Data Integrity Proof for Cloud Storage

Haritha Nuthi, Hemalatha Goli, Ramakrishna Mathe

***Abstract*—Cloud computing has been envisioned as the paramount solution to the rising storage device 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 expensive 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 rises 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 data integrity to customer. 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 a scheme is proposed 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 the organization.

***Index Terms*—Cloud, Service provider, Data Integrity, Cloud Client.**

I. INTRODUCTION

Cloud computing refers to the delivery of computing resources over the Internet. Instead of keeping data on your own hard drive or updating applications for your needs, you use a service over the Internet, at another location, to store your information or use its applications.

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. Storage outsourcing of data to a cloud storage helps such firms by reducing the costs of storage, maintenance and personnel. User can store as much large amount of the data as user wants. 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.

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.

Many problems like data authentication and integrity (i.e., how to efficiently and securely ensure that the cloud storage server returns correct and complete results in response to its clients' queries [1]), outsourcing encrypted data and associated difficult problems dealing with querying over encrypted domain [2] were discussed in research literature.

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 which 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 cloud achiever and thereby 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, web-service object stores, and database systems. 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. 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

Manuscript received August , 2014.

***HarithaNuthi**, M.Tech Student, Department of Computer Science and Engineering, MalineniLakshmaiah Women's Engineering College Guntur, India.*

***HemalathaGoli**, Asst. Professor, Department of Computer Science and Engineering, MalineniLakshmaiah Women's Engineering College Guntur, India.*

***Ramakrishna Mathe**, Asst. Professor, Department of Computer Science and Engineering, MalineniLakshmaiah Women's Engineering College Guntur, India.*

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.

II. RELATED WORK

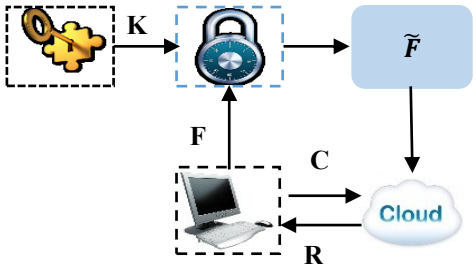

Figure 1 Schematic view of a proof of retrievability based on inserting random sentinels in the data file F [3]

The simplest Proof of Retrievability (PoR) scheme can be made using a keyed hash function $h_k(F)$. In this scheme the verifier, before archiving the data file ***F*** in the cloud storage, pre-computes the cryptographic hash of ***F*** using $h_k(F)$ 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 achiever and asks it to compute and return the value of $h_k(F)$. 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 is "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 (PDAs, mobile phones, etc). As the achiever side, each invocation of the protocol requires the achiever to process the entire file F. This can be computationally burdensome for the achiever even for a lightweight operation like hashing. Furthermore, for each proof verification for each file, it requires that the achiever to read the entire file ***F*** - a significant overhead for an archive whose intended load is only an occasional read per file [3].

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-hash* scheme which required the achiever 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. 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 *prover* (cloud archive) by specifying the positions of a collection of sentinels and asking the *prover* to return the associated sentinel values. If the *prover* 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 achiever. 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.)

Juels et al. [12] describe a "Proof of Retrievability" (PoR) model, where spot-checking and error-correcting codes are used to ensure both "possession" and "retrievability" of data files on remote archive service systems. However, the number of audit challenges a user can perform is fixed a priori, and public auditability is not supported in their main scheme. The authors complete their dynamic auditing system to be privacy preserving and it support the batch auditing for multiple owners [16]. However, due to the large number of data tags, their auditing protocols will incur a heavy storage overhead on the server.

Ateniese et al. [10] consider public auditability in their "Provable Data Possession" (PDP) model for ensuring possession of data files on untrusted storages. They utilize the RSA-based homo-morphic linear authenticators for auditing outsourced data and suggest randomly sampling a few blocks of the file. However, among their two proposed schemes, the one with public auditability exposes the linear combination of sampled blocks to external auditor. When used directly, their protocol is not provably privacy preserving, and thus may leak user data information to the external auditor.

By referring different existing system, we have described some suggested requirements for public auditing services and the state of threat that fulfills them. However, this is still not enough for a secure cloud data storage system, and further challenging issues remain to be supported and resolved.

1. What will happen if the data owner and TPA are unreliable? In this case the auditing result should identify the data correctness as well as it should be able to determine which entity is responsible for the problem like owner, TPA or cloud server. So systems accountability should be achieved.
2. Performance is another important aspect in cloud computing data storage security and its integrity for any physical system.

ISSN: 2278 – 1323

3. Cloud data storage provides dynamic and scalable storage services. It also allows easy on-demand file sharing on cloud. The challenge in this case is that legacy users, who access data and it can also modify the owner's data in the cloud. So major challenge is dynamics support for public auditing services while maintaining system runtime.

To securely launch an effective third party auditor (TPA), the following two essential requirements should be met;

1) TPA should be able to efficiently audit the cloud data storage without demanding the local copy of data, and introduce no additional on-line burden to the cloud user.
2) The third party auditing process should bring in no new vulnerabilities towards user data privacy.

Therefore our system protocol provide Confidentiality, Dynamic auditing and Batch auditing i.e. auditing protocol is able to support the batch auditing for many owners and many clouds[15][16].

III. PROPOSED SCHEME

We present a scheme which 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.

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 achiever, preprocesses the file and appends some *metadata* to the file and stores at the archive. At the time of verification the verifier uses this metadata 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. It does not prevent the achiever from modifying the data. In order to prevent such modifications or deletions other schemes like redundant storing etc, can be implemented which is not a scope of discussion.

