Journal of Theoretical and Applied Information Technology
20th September 2014. Vol. 67 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DESIGN OF PRIVACY MODEL FOR STORING FILES ON CLOUD STORAGE

[1]AMBIKA VISHAL PAWAR, [2]DR. AJAY R. DANI

[1]Research Scholar, Symbiosis Institute of Technology (SIT), Symbiosis International University (SIU), Lavale, Pune - 412 115, Maharashtra State, India
[2]Research Guide, Symbiosis International University (SIU), Lavale, Pune - 412 115, Maharashtra State, India
Email: [1]ambikap@sitpune.edu.in, [2]ardani_123@rediffmail.com

ABSTRACT

Cloud Storages like Google drive, Dropbox are popular for personal and institutional file storage. Cloud storage is beneficial in terms of scalability, availability and economy. But cloud storage has privacy challenges due to which many users are reluctant to use it for personal data storage. Storage of personal or sensitive files should be done carefully. Since cloud storage is third party storage and also has many other possibilities like files can be shifted from one server to another cloud server. Cloud storage needs special solution than traditional third party storages. The data can be encrypted for security purpose but this will further raises issues like key management and key distribution. Encryption also requires more computation cost and time on client/user. This paper introduces the design of novel privacy model for storing files in cloud storage, proposes architecture and algorithms for cloud storage. Paper also discusses about the proof of privacy and proposed system's performance evaluation parameters for future work.

Keywords: *Cloud Storage, Files, Privacy, Multicloud, Splitting, Information Dispersal Algorithm, Security, Randomization*

1. INTRODUCTION

Cloud computing has been growing tremendously in last few years. Among different computing services offered in cloud, cloud storage has attracted many users. Cloud storage allows organizations or individuals to store their data in scalable, anytime and anywhere available and affordable (pay as you go) powerful cloud infrastructure. Extensive use of Google drive, Dropbox and many other cloud storages proved popularity and need of cloud storages. E.g. In Apr 2013Amazon reported storing two trillion objects in their storage Amazon S3 [1].

Cloud storage i.e. third party storage has critical issues for adoption, to store sensitive data. Organization losses control over their sensitive data, while using third party cloud storage. Compromising customer data can impact many organizations, like healthcare, retail companies, manufacturing etc. Cloud storage creates many challenges in ensuring privacy of outsourced data since cloud service provider (CSP) may use user data for commercial purpose or third party may gain access to cloud storage user data. Such problems are actually faced and confessed by some service providers e.g. Web-based storage firm Dropbox confirmed that a programmer's error caused a temporary security breach that allowed any password to be used to access any user account. [2]

To provide data privacy, most of the cloud storage security solutions use encryption. Customer encrypts the data before storing it on cloud and keeps decryption key with them. However, encryption has computational complexity and complex key management issue. If encryption has computation overhead instead of encrypting file at client side, file can be encrypted by CSP to protect from outside attackers.

Other solutions to provide security and privacy are using information dispersal algorithms (IDA). There are many algorithms proposed on IDA, it addresses different issues as confidentiality, integrity and availability. IDA splits file into parts/chunks and apply further transformation to generate n slices of file. It further distributes generated slices among multiple CSPs. Out of n slices, if m number of slices (m<n) are available, it is possible to extract original file contents. If adversary gets access to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

m number of slices, possibility of extracting sensitive information in stored file.

One more approach to security and privacy solutions is use of multi-cloud environment. Multi-cloud environment means simultaneous use of multiple cloud services for the same purpose e.g. for storage of data. Many solutions using multi-cloud environment have been proposed, uses encryption and IDA based algorithms in order to provide security, privacy, availability, reliability in cloud. These approaches have proved improvements with this new paradigm shift from single cloud to multicloud architecture. So we have selected here an approach of using multiple clouds instead of single cloud.

