International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Data Integrity Proofs in Cloud Storage

Priyanka Deore[1], Moika Kale[2], Shital Jadhav[3,] Prof.S.N.Bhadane[4]
[1,2,3]*Student,* [4]*Guide Department of Computer Engineering*
, Pune Vidhyarthi Griha's College of Engineering, Nashik,
[1,2,3,4]*University Of Pune , Maharashtra, India*

Abstract — Cloud computing has been envisioned as the de-facto solution to the rising storage costs of IT Enterprises. With the high costs of data storage devices as well as the rapid rate at which data is being generated it proves costly for enterprises or individual users to frequently update their hardware. Apart from reduction in storage costs 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, 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 customer of the integrity i.e. correctness of his data in the cloud. As the data is physically not accessible to the user the cloud should provide a way for the user to check if the integrity of his data is maintained or is compromised. 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 agreed upon by both the cloud and the customer and can be incorporated in the Service level agreement (SLA). This scheme ensures that the storage at the client side is minimal which will be beneficial for thin clients.

Keywords – Cloud Computing, Data Integrity, TPA, POR, Cloud clients.

I. INTRODUCTION

Data outsourcing to cloud storage servers is raising trend among many firms and users owing to its economic advantages. This essentially means 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 provide it back to the owner whenever required.

Storing of user data in the cloud despite its advantages has many interesting security concerns which need to be extensively investigated for making it a reliable solution to the problem of avoiding local storage of data. 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 a remote data storage in the cloud (called cloud storage archives or simply archives) is not modified by the archive and thereby the integrity of the data is assured. Such verification systems prevent 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 modify some of the data.

Reducing costs, accelerating processes and simplifying management are all vital to the success of an effective IT infrastructure. Companies are increasingly turning to provide more flexible IT environments to help them realise these goals. One such solution is Cloud Computing. Cloud Computing enables tasks to be assigned to a combination of software and services over a network. For example storage of large data in cloud reduces costs and maintenance. But the customer is unaware of the storage location. Here risk involved is modification of data or tampering of data. Since the customer does not have control over data the cloud provider should assure the customer that data is not modified. In this paper we propose a data correctness scheme in which a Third Party can audit the data stored in the cloud and assure the customer that the data is safe. This scheme ensures that the storage at the client side is minimal which will be beneficial for thin clients.

A POR is a protocol in which a server/archive proves to a client that a target file F is intact, in the sense that the client can retrieve all of F from the server with high probability. In a POR protocol, a file is encoded by a client before being transmitted to a storage provider for archiving. A POR enables bandwidth-efficient challenge-response protocols to guarantee probabilistically that a file is available at a remote storage provider.

II. EXISTING SYSTEM

As data generation is far outpacing data storage it proves costly for small firms to frequently update their hardware whenever additional data is created. Also maintaining the storages can be a difficult task. It transmitting the file across the network to the

client can consume heavy bandwidths. 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.

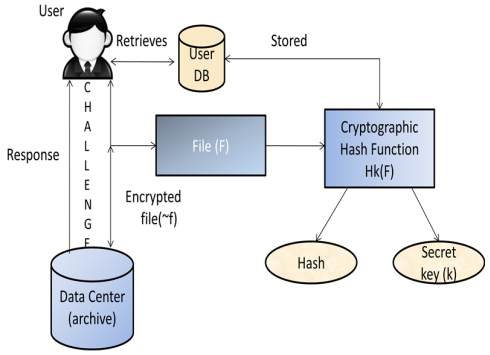

Fig. 1: Existing System

A. Disadvantages

1) The main drawback of this scheme is the high resource costs it requires for the implementation.

2) Also computing hash value for even a moderately large data files can be computationally burdensome for some clients (PDAs, mobile phones, etc).

3) Data encryption is large so the disadvantage is small users with limited computational power (PDAs, mobile phones etc.).

III. PROPOSED SYSTEM

One of the important concerns that need to be addressed is to assure the customer of the integrity i.e. correctness of his data in the cloud. As the data is physically not accessible to the user the cloud should provide a way for the user to check if the integrity of his data is maintained or is compromised. 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 agreed upon by both the cloud and the customer and can be incorporated in the Service level agreement (SLA). 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.

We propose a data correctness scheme which involves the encryption of the few bits of data per data block thus reducing the computational overhead on the clients. This is based on the fact that high probability of security can be achieved by encrypting fewer bits instead of encrypting the whole data. The client storage overhead is also minimized as it does not store any data with it and it reduces bandwidth requirements. In our data integrity protocol the TPA 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 TPA does not store any data with it. The TPA 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 TPA 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. But the data can be stored, that is duplicated at redundant data centers to prevent the data loss from natural calamities. If the data has to be modified which involves updation, insertion and deletion of data at the client side, it requires an additional encryption of fewer data bits. So this scheme supports dynamic behavior of data.

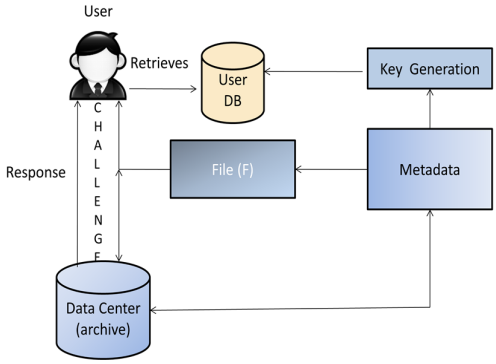

Fig. 2: Proposed System

A. Advantages

1) Apart from reduction in storage costs data outsourcing to the cloud also helps in reducing the maintenance.
2) Avoiding local storage of data.
3) By reducing the costs of storage, maintenance and personnel.
4) It reduces the chance of losing data by hardware failures.
5) Not cheating the owner.

