ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 4, June 2013

An Effective Data Storage Security Scheme for Cloud Computing

Kalpana Batra[1], Ch. Sunitha[2], Sushil Kumar[3]

Student M.Tech , Dept. of CSE, SGT Institute of Engineering and Technology, Gurgaon, India[1]

Assistant Professor, Dept. of CSE, SGT Institute of Engineering and Technology, Gurgaon, India[2]

Assistant Professor, Dept. of CSE, SGT Institute of Engineering and Technology, Gurgaon, India[3]

ABSTRACT: Cloud computing has been envisioned as the next generation information technology architecture for enterprises. Cloud Computing moves the data on the cloud storage servers maintained by service providers, which deprive the user of their control of the physical possession of data, even though they are the owners of the data. This unique feature however has raised many new security challenges which have not been well understood. In this paper we focus on cloud data storage security, which is an important aspect of Quality of service. We introduced an effective and flexible distribution verification mechanism to address data storage security in cloud computing. In this mechanism, we rely on token pre-computation to verify the correctness coded data rather than Pseudorandom Data. We present the system with design goals of Storage correctness and Fast Localization of Data Errors.

Keywords: Data Security; Cloud Computing; Storage Correctness, Fast Localization of data errors.

I. INTRODUCTION

Cloud Computing (CC) is an emerging computing paradigm that can potentially offer a number of important advantages. One of the fundamental advantages of CC is pay-as you-go pricing model, where customers pay only according to their usage of the services.[8].Cloud Computing is an internet based computing. It dynamically delivers everything as a service over the internet based on user demand, such as network, operating system, storage, hardware, software and resources. These services are classified into three types: Infrastructure as a service (Iaas), Platform as a service (Paas), and Software as a Service (Saas). Cloud Computing is deployed as three models such as Public, Private and Hybrid Clouds. [1].

Outsourcing data to a remote Cloud Service Provider (CSP) is a growing trend for numerous customers and organizations alleviating the burden of local data storage and maintenance. At the same time, when one's data are outsourced, he want to know whether the data is truly stored at the correct servers and be intact as stated in the Service Level Agreement (SLA). The pioneer of Cloud Computing vendors, Amazon Simple Storage Service and Amazon Elastic Compute Cloud [2] are both well known examples.

From the perspective of data security, Cloud Computing poses new challenges security threats for many reasons. Firstly, traditional cryptographic primitives for the data security protection cannot be directly adopted because user has no control over the data under Cloud Computing. Therefore verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. Secondly, Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion, deletion, modification, append etc. Last but not least, the deployment of cloud computing is powered by data centres running in simultaneous, distributed and cooperated manner.

Recently, the importance of ensuring the remote data integrity has been highlighted by the following research works [3]-[7]. These techniques, which can be useful to ensure the storage correctness cannot address all the security threats in cloud data storage, since they all are focusing on single server scenario. In this paper we propose an effective and flexible distribution verification mechanism to ensure the correctness of data in cloud.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 4, June 2013

II. PROBLEM STATEMENT

A. System Architecture

Representative system architecture for cloud data storage is illustrated in Figure 1. In cloud data storage, a user stores his data through a cloud Service Provider into a set of cloud servers, which are running in a simultaneous, cooperated and distributed manner. Web interface provides facilities to upload, download and perform the dynamic operations on data file. Authentication System allows the authorized user to get the access to the cloud storage. User Authentication Procedure is done prior to the storage and retrieval. It provides the data privacy to the user. It also allows user to change his profile. File explorer provides the options to search a file and perform dynamic operations. With the Block operation, it provides various operations like update append and delete on particular server. We can view the contents of the file stored on the particular server. With the upload option, user can upload the data file from his system on the cloud servers. With the download option, it will list down all the users file, and selected file can be downloaded on user's site.

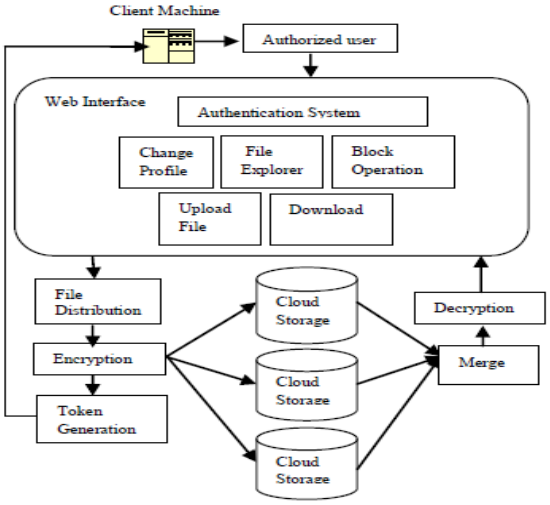

Fig 1: System Architecture for Cloud Data Storage

B. Design Goals

To ensure the security for the cloud data storage, we to design an efficient mechanisms for dynamic data verification and operation and achieve the following goals: (1) Storage correctness: to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud. (2) Fast localization of data error: to effectively locate the malfunctioning server when data corruption has been detected. (3) Dynamic data support: to maintain the same level of storage correctness assurance even if users modify, delete or append their data files in the cloud.