The rest of the paper is organized as follows. Section 2 gives an overview of related research work. In Section 3 privacy problem is defined with the help of basic cloud storage architecture, also the threat model and objectives of work are detailed; Section 4 describes the approach to solve the defined privacy problem. Solution is proposed with the help of design of privacy model and detail design of algorithms for file operations in cloud storage. In section 5 presents the proof of privacy for proposed privacy model. Section 6 describes future work, discusses different metrics for performance analysis. Finally in section 7 conclusions.

2. RELATED WORK

Motivations to design this approach are IDA proposed in [3] and the survey and design of different multi-cloud architectures (MCA) discussed in [4]. Reasons behind adoption of multi-cloud architecture are proof of advantages to resolve many single cloud architecture issues. Multicloud architecture lowers the risk of malicious data disclosure, manipulation and tampering. Thus by use of MCA, the trust assumption of trust on single CSP lowered to assumption of non-colluding CSPs. Further this architecture will make it difficult for attackers to retrieve scattered data over multiple CSPs.

[4] Presents different MCA for improving security and privacy. We selected MCA based on data partitioning. Data partitioning divides original data into small parts. These data parts can be further stored on different CSPs, thus no single CSP will have a complete data, which hides the relation of CU/data owner to the sensitive data by not exposing data as a whole to the third party CSP. But still it will revile part of data to CSPs. We need to make some provision to resolve this issue; detail solution is discussed in section 4.

Further in [4] different data partitioning methods are discussed. Data can be either in structured form e.g. relational database or unstructured form e.g. doc, PDF files. Files typically contains images, text etc. In databases data get organized in the form of rows and columns. Vertical and horizontal partitioning of databases has been applied by many of the researchers. File splitting and merging is very popular technique to overcome different storage related problems. Many tools are available for file splitting and merging of files [5] explains the fundamentals and implementation of file splitting and merging.

Cryptographic approach for data partitioning is used in different ways. Data can be encrypted before storage to protect it. While doing this cryptographic keys need to be maintained on client/user side. But to allow sharing of data among multiple users, the key should be available online [6]. Many solutions are proposed using encryption for secure cloud storage. The fist solution using cryptograph for cloud storage [7] is for encrypted key-value cloud storage. This approach supported easy access to the stored data in secure cloud storage. Searchable encryption [8][9] is the heart of this solution. Searchable encryptions allows to search even on the encrypted data. It allows keyword search on a data for authorized tokens of the keyword. It uses public and private cloud infrastructure. The encrypted data resides in untrusted public cloud environment while the keys are stored in trusted private cloud environment.

CryptDB[10] supports data processing on encrypted relational databases. The data is encrypted using different types of encryptions as order-preserving encryption [11], homomorphic encryption [12], searchable encryption [8] and AES. In CryptDB, there is a database server which stores encrypted data and a proxy which holds the keys and also provides a user interface for running SQL queries. Proxy provides only necessary keys to the server. This may cause the database server to gradually learn about the encrypted data.

One more way is a secret data sharing. It splits data into multiple shares such that original data

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

can be reconstructed if and only if minimum required shares come together.

Above cryptographic approaches are mostly applied in relational databases for storage and processing of data in cloud storage.

In distributed file system files get divided into chunks and get distributed to different servers. Information dispersal algorithms are proposed in [3] to support implementation of distributed file system and also provide data confidentiality, integrity and availability.

3. PROBLEM DEFINITION

3.1 Cloud Storage Architecture

In typical cloud storage scenario as shown in Figure 1, there are 2 main entities

1. Cloud Service (storage) Provider (CSP) e.g. Amazon provides Amazon S3. It provides storage as a service.
2. Cloud (storage service) User (CU) e.g. an individual or organization using cloud storage service. Multiple users may exist who want to access same file. The file is stored/uploaded and accessed/downloaded into/from CSP.

Two basic operations are uploading and downloading file (F) on/from cloud storage.