B. Correctness Proof by Selecting Random Bits in Data Blocks

The TPA first selects fewer bits of the entire file and preprocesses the data. This fewer bits constitute metadata. This Meta data is encrypted and appended to the file and sent to the cloud. Then whenever the client needs to verify the data correctness and availability it challenges the cloud through TPA and the data it got is correct, then integrity is ensured. This scheme can be extended for data updation, deletion and insertion at the client side. This involves modification of fewer bits at the client side. There are two phases. One is Setup phase and the other is verification phase. Setup phase include generation of metadata and its encryption. Verification phase includes issuing a challenge to the Cloud server and getting a response and checking its validity.

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 fig 1. The initial setup phase can be described in the following steps.

a) The data file is named as F.
b) Number of blocks n in each file is n.
c) Each block comprises of m bits.
d) k number of bits out of m bits of n blocks are selected for the construction of Meta data.

Fig.3: Data Blocks

Fig. 4: k bits of each block < m bits of each block

2) 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 jth bit in the ith data block. The value of k is the choice of the TPA and is a secret known him only. 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 mi represent the k bits of meta data for the ith block.

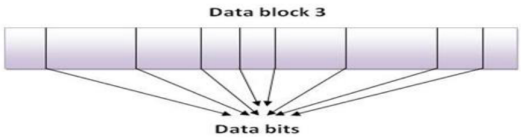

Fig.5: A data block of the file F with random bits selected in it

3) Encrypting the Meta data: Each of the meta data from the data blocks mi is encrypted by using a suitable algorithm to give a new modified meta data Mi . 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 TPA.

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

For the meta data (mi) of each data block the number αi is added to get a new k bit number

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

In this way we get a set of n new meta data bit blocks. The encryption method can be improved to provide still stronger protection for data.

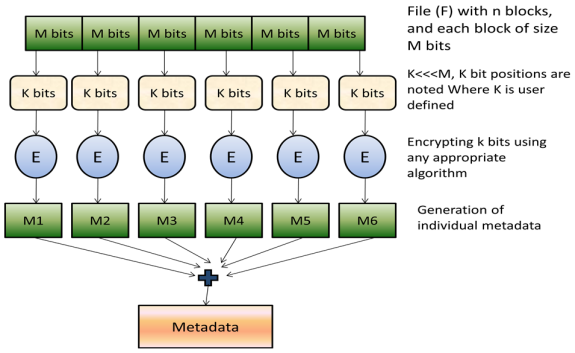

Fig.6: Generation of Metadata

4) *Appending of Meta data:* the Meta data bit blocks are generated using the above procedure .Now they are concatenated together. This concatenated Meta data should be appended to the file F before storing it at the cloud server. The file F when it is appended with meta data becomes F' This file is archived with the cloud.

Fig.7: The encrypted file F which will be stored in the cloud

5) *Verification phase:* Let the TPA 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 if the result is TRUE the TPA accepts the integrity proof. Else if the result of comparison is FALSE it rejects the integrity proof. Suppose the verifier wishes to check the integrity of ith block the TPA 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 TPA knows. The TPA also specifies the position at which the meta data corresponding to the block i is appended. This meta data will be a k-bit number. Hence the cloud storage server is required to send the bits for verification by the client. The meta data sent by the cloud is decrypted by using the number αi .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.

IV. ACKNOWLEDGEMENT

With deep sense of gratitude we would like to thanks all the people who have lit my path with their kind guidance. We are grateful to these intellectuals who did their best to help us during our project. It is my proud privilege to express deep sense of gratitude to, Prof. Dr. N. S. Walimbe, Principal of PVG COE, Nashik, for his comments and kind permission to complete this rst phase project. We remain indebted our Prof.M.T.Jagtap HOD of computer Department and our project guide Prof.S.N.Bhadane and project co-ordinator Prof. J.Y.Kapdnis e, of Computer Department for this timely suggestion and valuable guidance. We thank all colleagues for their appreciable help in the project.

V. CONCLUSION

In this paper we 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. Our scheme was developed to reduce the computational and storage overhead of the client as well as to minimize the computational 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. Hence this scheme proves advantageous to thin clients 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. 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. 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 noted that our scheme applies only to static storage of data. It cannot handle to case when the data need to be dynamically changed. Hence developing on this will be a future challenge. Also the number of queries that can be asked by the client is fixed

a priori. But this number is quite large and can be sufficient if the period of data storage is short. It will be a challenge to increase the number of queries using this scheme.

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 communica- tions security. New York, NY, USA: ACM, 2007, pp. 598–609.
5] H. Shacham and B. Waters, *"Compact Proofs of Retrievability,"* Proc. of Asiacrypt '08, Dec. 2008
[6] K. D. Bowers, A. Juels, and A. Oprea, *"Proofs of Retrievability: Theory and Implementation,"* Cryptology ePrint Archieve, Report/175,2008, http://eprint.iacr.org/.
[7] K. D. Bowers, A. Juels, and A. Oprea, *"HAIL: A High-Availability and Integrity Layer for Cloud Storage,"* Cryptology ePrint Archive, Report 2008/489, 2008, http://eprint.iacr.org/.