A typical data file F which the node wishes to store in the cloud is indicated in Figure 2. The initial setup phase can be identified in the next steps

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

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

Where k is the number of bits per data block which we care to read as meta information. The function g generates for each data block a set of k bit positions within them bits that are in the data cube.

Hence g (i, j) gives the jth bit in the ith data block. The value of k is in the selection 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 then blocks we get n∗k bits. Let mi present the k bits of metadata for the if block. Design 3 indicates a data cube of the file F with random numbers selected using the function g. Figure 3 shows a data block of the file F with random bits selected using the function g.

2) Encrypting the meta data: Each of the meta information from the data blocks mail is coded by utilizing a suitable algorithm to make a new modified meta data Mi.

Without loss of generality, we show this operation by utilizing a simple XOR operation. Let h be a function which generates a k bit integer αi for each i. This function is a secret and is known only to the verifier V.

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

For the Meta data (mi) of each data block the number αi is added to draw a new k bit number Mi.

$$Mi = mi + \alpha i \quad (3)$$

In this mode we obtain a set of n new Meta data bit blocks.

The encoding method can be improvised to provide even more substantial protection for verifiers data.

3) The appending of Meta data: All the Meta data bit blocks that are generated using the above procedure are to be attached together. This concatenated meta data should be added to the file F before storing it on the cloud server. The file F along with the appended Meta data e F is archived with the swarm. Image 4 shows the encrypted file F after appending the Meta data to the data file F.

B. Verification phase

Let the verifier V wants to affirm the integrity of the file F. It throws a challenge to the archive and asks it to react. The challenge and the response are compared and the verifier accepts or rejects the integrity proof.

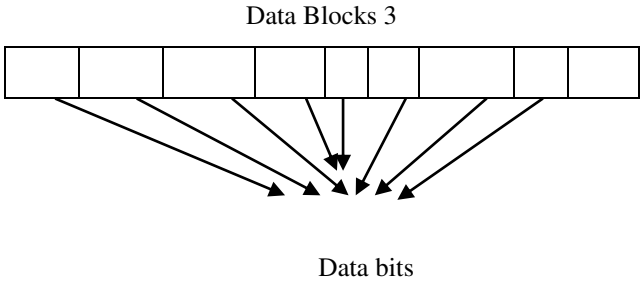

Figure 3: A data block of the file F with random bit selected in it.

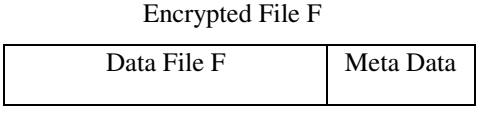

Figure 4: The encrypted file F which will be stored in the cloud

Suppose the verifier wishes to hold the integrity of nth block. The verifier challenges the cloud storage server by setting the block number i and a bit number j generated by applying 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 asked to send k+1 bits for verification by the customer. The meta data sent from the cloud is decrypted by

using the number αi and the corresponding piece in this decrypted meta data is compared to the number that is transmitted from the swarm. Any mismatch between the two would mean a loss of the wholeness of the customer data in the cloud storage.

V. CONCLUSION AND FUTURE WORKS

In this paper, we have acted to help the client in generating a proof of surety of the data which he wishes to store in the cloud storage servers with bare minimum costs and attempts. Our scheme was prepared to scale down the computational and storage overhead of the guest as well as to minimize the computational overhead of the cloud storage server.
We also minimized the size of the proof of data security so as to cut the network bandwidth consumption. Many of the schemes proposed earlier require the archive to do jobs that require a great deal of computational ability to bring forth the validation of data protection. Only in our scheme the archive just needs to fetch and send a few bits of information to the customer. On the client we only store two functions, the bit generator function g, and the function h which is used for encrypting the information. Hence the storage at the client is very much minimal compared to all other schemes [6] that were produced.

The operation of encryption of data generally consumes a great 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 customer.

Many of the schemes proposed earlier require the archive to do jobs that require a great deal of computational ability to bring forth the validation of data protection. Only in our scheme the archive just needs to fetch and send a few bits of information to the customer. The network bandwidth is also minimized as the size of the proof is comparatively very less (k+1 bits for one proof).

It should be mentioned that our system applies only to static storage of information. It cannot treat in case when the data need to be dynamically modified. It will be a challenge to increase the number of queries using this system.

REFERENCES

[1] Vukolic, Marko. ”The Byzantine empire in the intercloud.” ACM SIGACTNews 41.3 (2010): 105-111.
[2] X. Yu and Q. Wen, “A View about Cloud Data Security from Data Life Cycle,” Computational Intelligence and Software engineering, no. 4072020, pp. 0–3, 2010.
[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the clouds: A Berkeley view of cloud computing,”

EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS 2009-28, Feb 2009.
[4] Circle ID Reporter, " Survey: Cloud computing 'no hype', but fear of security and control slowing adoption," Feb. 26, 2009, [Online]. Available: http://www.circleid.com/posts/20090226_cloud_computing_hype_security/
[5] 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.
[6] 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.
[7] E. Mykletun, M. Narasimha, and G. Tsudik, “Authentication and integrity in outsourced databases,” Trans. Storage, vol. 2, no. 2, pp. 107–138, 2006.
[8] "Cloud services outage report," [Online]. Available: http://bit.ly/cloudoutage.

Nisha Gautam, M.Tech Scholar, Department of Computer Science & Engineering, Kanpur Institute of Technology, Kanpur, India.
Rahul Singh Assistant Professor , Department of Computer Science & Engineering, Kanpur Institute of Technology, Kanpur, India