III. RELATED WORK

In cloud data storage system, user stores their data in the cloud and has no longer possessed the data locally. Thus the correctness and availability of the data files being stored on the distributed cloud server must be guaranteed. One of the key issues is to effectively detect any unauthorized data modification and corruption. Besides, in the distributed case when such inconsistencies are detected, to find which server the data error lies in is also great significance since it can be the first step to fast recover the storage errors. To address the problem, our main effective data storage scheme is presented in this section. In File distribution [2], the file is divided into the blocks and disperses the file F redundantly across a set of distributed servers. Data is stored in the encrypted form on the servers. All the dynamic operations like insert, update, append and delete can be performed on the data blocks. Accordingly the changes in file, data blocks are divided and stored on the data servers again. While retrieving the data, the data blocks of respective files are merged and return to the user. To check the correctness of the file, challenge tokens have been sending to the cloud storage system.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 4, June 2013

Based on this it receives whether the file is correct or modified by the unauthorized user.

IV. EFFECTIVE CLOUD DATA STORAGE SCHEME

I. *Following are the steps involved in data storage security scheme:*

A File Distribution Preparation

F – The data file to be stored. We assume that F can be denoted as a matrix of m equal-sized data vectors, each consisting of *l* blocks. Data blocks are all well represented as elements in Galois Field GF(2^p) for p = 8 or 16. File is split into fixed-size blocks which are stored on servers and all blocks are replicated and dispersed over distributed servers. We rely on this technique to disperse the data file F redundantly across a set of n = m+ k distributed servers. File data should be greater than or equal to the number of server on which it is going to store.

Step1. Check the file's data length greater than or equal to the number of server. If it is blank file or the data length is less, then prompt the message to the user.

Step2. Generate the file matrix. The *Generate_Matrix(file)* function divides the file into equal data vector length. Data vectors are created depending on the distribution of the file.

$$datavector\_length = total\ number\ of\ bytes\ in\ file\ /\ number\ of\ servers \quad (1)$$

If $file\_length \bmod n <> 0$ then

$$datavector\_lengh = datavector\_length+1 \quad (2)$$

This divides the file into n number of blocks. Vector1 contains the data bytes from 1 to *datavector_length* from the file. Vector 2 contains the data bytes from the location of *datavector_length* to (*2 * datavector_length*) and so on. Each vector represents the data block of matrix of Galois Field. Use the Vandermonde matrix for the Matrix_Multiply() function.

Step3. Perform Matrix *multiply (vector, A)* operation on each data vector. Use $g1, g2, g3 \ldots g_m$ as data vector names.

```
sum = 0; count = 0;
For 1 → k to File_length Do
For 1→ i to datavector_length do
      sum = sum +vector[i] *A[count];
        For all data vectors do
        If k = count Then Gm [i] ← sum;
              end for
         count = count + 1;
          End For
End For
```

Step4. Encryption is applied on each data vector and data vectors are disperse on the distributed servers. The data vector of a file is replicated on the cloud servers.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 4, June 2013

