International Journal of Electronics Communication and Computer Engineering
Volume 5, Issue (4) July, Technovision-2014, ISSN 2249–071X

Technovision-2014: 1st International Conference at SITS, Narhe, Pune on April 5-6, 2014

Data Integrity Proofs in Cloud Computing

Deshmukh Ashwini B.
Deptt. of Computer Science and Engg.
M.B.E.S College of Engineering,
Ambajogai, Dist. Beed, Maharashtra
Email: ashwinid21991@gmail.com

Galphade Anuja H.
Deptt. of Computer Science and Engg.
M.B.E.S College of Engineering,
Ambajogai, Dist. Beed, Maharashtra
anujagalphade@gmail.com

Amruta N. Mulay
Deptt. of Computer Science and Engg.
M.B.E.S College of Engineering,
Ambajogai, Dist. Beed, Maharashtra
mulayamruta32@gmail.com

***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. 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.**

***Keywords* – Cloud Computing, Data Integrity.**

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 free faithfully store the data with it and provide it back to the owner whenever required. As data generation is far outpacing data storage it proves frequently update their hardware whenever additional data is created. Also maintaining the storages can be a difficult task. 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 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 project 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 irretrievability (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 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.

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 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.

II. RELATED WORK

Drata integrity proofs in cloud storage has drawn a lot of research interest and technique, with special emphasis on consistency and integrity of data in cloud storage.

[1]. 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. The simplest Proof of retrivability (POR) scheme can be made using a keyed hash function hk (F).

In this scheme the verifier, before archiving the data file F in the cloud storage, pre-computes the cryptographic hash of F using hk(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 archive and asks it to computer and return the value of hk (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 are the high resource costs it requires for the implementation.

[4]. Ari Juels and Burton S. Kaliski Jr proposed a scheme called Proof of irretrievability for large files using sentinels. 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 irretrievability 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. This small portion of the file F is in fact independent of the length of F. 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 archive. The use of encryption here renders the sentinels indistinguishable from other file blocks. This scheme is best suited for storing encrypted files.

[6].Internet is viewed as cloud computing hence the term cloud computing for computation done through internet. With cloud computing the users can store and access their data through internet without worrying about the local maintenance and management of data. Cloud computing is a general term for anything that involves delivering hosted services over the internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a- Service (PaaS) and Software-as-a-Service (SaaS). The data placed in the cloud is accessible to everyone, security is not guaranteed.

Now there was a problem that how to efficiently verify the correctness of the outsourced cloud data without the local copy of data files becomes a big challenge for data storage security in cloud computing. Downloading the data for verification is an expensive process. It makes the process much slower. To ensure security, cryptographic techniques cannot be directly adopted. Sometimes the cloud service provider may hide the data corruptions to maintain the reputation. To avoid this problem, we introduce an effective third party auditor to audit the user's outsourced data when needed.

III. Phases and Technique Use

The A data integrity proof in cloud based on selecting random bits in data blocks. 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 of its data stored in the client.

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

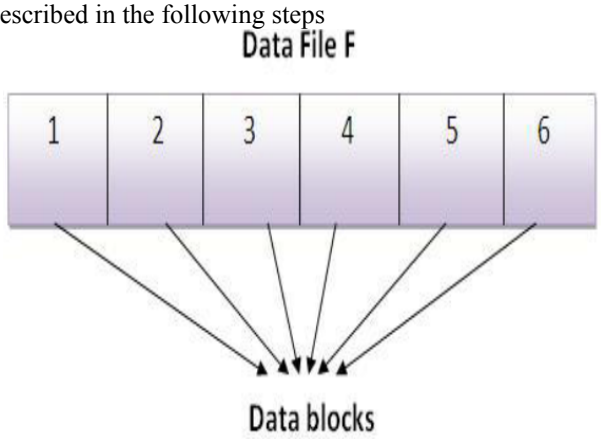


Fig.1. Data file F with 6 data blocks

Generation of meta-data:

Let g be a function defined as follows g (i, j) {1.m}, i 2 {1..n}, j 2 {1..k} (1) Where k is the number of bit 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 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 mi represent the k bits of meta data for the ith block. Figure 2 shows a data block of the file F with random bits selected using the function g.

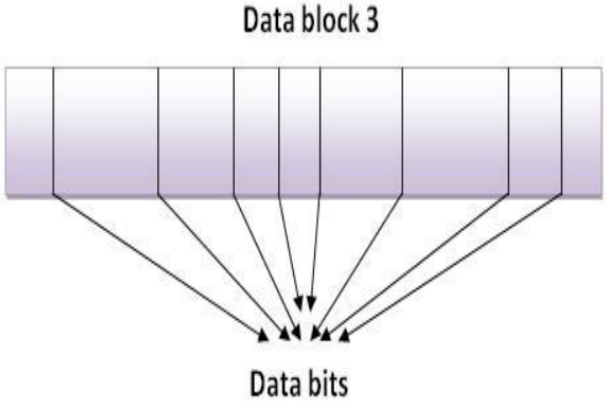


Fig.2. Data block of the file F with random bits selected.

