ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 4, June 2012

ICCC: Information Correctness to the Customers in Cloud Data Storage

Gangolu Sreedevi
M. Tech (CS),
ASCET, India
gangolusreedevi@gmail.com

Prof. C. Rajendra,
H.O.D, CSE Department,
ASCET, India

Abstract- **Cloud storage is the only solution to the IT organizations to optimize the escalating storage costs and maintenance. Data outsourcing in to the cloud has become today trending environment for the thin organizations. The entire user's data is store in large data centers, those which are located remotely and the users don't have any control on it. on the other hand, this unique feature of the cloud poses many new security challenges which need to be clearly understood and resolved. Even the information is not accessible by the user the data centers should grant a way for the customer to check the correctness of his data is maintained or is compromise.**

As a solution this we are proposing providing information correctness to the customers in Cloud Storage. The proposing system gives a information correctness proof, the customer can employ to check the correctness of his data in the cloud. The correctness of the data can be approved upon by both the cloud and the customer and can be included 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 correctness, Cloud Storage

I. INTRODUCTION

Many organization and users outstanding their economic advantages through the information outsourcing to cloud storage servers. Because cloud outsourcing is the raising trend today's computing. The cloud storage means that the holder of the data store his data in a third party cloud server which is supposed to apparently for a fee and provide back the data to the client whenever required. The data storage becomes costly for small Organizations to regularly update their hardware whenever additional data is made and maintenance of the stored data can be hard task. Cloud data outsourcing helps such organizations by reducing the maintenance costs, personnel costs and storage costs. The cloud storage also offers a reliability of important data by maintaining multiple copies of the data thereby reducing the chance of data lose by hardware failures.

The cloud in spite of many protection concerns for the users data storing which need to be expansively investigated for making it a reliable solution to the problem of avoiding local storage of data.

Many problems like data verification and truthfulness [1] outsourcing encrypted data and associated difficult problems dealing with querying over encrypted domain [2] were discussed in research review.

In this paper our proposal ICCC deal with the crisis of implementing a protocol for obtaining the correctness of data possession in the cloud sometimes referred to as Proof of retrievability. The ICCC tries to achieve and validate a correctness proof that the data that is stored by a user at remote data storage in the cloud archives is not modified by the archive and thereby the correctness of the data is assured. Such kinds of correctness proofs are much helpful in P to P (peer-to-peer) storage systems, long term archives, database systems, file storage systems. The authentication systems avert the archives of cloud from mis-representing or modifying the data stored at it without the approval of the data holder by using frequent checks on the storage archives. Such checks must allow the data holder to efficiently, frequently, quickly and securely validate that the cloud archive is not cheating the data holder. Cheating, in this situation, means that the cloud storage archive might delete some of the data or may tamper or modify some of the data. It must be noted that the cloud storage server might not be spiteful; instead, it might be simply undependable and lose or unconsciously corrupt the hosted data. But the information correctness schemes that are to be developed need to be often applicable for spiteful as well as unreliable cloud storage servers. Any such correctness proofs possession schemes will not protect the information from dishonesty by the archive. To guarantee file toughness other kind of techniques like information redundancy across multiple systems can be maintained. It just allows invention of modification of a remotely located file at an unreliable cloud storage server.

While implementing the information correctness proofs for data possession at un-trusted cloud storage servers we are habitually partial by the property at the cloud server as well as at the client (customer). Given that the data sizes are large and are stored at remote servers, accessing the entire file can be expensive in hardware costs to the storage server.

Transmitting the information across the network to the user (client) can devour heavy bandwidths. Because development in cloud storage capacity has fat outpaced the development in cloud data access as well as network speed or bandwidth, transmitting and accessing the entire network resources.

Furthermore, the hardware to establish the information correctness proof obstruct with the on-demand bandwidth of the cloud server used for normal storage and retrieving purpose. The problem is more intricate by the fact that the holder of the data may be a small device, like a Personal Digital Assist (PDA) or a mobile phone, which have limited battery power, CPU power and communication bandwidth. Therefore a information correctness proof that has to be urbanized needs to take the above limitations into consideration. The ICCC should be able to produce a

evidence without the need for the cloud server to access the entire data file or the client (user) retrieving the entire file from the server. Also the ICCC should diminish the local computation at the client as well as the bandwidth consumed at the client (user).

II. RELATED WORK

Cloud computing refers to the delivery of computing and storage capacity as a service to a heterogeneous community of end-recipients. The name comes from the use of clouds as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts services with a user's data, software and computation over a network. It has considerable overlap with software as a service (SaaS).

End users access cloud based applications through a web browser or a light weight desktop or mobile app while the business software and data are stored on servers at a remote location. Proponents claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and enables IT to more rapidly adjust resources to meet fluctuating and unpredictable business demand.

Cloud computing relies on sharing of resources to achieve coherence and economies of scale similar to a utility (like the electricity grid) over a network (typically the Internet). At the foundation of cloud computing is the broader concept of converged infrastructure and shared services.

