International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 4 161 – 165

Proofs for Integrity of Data in Cloud Storage

Roopa G. K [1] and Radhika Shetty D S [2]
[1] Asst. Professor , Department of Computer Science , VCET, Puttur.
[2] Asst. Professor, Department of Computer Science , VCET, Puttur.

Abstract:- Data is moved to a remotely located cloud server in cloud computing. The cloud stores the data and return back to the owner whenever it is needed. But there is no guarantee that data stored in the cloud is secured and not altered by the cloud or Third Party. In order to overcome the problem of integrity of data, the user must be able to use the assist of a third party. The third party has experience in checking integrity of data, that cloud users does not have, and that is difficult for the owner to check. The data in the cloud should be correct, consistent, accessible and high quality. The aim of this research is to ensure the integrity of data and provides the proof that data is in secured manner and to provide cryptographic key to secure the data in the cloud. The proposed approach has been implemented.

Keywords*: Data integrity, Cryptography, Cloud storage.*

*****

1. Introduction

Cloud storage is visualized pools where data and applications are stored which are hosted by the third party. Company, who desires to store their data in the cloud, buy or lease storage capacity from them and use it for their storage needs. Cloud storage offers benefits like reduction of costs, providing more flexibility, reduction of IT management of hardware and data, and providing greater storage capacity. Cloud also lacks in some of the issues like data integrity, data loss, unauthorized access, privacy etc.

Data integrity is very important and essential among the cloud storage issues. After moving the data to the cloud, owner hopes that their data and applications are in secured manner. But that hope may fail sometimes. The owner's data may be altered or deleted. In that scenario, it is important to verify if one's data has been altered with or deleted. To validate data, often a user must download the data. If the outsourced data is very large files or entire file systems, such downloading to determine data integrity may become prohibitive in terms of increased cost of bandwidth and time, especially if frequent data checks are necessary. This paper proposes a method that, owner need not download the data or files to check the integrity and provides the proofs that data is stored at a remote storage in the cloud is not modified by anyone and thereby integrity of data is assured. Some of the best examples for cloud storage are Amazon S3, Windows azure, EMC Atoms, FilesAnyWhere, Google app Engine etc.

The remainder of the research paper is organized as follows: Section two analyses about the cloud storage architecture and along with its characteristics. Section three of this paper briefly describes the proof of retrievability and role of third party auditor. Section four is explaining the existing system. Section five deals with how the data integrity is verified in the cloud. Section six shows the results of implementation. We concluded the paper in section seven.

2. Cloud Storage

The process of storing the data in the remotely located cloud servers is called cloud storage. The architecture of the cloud storage as shown in Fig.1.

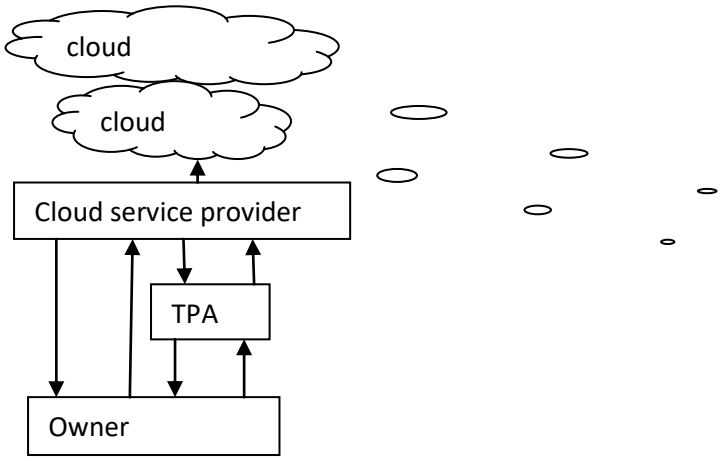

Fig. 1 Architecture of cloud storage

The cloud storage is better than all traditional storage methods because of the following reasons:

- ✓ Companies do not need to install physical storage devices in their own data center or offices.
- ✓ Storage maintenance tasks, such as back up are offloaded to the responsibility of a service provider.
- ✓ Companies need only pay for the storage they actually use.