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. 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 _i for each i. This function is a secret and is known only to the verifier V .h: i _i, _i 2 {0..2n} For the meta data (mi) of each data block the number _i is added to get a new k bit number M 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 verifier's data.

Appending of meta data:

All the metadata bit blocks that are generated using the above procedure are to be concatenated together. This concatenated metadata should be appended to the file F before storing it at the cloud server. The file F along with the appended metadata F is archived with the cloud. Figure 3 shows the encrypted file F after appending the metadata to the data file F. the encrypted file will stored in cloud with metadata.

Fig.3. The encrypted file.

V. Verification Phase

Let the verifier V 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 the verifier accepts or rejects the integrity proof. Suppose the verifier wishes to check the integrity of nth 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 i and the corresponding bit in this decrypted metadata 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.

VI. Proposed Work

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.

VII. System Implementation

Mysql:

Mysql is the most trusted and depended-on open source database platform in use today. As such, 9 out of the top 10 most popular and highly-trafficked websites in the world rely on Mysql primarily due to its ubiquity across heterogeneous platforms and application stacks and for its well-known performance, reliability and ease of use. Mysql 5.6 builds on this momentum by delivering across the board improvements designed to enable innovative DBAs and Developers to create and deploy the next generation of web, embedded and Cloud/Seas/DaaS applications on the latest generation of development frameworks and hardware platforms.

Connecting to Mysql database:

In java we have been provided with some classes and APIs with which we can make use of the database as we like. Database plays as very important role in the programming because we have to store the values somewhere in the back-end. So, we should know how we can manipulate the data in the database with the help of java, instead of going to database for a manipulation. We have many database provided like Oracle, Mysql etc. We are using Mysql for developing this application. In this section, you will learn how to connect the Mysql database with the Java file.

Firstly, we need to establish a connection between Mysql and Java files with the help of Mysql driver. Now we will make our account in Mysql database so that we can get connected to the database. After establishing a connection we can access or retrieve data form Mysql database.

Connection:

This is an interface in java.sql package that specifies connection with specific database and java files. The SQL statements are executed within the context of the Connection interface.

Class.forName (String driver)

This method is static. It attempts to load the class and returns class instance and takes string type value (driver) after that matches class with given string.

Driver Manager:

It is a class of java.sql package that controls a set of JDBC drivers. Each driver has to be register with this class.

Get Connection (String URL, String username, String password):
This method establishes a connection to specified database URL. It takes three string types of arguments like:
URL: Database URL where stored or created your database
Username: -User name of Mysql (root)
Password: -Password of Mysql (root)

Conclude ():

This method is used for disconnecting the connection. It frees all the resources occupied by the database.

PrintStackTrace ():

The method is used to show error messages. If the connection is not established then exception is thrown and prints the message.

VIII. RESULTS

Main Form:

This is the main form of cloud server. It contains sub forms like send file, upload file, include byte number, view file information, and download files.

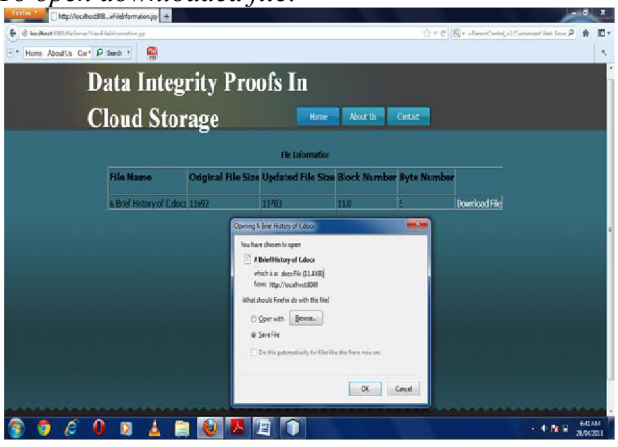
Upload file form:

The above form is used to upload file from client side on the cloud server, it includes file name with specified drive.

Form for entering byte number:

Uploaded file information form:

To open downloaded file:



IX. CONCLUSION

In this project 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 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 [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 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] Sravan Kumar, Ashutosh Saxena, "Data Integrity Proofs in Cloud Storage", *IEEE Conference 2011*.

[2] E. Mykletun, M. Narasimha, and G. Tsudik, "Authentication and integrity in outsourced databases," *Trans. Storage, vol. 2, no. 2, pp. 107–138, 2006*

[3] 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.*

[4] Juels and B. S. Kaliski, Jr., "Pors: proofs of retrivability 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.*

[5] 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.*

[6] Siva Ramakrishna, CH. Sandhya Rani "Providing Data Integrity for Dynamic Cloud Storage''*Bomma Institute of Technology And Science Allipuam, Khammam (dt).*