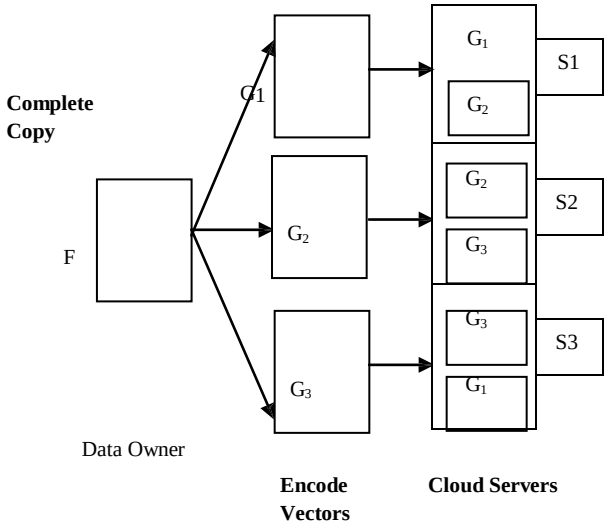

Fig 2: Distribution of File Stored on Cloud Storage

B. Token Pre-Computation

Before storing the data vectors on cloud servers, user pre-computes the verification tokens. These tokens are used to achieve assurance of data storage correctness and data error localization simultaneously. Main idea is as follows: before file distribution user pre-computes the short verification tokens on individual data vectors. When user wants to check the correctness of the data, he sends the challenge tokens to the cloud servers. User may send challenge on particular data block also. Upon receiving challenge token, each cloud server computes the token on the data vector, and sends them to user. If the tokens of users and computed tokens from cloud servers are matched, then the data is correct. If the tokens are not matched, then a modification has been done in the data by unauthorized user. This shows cloud sever misbehaving.

Step1. Choose the parameter r for number of indices per verification and pseudorandom function *f*, pseudorandom Permutation function ϕ

Step2. Choose the parameters *t*, the number of tokens to be generated and *l* the length of the data vector.

Step3. Chose α as a vector to compute the tokens and V to store the tokens.

Step4. Read the file as byte array. Divide this byte array into parts. Each part contains byte and represents it into polynomial. (for example dividing byte array in 3parts).

```
For j←1 to n   Do
For i ← 1 to t  Do α[i]
        = f(i) sum= 0
For q ←1 to  r do a←gj[ϕ(i)]
b ←Power (α[i], q)


        sum = sum + a * b
End For
       if   j = 0 then V1[i] = V1[i] + sum
        if  j= 1  then V2[i] = V2[i] + sum
        if j= 2  then V3[i] = V3[i] + sum
     End For
```

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 4, June 2013

End For
$V_j \leftarrow$ Add V1, V2, V3

Step5. Store the token into Data storage

After token generation, the user has the choice of either keeping the pre-computed tokens locally or storing them in encrypted form on the cloud servers. In our case, the user stores them locally to obviate the need for encryption and lower the bandwidth overhead during dynamic data operation. Figure3 describes token pre-computation before dispersing the file to the cloud server.

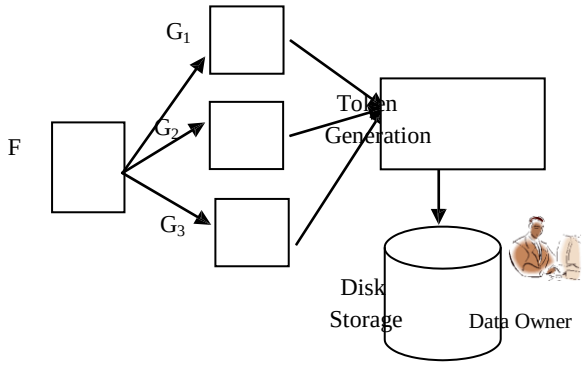

Fig 3 Token Pre-computation

C. Challenge to the Cloud Servers and Error Localization

If user wants to verify the correctness of the data stored on the cloud server, then tokens are compared. User may select the file name to check whether the data (file part) is correct or not. Upon the challenge request, cloud service provider selects the server and sends data to the client. Client generates the token and matched with tokens stored on the client side. If the tokens are matched then data stored on the server is correct. But if the tokens are mismatched, the particular server is misbehaving. In response to this user ask to correct the data file on the server.

Step 1. The user reveals the i^{th} challenge as well as the i^{th} permutation key to each server.

Step2. The server storing the vectors g_j aggregate those r rows specified by i^{th} index are collected in linear combination R_i

Step 3. Upon receiving all R_i from all servers, decryption has been performed. The tokens are generated for this data on user side.

Step 4. These tokens are matched with the stored token on the user side storage. The user verifies whether the received data remain valid determined by tokens T. If the match holds, the challenge is passed. Otherwise, it indicates that among those specified rows, there exist file block corruptions.

D . Retrieval of file and Recovery of Errors

Since our layout of file matrix is systematic, the user can reconstruct the original file by downloading the data vectors from the first *m* servers, assuming that they return the correct response values. Whenever the data corruption is detected, the comparison of pre-computed tokens and received response values can guarantee the identification of misbehaving server(s). Therefore, the user can always ask servers to send back blocks of the r rows specified in the challenge and regenerate the correct blocks by erasure correction. The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.

Step 1. The block corruption is detected in specified r rows using algorithm explained under challenge to Cloud server and Error Localization.