3. Literature Overview

Storing of user data in the cloud has many interesting security concerns which need to be investigated for making it a reliable solution to the problem of avoiding local storage of data.

The POR scheme can be made by using keyed hash function hk(F). In this scheme, the verifier, before archiving the data file F in the cloud storage, precomputes the cryptographic hash of F using hk(F) and stores this hash as well as the secret key. To check if the integrity of the file f is lost the verifier releases the secret key K to the cloud and asks it to compute 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. 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 archive side, each invocation of the protocol requires 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 requires that each proof requires the prover 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.[1]

Ari Juels and Burton S. Kaliski Jr(2007) proposed a scheme called Proof of Retrievability(POR) for large files using “sentinels”. In this scheme unlike in key hash approach scheme, only a single key can be used irrespective of the size of the file or number of files whose retrievability it wants to verify. At the client only 2 functions are stored, the bit generator function g, and the function h used for encrypting the data. Hence the storage at the client is very much minimal compared to all other schemes that were developed. [4]

The ultimate challenge in cloud computing is data level security. In a data possession work(Ateniese et al., 2007) defined the “provable data possession”(PDP) model for ensuring possession of file on untrusted starages. Their scheme utilized public key based homomorphic tags for auditing the data file, thus providing public verifiability.

4. Existing system

In the existing cloud storage system, the owner want to check the data integrity, he need to access the entire file so its expensive to the cloud server. 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.

Disadvantages:

- ✓ The main drawback of this scheme is the high resource costs it requires for the implementation.
- ✓ Also computing hash value for even a moderately large data files can be computationally burdensome for some clients (PDAs, mobile phones, etc).
- ✓ Data encryption is large so the disadvantage is small users with limited computational power (PDAs, mobile phones etc.).

5. Proposed Scheme

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.

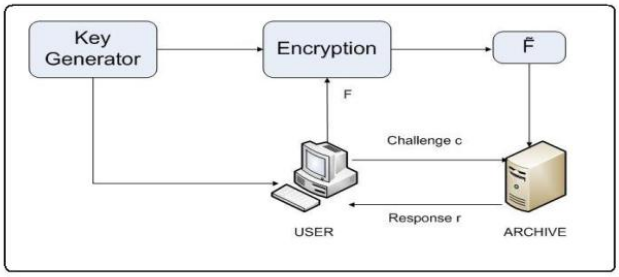

Fig. 2. Proposed architecture

Advantages:

- ✓ Reduction in storage costs and maintenance.
- ✓ Avoiding local storage of data.
- ✓ It reduces the chance of losing data by hardware failures.
- ✓ Not cheating the owner.

Algorithms used:

Meta-Data Generation:

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.

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. The encryption method can be improvised to provide still stronger protection for verifier's 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 e F is archived with the cloud.

Let g be a function defined as follows:

g(i, j) -> {1..m}, i€ {1..n}, j€ {1..k} ---(1)

Where k is the number of bits per data block which we wish to read as meta data. The function 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 j^{th} bit in the i^{th} data block. The value f 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 ith block. Figure 3 shows a data block of the file F with random bits selected using the function g.[1]

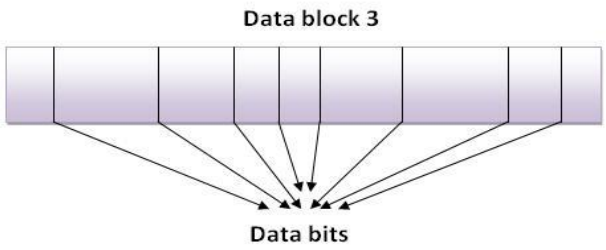

Fig 3. Random selection of bits

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 a k bit integer a_i for each i. This function is a secret and is known only to the verifier V .

h : i ->a_i, a_i€ {0..2n} ----(2)

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

$M_i = m_i + a_i$ ----(3)

In this way we get a set of n new meta data bit blocks. [1]