3.2 Problem Statement

As CU's sensitive data (files) resides with third party CSP. Privacy can be violated by CSP, if files are stored in plain text or in original form. Privacy violation here we mean accessing/viewing, sharing sensitive information in uploaded file e.g. customer and their contacts in file. Privacy of the cloud user's uploaded sensitive file. To protect privacy means CSP/outsider unauthorized user should not be able to read CU's sensitive data without their permission. So we cannot keep plain data on CSP. Privacy can also be violated by outsiders i.e. attackers who gain access to CSP storage. Once accessed sensitive information can be misused, shared or sold for benefit.

Privacy (CU, sensitive info file) =! CSPs or outsider (reconstruct original file)

Privacy (data subject, sensitive info) =! CSPs or outsider (link, data subject, sensitive info)

3.3 Threat Model

We assume that all servers in CSP network are strongly protected against network attacks e.g. firewall. We assume that adversary CSP, is capable of monitoring and observing n/w traffic in and out of CSP server network. Assumption of complete secure CU network is made. Since application server is single point of failure in our system and cannot be compromised. It is also assumed that all communications between system components are secured.

3.4 Objectives

Privacy protection model is designed to achieve following objectives

1. Design of privacy protection scheme for unstructured data e.g. files. This will enable popular storage approach of using cloud storage. Protecting privacy of sensitive files (by making file unreadable.).
2. Design scheme as a middle layer between CSP and CU without making any change to existing CSP infrastructure.
3. Utilizing multi-cloud environment which will improve privacy and availability.
4. Less computation and communication overhead on CU.

Two basic operation models in cloud storage are as below:

1. Storage as a service: CSP acts as a storage service provider. F is created and opened at CU. CU uploads F into CSP storage. To access uploaded file F, CU downloads F from CSP. CU will download F from CSP to modify it locally and re-upload it to CSP storage.
2. Computing as a service: CSP provide file creation as well as other file operations.

We consider the first operation in this paper. In the second operation, CS needs to be fully trusted. In second operations privacy or security problem is trivial.

4. PROPOSED APPROACH

4.1 Design Of Privacy Model for Storing Files in Cloud

Figure 2 shows the design of privacy model for storing files in cloud storage. This model has three components: 1. Multi-cloud environment

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(CSPs), 2. Cloud Users and 3. Client side, application server.

Description of the components:

Multi-cloud environment: Set of cloud storage service providers e.g. Sugar Sync, Dropbox and Google drive etc., to store file chunks.

Cloud User: User will interact with CSPs through application server. Cloud users create files locally, store and share and retrieve files on/from cloud storages. User will access web application hosted on application server, discussed below. This web application will provide user interface to store/upload files to CSPs, retrieve/download files from CSPs and also share files with other users.

Client side Application Server: Web application and database with file metadata (complete file preprocessing information as file name, id, chunks, chunk groups, chunk dada randomization key, chunk sequence randomization key and chunk destination addresses CSPs). Pre-processes file upload request by user.

Pre-processing of upload request has following steps:

Step 1: Splitting a file into chunks of fixed size.

Step 2: Generate chunk sequence randomization key. Store the key in repository. Randomize chunk sequence.

Step 3: Generate data randomization key, store the key in repository. Randomize the chunk data with randomization key, makes data non readable.

Step 4: Grouping of Chunks based on no. of CSPs and sending group to different CSPs. E.g. uploading first N chunks to CSP1 and so on.

Step 5: Generate random redundancy key and creating redundant chunks.

Step 6: Storing redundant chunks on selected CSPs to guarantee availability in absence of any of the CSPs.

Pre-processing of file download request has following steps:

Step 1: Fetch file to chunk mapping information from file-chunk mapping repository.

Step 2: Fetch destination (CSPs) addresses of all file chunk groups from repository.

Step 3: Send download request based on destination addresses to different CSPs.

Post-processing of file download request has following steps:

Step 1: Receive chunk data from different CSPs.