Step 2. Assume $s \leq k$ servers have been identified misbehaving.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 4, June 2013

Step 3. Download *r* rows of blocks from the servers.
Step 4. Treat *s* servers as erasures and recover the blocks.
Step 5. Resend the recovered blocks to corresponding servers.

II. Dynamic Data Support

From the review of the previous work, it is come to know that most of the work done on static files. This static data may be suitable for libraries and scientific datasets. But in today's scenario, dynamic data required for many applications like electronic documents, photos, log files, medical data etc. Therefore it is crucial to consider the dynamic case, where user can do the various block level operations like insert, update, delete, append to modify the data file, simultaneously it should maintain the storage correctness[3].

In cloud storage, user may want to **modify** the data blocks, and can do it without affecting the other blocks. Suppose user wants to update the data block m_i. After update the data block m_i becomes m_i'. Due to the linear property of Reed-Solomon code, a user can perform the update operation without affecting the other block. The previous tokens of m_i $block$ are erased and replaced with new tokens generated for mi'. This can be done with the homomorphic tokens.

Insert is the special operation of the update. It inserts the new data block m_j after m_i data block. The tokens are generated for the new data block and stored appropriately while maintain the storage correctness. Because the data update operation inevitably affects some or all of the remaining verification tokens, after preparation of update information, the user has to amend those unused tokens for each vector to maintain the same storage correctness assurance. In other words, for all the unused tokens, the user needs to exclude every occurrence of the old data block and replace it with the new one. But due to homomorphic construction of the verification token, the user can perform the token update efficiently.

It may possible in the application; certain data need not be required after use. It should get deleted from the file. The **delete** operation is the special case of update operation. There must be provision to support the delete operation by the update operation. In the delete operation, the original data block can be replaced with zeros or with predefined special blocks. Also all the affected blocks should be modified.

In some cases, the user may want to increase the size of his stored data by adding blocks at the end of the data file, which we refer as data **append**. We anticipate that the most frequent append operation in cloud data storage is bulk append, in which the user needs to upload a large number of blocks (not a single block) at one time. In file distribution preparation, appending blocks towards the end of a data file is equivalent to concatenate corresponding rows at the bottom of the matrix layout for file *F*.

V. EVALUATION METRICES

Some of cloud applications are distributed storing the same data in different storages to make the execution more effective in speed aspect. It gives us the point to think that how we using the concept of distributed storage in security aspect. Thus, we consider that implementing the data division after encrypting the data. Although it is assumed that data will be able to obtain by attacker in cryptography, the advantage of this method is making data more secure, because the data is encrypted to cipher text and divided into many parts, and the data can be decrypted only by collecting all of division parts. If attackers cannot take any of all division parts by hacked the storing servers, they cannot recover the encrypted data to crack. We assume that the hacked probability of every storing division server is the same, and the probability of server hacked is *H* that the value is between 0 and 1. The probability of all of *n* storing division servers simultaneously hacked is H^n . If the data is divided into $n+1$ division parts and comparing with *n* division parts, the probability of collecting all division parts is shown as follow. $H^n > H^{n+1}$

We can find that the probability of all division parts hacked is smaller when the number of division part increasing, and the attacker obtaining all of division parts for recovering the encrypted data is more difficult. In other words, more division parts make data more safety. Therefore, we ensure that using this method in cloud environment can enhance data security, and it can combine with encryption algorithm to offer a security composition for protecting confidentiality.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 4, June 2013

Tokens storage requires very less storage space [2]. It may be stored on client site or on the cloud server because extra space required for pre-computed tokens require less than 1MB for 8GB data file. Thus it is negligible. For 1 MB file, we are getting the results as shown in table 1. It can be better, depends on the bandwidth of the network. In this we are using n=m + k, where n is number of cloud servers and m is number of data vectors i.e. data file is divided into m data vectors and k is the replication servers. In this we have replicated data vectors on m cloud servers itself.

TABLE 1: Cost of upload, download and token generation for 1MB file

Upload time for file	30s
Download time	15s
Token Generation Time	10s

Upload time consists of data vector generation, server computation time and communication time. This gives better result than result for parity matrix generation and storing the data vectors in [2]. There is no need to maintain the parity matrix along with the replicated data vectors for the same file. It also reduces the storage used parity matrix. In parity matrix if more than one digit changed then correctness cannot be maintained.