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 is archived with the cloud. Figure 4 shows the encrypted file e F after appending the meta data to the data file F.[1]

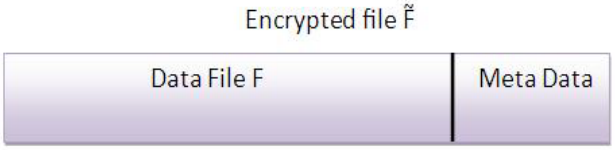

Fig. 4 Encrypted file stored in cloud server

Algorithm for cloud storage:

File is denoted as 'F'
Owner of data is represented as 'cc'
Cloud server is denoted as 'cs'
Security key is represented as 'Skey'
Begin
If Login==true
Then
Use the application for upload, download and sharing of files
If option==upload then
Upload F in 'cs'
Skey is sent to cc (user's mail id)
Else if option==download then
Enter Skey
If Skey==true then
F will be downloaded
Else print the message as wrong Skey
Else if option==sharing then
Check for F owned by other users
Else
Report:=invalid owner
Sign up as new user
End if

ISSN: 2321-8169
161 – 165

6. Results

The user has to login to the application by using username and password. If he is new user, he has to register by entering the details like email id, mobile number etc. The password is encrypted by using Rijindael algorithm(AES) and stored in the database.

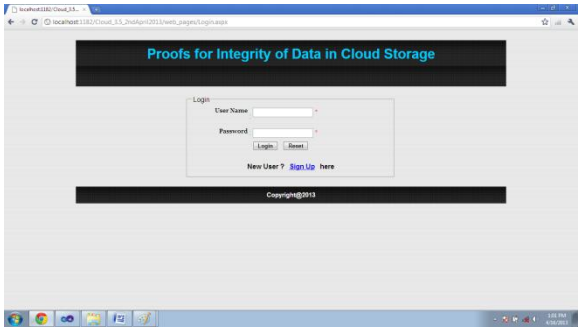

Fig. 5 User Login

The user can upload the file to the cloud archive by clicking on 'File Upload' option. The metadata has been appended to the file for security. We have taken the time of upload in terms of clock ticks and it is been added to the filename at the time of upload to ensure unique filename.

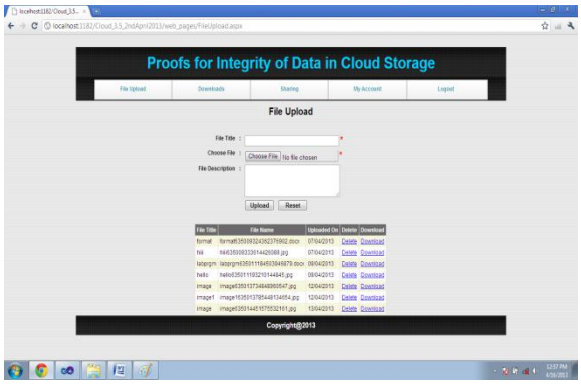

Fig. 6 File upload by the user.

The authorized users can download the file by using security key which will be sent to their mail id when any other user has shared the file with him. The owner also has to enter the key to download the file by checking mail.

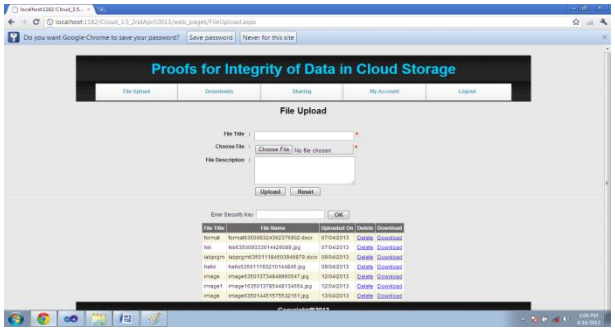

Fig. 7 File download by the user by entering key.

The owner will come to know if anybody downloads his file by clicking on 'Downloads'.