Step 2: Check for any of the chunk failure and request for redundant copy at other CSP.

Step 3: Normalize/arrange chunk data with data randomization key from repository.

Step 4: Re-order (sort) chunks with chunk sequence randomization key stored in repository.

Step 5: Re-assemble the file using its chunks.

Step 6: Send file to requested cloud user.

The application server is the heart of the system and cannot be compromised at any point. Server should be protected with powerful infrastructure such as firewall.

4.2 Algorithms

In this section we propose algorithms for upload download operation. We list here notations used in the algorithms in Table 1.

Table 1 Notations

F	File/Data to be uploaded
FN	File Name of F
FS	File Size of F (in bytes)
CSPs	Cloud Service Providers
N	No. of CSPs
CS_id	Cloud Storage Id
CU	Cloud User
FC	Set/Array of file chunks
fc	File Chunk of F
fc_Size	File Chunk Size (in byte)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

n	No. of Chunks
K_1	file chunk sequence randomization key
K_2	chunk data (bytes) randomization key
G	Chunk Group
GID	Group ID

Algorithm 1: File Splitting Uploading Operation

Input: F, n, N

Output: 1.Sending 'fc' groups to destination CS_id

Step 1: Find File Size

FS = Sizeof(FileName) //finding file size in bytes

Step 2: Make n Chunks of file

Where fc_Size = FS/n

Step 3: Generate K_1 (file chunk sequence randomization key). Randomize file chunk sequence with this key. Store K_1 in repository

K_1=GenerateRND();

Randomize (FC, K_1)

Step 4: Generate K_2 (chunk data randomization key). Randomize every chunk data (bytes) using K_2. Store K_2 in the repository.

K_2=GenerateRND();

m=fc_size

Let each chunk has m bytes fc_1= {b_1, b_2….. b_m}

//Randomize chunk data of all generated chunks.

For i=1 to n

Randomize(fc_i, K_2)

Step 5: Generate N groups of n chunks in FC. Store Chunk GIDs in repository.

FC = {G_1, G_2,…,G_N}

Step 6: Send N chunk groups to N CSPs. Store group and their destination CSP id info in repository.

For all i=1to N groups G_i

//In Parallel send j^{th} chunk of G_i to CSP_i

for j=1 to n/N

$G_i[C_j]$->CSP_i

Algorithm 2: File Combine Downloading Operation

Input: FN

Output: Downloads File F with FN on client side

Step 1: Map FN, file name to chunk groups from repository.

Step 2: Fetch chunk group and their destination CSP id info from repository. Now send parallel request to different CSPs for chunks in respective groups.

FC= {G_1, G_2,…,G_N }

//In Parallel request j^{th} chunk of G_i to CSP_i

For all i=1to N groups G_i

For j=1 to n/N

$G_i[fc_j]$<-CSP_i

Step 3: Fetch K_2 (chunk data randomization key) from repository. Rearrange file data (bytes). m=fc_size

Let each chunk has m bytes e.g. fc_1={ b_7, b_m….. b_2}

//Rearrange chunk data of all generated chunks.

For i=1 to cn

Rearrange (fc_i,K_2)

After randomize fc_1={b_1, b_2….. b_m }

Step 4: Fetch K_1 from repository (File chunk sequence randomization key). Rearrange file chunks with this key. FS

Journal of Theoretical and Applied Information Technology
20th September 2014. Vol. 67 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Rearrange (FC, K_1)

After rearranging function, FC = {fc_1, fc_2,... fc_n }

Step 5: Generate File F with FC

FS=Sizeof (FileName)

Compare with repository File size

Step 6: Send/ Display file to CU

4.3 Example

Let file F be the original file of size FS and b_i be the byte array of F.