The ICCC system can be prepared using a encryption function h_k (F), ICCC the correctness verifier, before archiving the data file F in the cloud data storage, pre-computes the cryptographic encrypted hash value of F using h_k (F) and stores this hash value as well as the encrypted key K. To check if the correctness of the file F is lost, the correctness verifier releases the encrypted key K to the cloud storage archive 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 correctness of the file F for multiple times, each one being independent evidence.

Though this ICCC approach is very simple and easily implemental, the main disadvantage of this approach is the high supply costs it requires for the implementation. The verifier in this approach requires storing as many keys and the number of checks to perform and the hash value of the information file F with each hash key. Computing hash value for even a fairly large information files can be computationally arduous for some clients (PDAs, mobile phones, etc).

The archive side requires each incantation of the protocol; the archive requires processing the entire information file F. This can be computationally difficult for the archive even for a lightweight operation like hashing. Furthermore, it requires that each information correctness proof requires the proverb 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 [3].

Juels and Burton proposed a approach called Proof of retrievability for large files using" sentinels" [3]. In this approach, unlike in the key-hash approach, 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 protocol verification. This small portion of the file F is in fact independent of the length of F. The Figure 1 shows the pictorial view of the proposing ICCC.

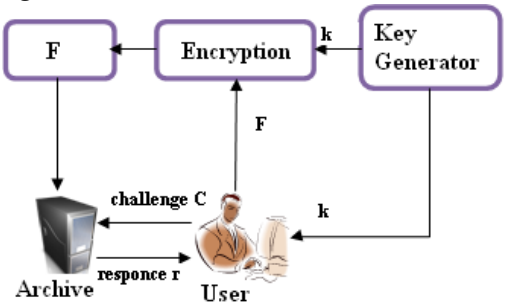

Figure 1 pictorial view of information correctness proof of retrievability based on inserting random sentinels in the data file F

In this approach 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. In the verification phase, to check the correctness 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.

When the encrypted information is large, this approach involves the encryption of the F utilizing the key it becomes computationally awkward. There will also be storage transparency at the server, partly due to the newly inserted sentinels and partly due to the error correcting codes that are inserted. The client wants to store all the sentinels with it, which may be storage overhead to thin clients (PDAs, low power devices etc.).

III. ICCC APPROACH

Our ICCC information correctness proof approach does not encrypt the whole information. The ICCC encrypts only few bits of the data per data block thus reducing the computational transparency on the clients. Our approach is suitable for thin organizations because the client storage transparency is also minimized as it does not store any data with it.

The verifier has to store only a single encrypted key in our information correctness protocol. Because of the irrespective of the size of the information file F - and the two functions which produce a random sequence. The verifier does not store any data with it. The verifier appends meta data to the file F before storing the file at the archive and stores at the archive side. In the verification phase the verifier uses this meta data to verify the correctness 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.

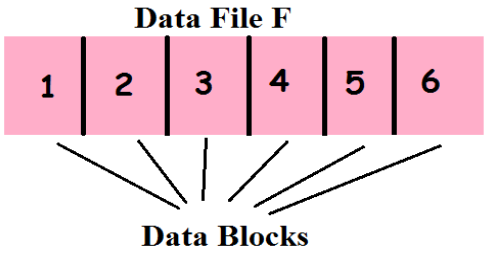

Figure 2 A data file F with 6 data blocks

IV. A INFORMAION CORRECTNESS PROOF IN CLOUD

In this ICCC approach the user should route and create suitable Meta data which is used in the later verification stage before storing the information file at the cloud storage the user queries the cloud data storage for proper replies for checking the correctness of the information which it concludes the correctness of its information 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 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 fllows

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

Where k is the number of bits per data block which we wish to read as meta data. The function g generates for each 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. 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 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 k bit integer α_i for each i. This function is a secret and is known only to the verifier V.

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

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

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

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 verifiers data.

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 F^* is archived with the cloud. Figure 4 shows the encrypted file F^* after appending the meta data to the data file F .

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

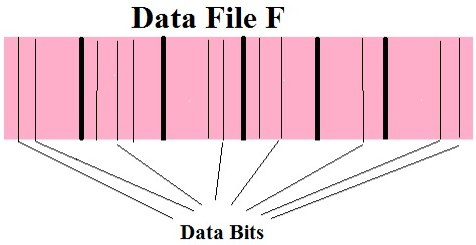

Figure 3 A data block with random bits selected in the file F

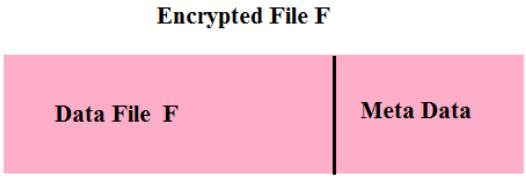

Figure 4 The encrypted file F which will be stored in the cloud.

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 α_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 at the cloud storage.

V. CONCLUSION

In this paper we have proposed ICCC approach to facilitate the client (user) in getting information correctness which he wishes to store in the cloud servers with bare minimum costs and efforts. Our approach was developed to reduce the computational and storage overhead of the customer 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

It should be noted that our approach applies to static storage of information and also dynamic storage information. Also the number of queries that can be asked by the client is fixed apriori. 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,