Our system reduces the maintenance and communication cost because there is no need to store complete file on each of n storage servers. Existing system uses single machine to show the results through simulation [2], where as we used heterogeneous environment for Data Security Model. To prevent data dropout or corruption, the integrity of data stored in each server needs to be periodically checked. Owner of the data sends challenge to the cloud servers. If server fails to pass the integrity check, the repair task is accomplished by generating the exactly same packets as those previously stored in corrupted storage servers. Such repair method does not introduce any additional linear dependence among newly generated packets and that packet stored in healthy storage servers, and therefore maintains the data availability

Figure 4.1shows the number of blocks accessed by the data owner. If any unauthorized user modifies the data, changes will be reflected. Figure 4.2 can detect the modifications in the original file.

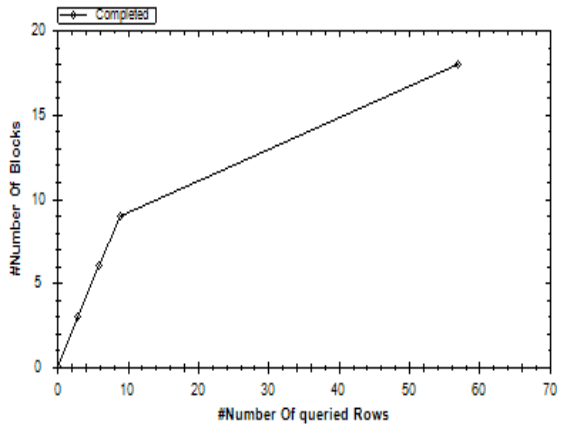

Fig 4.1: Number of queried rows by data owners

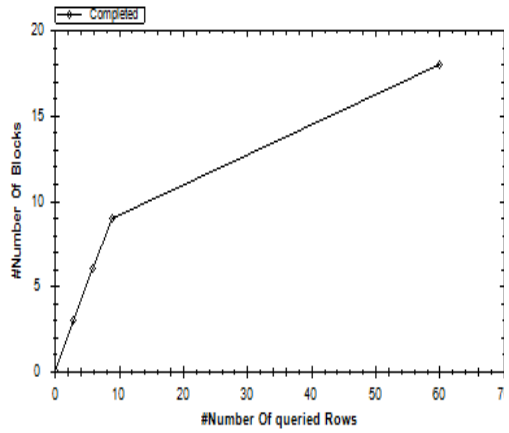

Figure4.2: Detection against data modification

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 4, June 2013

VI. CONCLUSION

In this, we investigated the problem of data security in cloud data storage, which is essentially a distributed storage system. To ensure the correctness of users ‘data in cloud data storage, we proposed an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, insert and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy data vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s). Through detailed security and performance analysis, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks.

REFERENCES

[1] I-Hsun Chuang, Syuan-Hao Li, Kuan-Chieh Huang, Yau-Hwang Kuo, ―An Effective Privacy Protection Scheme for Cloud Computing‖, ICACT-2011, Pages 260-265

[2] Cong Wang, Qian Wang, Kui Ren, Wenjing Lou ―Ensuring Data Storage Security in Cloud Computing,‖ Proc.IWQoS _09, July 2009, Pages 1–9.

[3] Qian Wang, Cong Wang, Jin Li, Kui Ren, and Wenjing Lou, ―Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing,‖ Proc. ESORICS _09, Sept. 2009, Pages 355–70.

[4] Qian Wang, Cong Wang, Jin Li, Kui Ren, and Wenjing Lou, ―Privacy-Preserving Public Auditing for Storage Security in Cloud Computing,‖ Proc. IEEE INFOCOM 10, Mar. 2010.

[5] Kevin D. Bowers, Ari Juels, Alina Oprea, ― HAIL: A High-Availability and Integrity Layer for Cloud Storage‖, Proceeding CCS '09 Proceedings of the 16th ACM conference on Computer and communications security Pages 187-198

[6] Mehul A. Shah, Mary Baker, Jeffrey C. Mogul, Ram Swaminathan., ―Auditing to keep Online Storage Services Honest,‖ Proc.USENIX HotOS _07, May 2007.

[7] Giuseppe Ateniese, Roberto Di Pietro, Luigi V. Mancini, and Gene Tsudik., ―Scalable and Efficient Provable Data Possession,‖ Proc.SecureComm _08, Sept.2008.

[8] Ayad F.Barsoum and M.Anwar Hasan, ―On Data Replication and Storage Security over Cloud Computing: Are we getting what we are paying for?‖ in digital library citeseerx.ist.psu.edu/viewdoc/ pg. 1-36

[9] Dai Yuefa, Wu Bo, Gu Yaqiang, Zhang Quan, Tang Chaojing‖ Data Security Model for Cloud Computing‖ Proceedings of the 2009 International Workshop on Information Security and Application (IWISA 2009) pg. 141-144