Let m be the size of file chunk. (F will spitted into chunks of fixed size m bytes)

F = (b_1, b_2,…b_m),(b_{m+1}, b_{m+2},…b_{2m}),…(b_{FS-m+1},….b_{FS})

Let generated file chunks are fc_1=(b_1, b_2,…b_m), fc_2=(b_{m+1}, b_{m+2},…b_{2m}),…fc_n= (b_{FS-m+1},….b_{FS}) where n = FS/fc_Size

So File chunk array, FC = {fc_1, fc_2,…., fc_n }

Randomize function file chunk array FC, e.g. FC = {fc_6, fc_{13}, fc_n,....fc_2 }

Randomize m bytes of all file chunks e.g After randomize fc_1={ b_7, b_m….. b_2}

Generating N groups of n file chunks fc in file chunk array FC

FC = {G_1, G2…G_N}

Storing Chunk Groups to different CSPs.

E.g. Let F is a file of size 100 bytes. Assign 10 bytes to each chunk and step to create an array of chunks as shown in Figure 3.

Let RNDGen function generates a random key, k_1= 5217380694. This key will be used to randomize chunk array sequence so randomized chunk sequence according to this key will be as below.

Randomized Chunk Array Sequence: [5] [2] [1] [7] [3][8][0][6][9][4]

Grouping:

[5] [2] [1] [7] [3] | [8] [0] [6] [9] [4]

G_1 **G_2**

Assuming here no. of CSPs, N=2. Creating unordered groups of chunks and sending/storing these groups G_1 and G_2 with different CSPs.

Similar to chunk sequence randomization, we will randomize chunk data. As shown in figure 3 chunk size is 10 bytes for each chunk. So we will randomize each file chunk bytes using same randomization key. E.g. chunk data randomization key k_2=8315207964. Byte sequences in original chunk and after data randomization are as shown in figure 4 and 5 respectively.

4.4 Application Server Repository

Application Server Repository stores information for processing before uploading and processing after downloading the file. It store data about authentic users created by administrator, who can use this interface to upload/download file on multiple CSPs. The structure for storing user information is as given below.

Structure User
{
UserName
MD5 hashed password
}

Then it also stores file information in below mentioned format. One user can create many files and further that user/file owner can share the same file with other users.

Structure File_Info
{
File_Name
File_Id
File_Type
File_Owner
File_Creation_Time
File_Access_Time
File_Size
File_Chunk_Seq_RND _Key
File_Chunk_data_RND_Key
File_No_of_Chunks
Shared_with
Access_rights

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

}

Chunk/File parts related information will be stored in repository. Single file will have many chunks. This information will get stored during upload processing and will get retrieved during download processing.

Structure Chunk
{
Chunk_Name = File_Name_Chunkseq
Chunk_id=File_id_Chunkseq
Destination_addr=Cloud_storage_id
Redundant_copy_addr=Cloud_storage_id
Chunk_Group_Id
}

Apart from above mentioned information repository will have CPS related information in the following format.

Cloud_Stoarge
{
Cloud_Stoarge_Name
Cloud_Storage_id
Stoarge_Type
User_Name
MD5_Hashed_Password
}

5. PROOF OF PRIVACY

Privacy can be defined as "Minimizing probability of reconstruction or reassembling of original file by CSPs in order to understand sensitive information in file."

Following cases discusses different scenarios from best to the worst cases:

Case 1: An individual CSP don't have access to all chunks/parts of file. So probability of generating whole/complete file is zero.

Case 2: If all N CSPs collude then they will have all chunks/parts of file. i.e. all n chunks will be available to CSPs.

Without K_1 and K_2, they (CSPs) won't be able to reconstruct original file. So even if all N CSPs collude probability is not one.

Probability in Case 2 depends on the total no. of chunks and the chunk size. Assembling a file will require arranging chunk data in proper order and also arranging chunks in a proper sequence.

As considered in above section fc_size – file chunk size and n- total no. of chunks.

Let us assume that the chunk size is m bytes and total no. of chunks n. So possible permutations of m bytes = m! and permutations of n chunks are n!