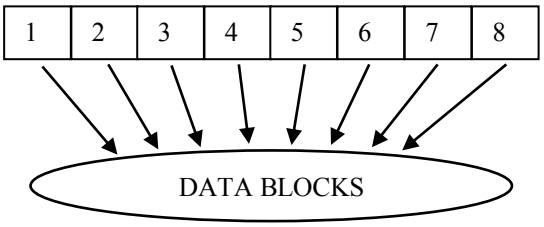

Figure 2 A data file F with 6 data blocks

IV. SYSTEM DESIGN

The client before storing its data file *F* at the client should process it and create suitable metadata 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.

A. Setup phase

Let the verifier ***V***wishes to the store the file ***F***with the achiever. 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 2. The initial setup phase can be described in the following steps

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

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

Where ***k***is the number of bits per data block which we wish to read as metadata. The function ***g***generates for each datablock 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 metadata for the i^{th}block. Figure 3 shows a data block of the file ***F***with random bits selected using the function ***g***.

2) Encrypting the metadata: Each of the metadata from the data blocks mi is encrypted by using a suitable algorithm to give a new modified metadata 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 α_i for each *i*. This function is a secret and is known only to the verifier *V*.

$$h: i \rightarrow \propto_i, \propto_i \in \{0 \dots 2^n\} \quad \dots\dots\dots(2)$$

For the metadata (mi) of each data block the number α_i is added to get a new *k* bit number Mi.

$$m_i \rightarrow m_i + \propto_i \quad \dots\dots\dots(3)$$

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

3) Appending of metadata: All the metadata bit blocksthat are generated using the above procedure are to be concatenated together. This concatenated metadata should beappended to the file *F* before storing it at the cloud server. Thefile *F* along with the appended metadata $\tilde{F}$is archived withthe cloud. Figure 4 shows the encrypted file *F* after appending the metadata to the data file *F*.

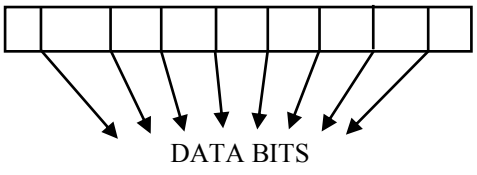

Figure 3 A data block of the file F with random bits selected in it

B. Verification phase

Let the verifier *V* want 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 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 metadata corresponding the block ***i*** is appended. This metadata will be a k-bit number. Hence the cloud storage server is required to send *k+1* bits for verification by the client.

The metadata sent by the cloud is decrypted by using the number $\alpha_i$and the corresponding bit in this decrypted metadata is compared with the data 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.

Encrypted File $\tilde{F}$

Data File (F)	Meta Data

Figure 4The encrypted file $\tilde{F}$ to be stored in the cloud

V. Conclusion and Future work

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 [4] 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 encryptingprocess 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 [3]. 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 priory. 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 communications security*. New York, ny, usa: acm, 2007, pp. 598-609.

[5] Yan Zhua,b, Hongxin Huc, Gail-Joon Ahnc, Stephen S. Yauc."Efficient audit service outsourcing for data integrity in clouds". In "The Journal of Systems and Software 85 (2012) 1083– 1095".

[6] 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, "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp. 50-58, 2010.

[7] T. Velte, A. Velte, and R. Elsenpeter, Cloud Computing: A Practical Approach, first ed., ch. 7. McGraw-Hill, 2010. [4] A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.

[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 598-609, Oct. 2007.

[9] M.A. Shah, M. Baker, J.C. Mogul, and R. Swaminathan, "Auditing toKeep Online Storage Services Honest," Proc. 11th USENIX Workshop Hot Topics in Operating Systems (HotOS '07), pp. 1-6, 2007.

[10] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 598-609, 2007.

[11] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-PreservingAudit and Extraction of Digital Contents," Cryptology ePrint Archive, Report 2008/186, 2008.

[12] A. Juels and J. Burton, S. Kaliski, "PORs: Proofs of Retrievability forLarge Files," Proc. ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.

[13] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE Trans. Parallel Distributed Systems, vol. 22, no. 5, pp. 847-859, May 2011.

[14] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving PublicAuditing for Data Storage Security in Cloud Computing," Proc. IEEE INFOCOM, pp. 525-533, 2010.

[15] C. Wang, K. Ren, W. Lou, and J. Li, "Toward Publicly Auditable Secure Cloud Data Storage Services," IEEE Network, vol. 24, no. 4, pp. 19-24, July/Aug. 2010.

[16]C. Wang et al., "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing," Proc. IEEE INFOCOM '10, Mar. 2010.

[17] Cong Wang and Kui Ren, Illinois Institute of Technology Wenjing Lou, Worcester Polytechnic Institute Jin Li, Illinois Institute of Technology "Toward Publicly Auditable Secure Cloud Data Storage Services". 0890-8044/10/2010 IEEE.

[18] Cong Wang, Student Member, IEEE, Qian Wang, Student Member, IEEE, Kui Ren, Senior Member, IEEE, Ning Cao, and Wenjing Lou, Senior Member, IEEE "Toward Secure and Dependable Storage Services in Cloud Computing" IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 5, NO. 2, APRIL-JUNE 2012.

[19] Kan Yang, Student Member, IEEE, and Xiaohua Jia, Fellow, IEEE "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing" IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 9, SEPTEMBER 2013.

[20] Cong Wang, Member, IEEE, Sherman S.M. Chow, Qian Wang, Member, IEEE, Kui Ren, Senior Member, IEEE, and Wenjing Lou, Senior Member, IEEE "Privacy-Preserving Public Auditing for Secure Cloud Storage" IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 2, FEBRUARY 2013.

ISSN: 2278 – 1323