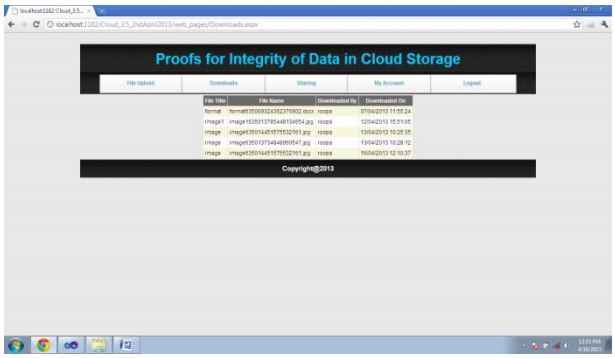

Fig. 8 Intimation for the owner about file download.

The owner of the file can share the files with other users who are already registered.

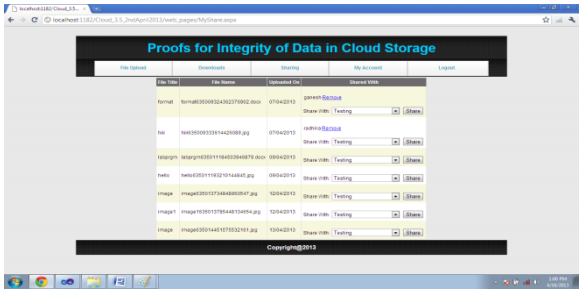

Fig. 9 File sharing with other users.

The owner can check if other users have shared any files with him. If so, he can download it.

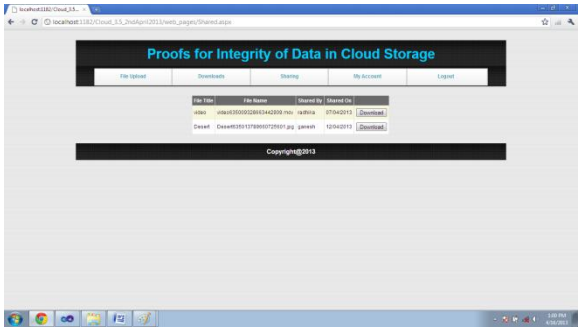

Fig. 10 Shared files

The user needs to check his e mail to obtain security key for downloading the files.

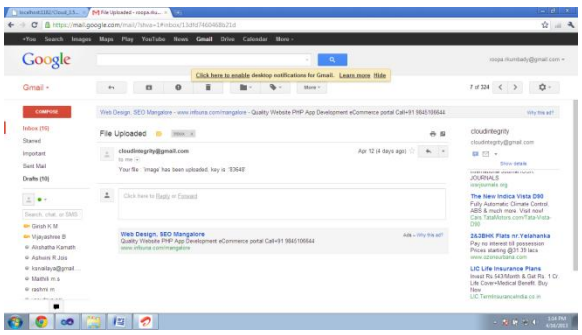

Fig.11 Security key is sent to mail id of user.

System Requirements:

Hardware Requirements:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

Software Requirements:

- Operating system : Windows XP.
- Coding Language : ASP.Net with C#
- Data Base : SQL Server 20**05**

6. Conclusion

The paper facilitates the client in getting a proof of integrity of the data in the cloud storage servers with minimum costs and efforts. This scheme was developed to reduce the computational and storage overhead of the client and to minimize the computational overhead of the cloud storage server. It minimizes the size of the proof of data integrity so as to reduce the network bandwidth consumption. At the client we 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 this 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 this project 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. The 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.

References

[1] Sravan Kumar and Ashuthosh Saxena, “Data integrity proofs in cloud storage”, 978-1-4424-8953-4/11,2011 IEEE

[2] E. Mykletun, M. Narasimha, and G. Tsudik, “Authentication and integrityin outsourced databases,” Trans. Storage, vol. 2, no. 2, pp. 107–138, 2006.

[3] D. X. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in SP ’00: Proceedings of the 2000 IEEE Security and Privacy. Washington, DC, USA: IEEE Computer Society, 2000, p. 44.

[4] 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.

[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.