So the probability of assembling original file in case 2 scenario is, P (A) =1/ (m! * n!). This probability will be very less, tends to zero for large values of m or value of n by basic limit theorem as shown in following equation 1. If we fix no. of chunks then chunk size will vary for different files. Larger the chunk size less will be the probability of reorganizing original data. Thus we will keep n constant and m variable. Thus for large files, it will provide strong privacy compare to small files. Even for small files, number of bytes will be large number.

$$P(A) = \lim_{\substack{n\to\infty \\ or \\ m\to\infty}} \left(\frac{1}{m! * n!}\right) \to 0$$

Equation 1: Probability of success in assembling file

Case 3: If someone gets access to client side application server then probability of original file construction will be one. Since an application and keys will be accessible.

Representing probability of assembling file on a probability scale, as discussed in above with 3 cases is as shown in Figure 6. As per probability theory probability zero tells us that the event will never happen thus case 1 gives probability zero i.e. it is impossible to assemble a file. Case 2 has not certain probability. It will be between zero to one. In case 3 file assembling is certain

The possibility of third case is rear rather not possible because of secured cloud user side infrastructure. Thus it is not possible for CSPs to assemble the original file easily. Thus without reconstruction of original file, one cannot link sensitive contents of file.

6. FUTURE WORK

Performance evaluation of proposed scheme, need to focus on quantitative results. The performance of proposed scheme focuses on two major operations on file while using cloud storage. In proposed model, time require to upload/ download file with respect to change in file size. File size is major independent variable in

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

our system, which will affect the file upload/download time. There are two minor parameters no. of chunks and size of chunk which can be observed during evaluation. With respect to above, we plan to measure three different metrics with respect to file size. First metric is file upload and download time (in msec) with respect to size of file.

Upload Time/Download Time is time from request to upload/download to complete file get uploaded/downloaded.

Upload Time = File Splitting Time + Total Chunks Upload Time

Download Time = Total Chunks Download Time + File Reassembling time

Second metric is communication cost; we need to observe additional communication overhead due to the multicloud environment and chunk distribution. We need to measure communication cost

Communication Cost = Data transfer in bytes/sec

Third metric is computation cost. Application Server/Client side computations overhead i.e. upload/download pre/post processing computations.

Client side Computation Cost = Processing time for upload processing + download pre & post processing.

In future we plan to do statistical analysis of our system with file size as independent variable and uploading and downloading time, communication cost and computation cost as dependant variables.

7. CONCLUSION

The paper has presented a novel approach for achieving privacy of non structured data i.e. file while using third party cloud storage service. Based on our analysis of existing security and privacy solutions on this problem, we proposed novel approach to solve privacy issue without use of encryption, which has heavy computation overhead. Further we will implement the proposed scheme and will analyze it for proposed performance metrics.

REFERENCES

[1] Amazon S3 - Two Trillion Objects, 1.1 Million Requests / Second 18 Apr 2013 in AmazonS3"http://aws.amazon.com/blogs/aws/amazon-s3-two-trillion-objects-11-million-requests-second/".

[2] Dropbox confirms security glitch--no password require 20 June 2011 "http://www.cnet.com/news/dropbox-confirms-security-glitch-no-password-required/".

[3] Rabin, Michael O. "Efficient dispersal of information for security, load balancing, and fault tolerance." *Journal of the ACM (JACM)* 36, no. 2 (1989): 335-348.

[4] Bohli, J-M., Nils Gruschka, Meiko Jensen, Luigi Lo Iacono, and Ninja Marnau. "Security and privacy-enhancing multicloud architectures." *Dependable and Secure Computing, IEEE Transactions on* 10, no. 4 (2013): 212-224.

[5] File Splitter (Part 1 Of 2 Parts) Started by Luthfi, May 02 2012.

[6] F. Pagano and D. Pagano, "Using In-Memory Encrypted Data- bases on the Cloud," Proc. First Int'l Workshop Securing Services on the Cloud (IWSSC), 2011, pp. 30-37.

[7] S. Kamara and K. Lauter, "Cryptographic Cloud Storage," Proc. 14th Int'l Conf. Financial Cryptography and Data Security, 2010, pp. 136- 149.

[8] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Con- structions," Proc. 13th ACM Conf. Computer and Comm. Security, 2006, pp. 79-88.

[9] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable Encryption Revisited: Consistency Properties, Relation to Anon- ymous IBE, and Extensions," Proc. 25th Ann. Int'l Conf. Advances in Cryptology (CRYPTO '05), 2005, pp. 205-222.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[10] R. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing," Proc. 23rd ACM Symp. Operating Systems Principles, 2011, pp. 85-100.

[11] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill, "Order- Preserving Symmetric Encryption," Proc. 28th Ann. Int'l Conf. Advances in Cryptology: The Theory and Applications of Cryptology (EUROCRYPT '09), 2009, pp. 224-241.

[12] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. ACM, vol. 21, no. 2, 1978, pp. 120-126.

Journal of Theoretical and Applied Information Technology
20th September 2014. Vol. 67 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

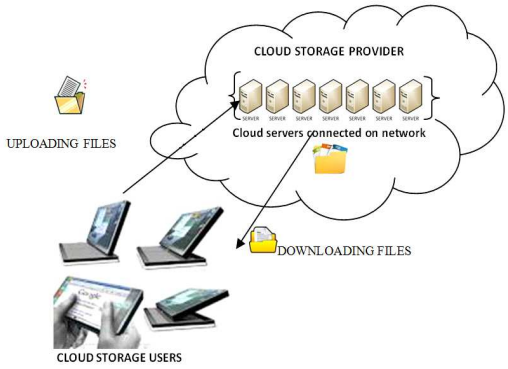

Figure 1 Cloud Storage Architecture

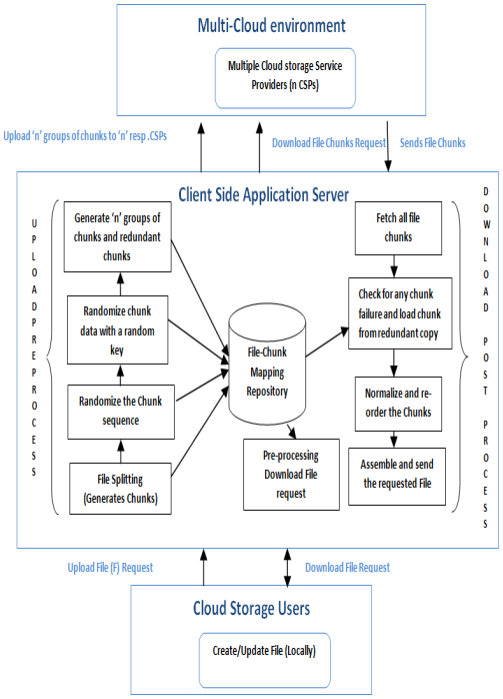

Figure 2 Privacy Model for Storing Files in Cloud

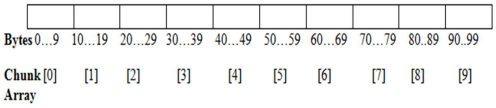

Figure 3 File data and chunk representation

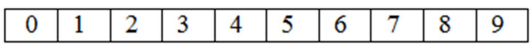

Figure 4 Byte sequence in original chunk

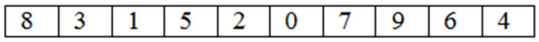

Figure 5 Byte sequence in data randomized chunk

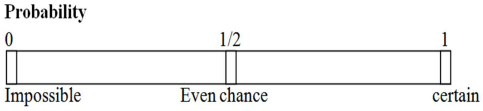

Figure 6 File Assembling Probability Scale