Secure Approach for Data in Cloud Computing

Bharti Dhote
Sinhgad Institute of Technology,
Lonavala, Pune

A.M. Kanthe
Sinhgad Institute of Technology,
Lonavala, Pune

ABSTRACT
Previously, computer software was not written with security in mind; but because of the increasing frequency and sophistication of malicious attacks against information systems, modern software design methodologies include security as a primary objective. With cloud computing systems seeking to meet multiple objectives, such as cost, performance, reliability, maintainability, and security, trade-offs have to be made. Any cloud server is vulnerable to an attacker with unlimited time and physical access to the server. Additionally, physical problems could cause the server to have down time. This would be a loss of availability, which is one of the key principles of the security triad — confidentiality, integrity, and availability (CIA). Availability addresses the issues that include attempts by malicious entities to control, destroy, or damage computing resources and deny legitimate access to systems. While availability is being preserved, confidentiality and integrity have to be maintained. In this paper, we propose an effective and flexible scheme opposing to its predecessor. By utilizing the homomorphic token and cryptographic encryption method achieves the integration of storage correctness insurance and error localization i.e. the misbehaving of servers. The new scheme further supports to dynamic operations on data blocks like delete, update, insert, append etc.

General Terms
Security, Encryption.

Keywords
Availability, Cloud Computing, Data Security.

1. INTRODUCTION
Cloud computing is an emerging business model. Cloud computing provides the services over the internet. Internet availability varies from 95% to 99.6%. On user demand, it dynamically delivers everything as a service over the internet such as network, operating system, storage, hardware, software, and resources.. Web applications are used to provide the cloud services and online data are highly irregular. Cloud service provides storage for e.g. Amazon S3, ElephantDrive, Gigaspaces, etc. Therefore storage capacity may be large and it should be further scale as online data continuously increasing. However, as data scales up, hardware failures in current datacenters become more frequent. To this end, the support of service level agreements (SLAs) [4] with data availability guarantees in cloud storage is very important. Moreover, in reality, different applications may have different availability requirements. Cloud computing is an emerging computing style which provides dynamic services, scalable and pay-per-use. These services are classified into three types: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [1].

- Software as a Service (SaaS): The consumer uses an application, but does not control the operating system, hardware or network infrastructure on which it's running. The cloud infrastructure provides the services which can be accessible by the thin client interface such as a web browser. Some examples are Google Apps (mail, docs, and etc.) and Salesforce.
- Platform as a Service (PaaS): Service provider provides a specific cloud environment, some software tools and programming language to consumer for developing, testing, and hosting their applications. In this service model, the consumer does not control or manage the underlying cloud infrastructure. An example of PaaS is Google App engine.
- Infrastructure as a Service (IaaS): IaaS allows consumer to rent hardware include processors, storages, network, and other fundamental computing resources. In this service model, consumers do not control or manage the underlying cloud infrastructure directly. They control the computing resources through operating systems. Cloud computing is deployed as three models such as Public, Private, and Hybrid clouds [1].

Data security for such a cloud service encompasses several aspects including secure channels, access controls, and encryption [10]. And, when we consider the security of data in a cloud, we must consider the security triad: confidentiality, integrity, and availability [5]. In the cloud storage model, data is stored on multiple virtualized servers. Physically the resources will span multiple servers and can even span storage sites.

Motivations
Large amount of data is generated due to applications. This data need to be stored which requires large amount of storage space. Data generation is currently outpacing storage availability, hence, there will be more and more need to outsource data. Cloud computing provides the storage and supports for outsourcing of data without having the local copy of data or files. However an important problem is to prevent unauthorized modifications. But, for this it requires either data replication or on-demand computation of a function (e.g., a hash) over the entire outsourced data [4].

The rest of the paper is organized as follows: Section II introduces the related work. The system architecture and proposed data security model are explained in Section III. In Section IV, describes dynamic data operations. The Practical environment and results are included in V, and conclusion is described in Section VI.

2. RELATED WORK
Quality of Service is an important aspect of the data security. Cloud computing has new challenges [8] and threats due to number of reasons. Traditional cryptographic primitives cannot provide the complete protection to the data stored on cloud servers. The user loss the control over the data stored on cloud server as user does not have the physical copy. Therefore verification of correctness [11][12] of the data must be checked without having the knowledge of the whole data. Hence the problem of verifying the correctness of the data stored on the cloud server is more challenging. The user performs the operations like insert, delete, append, etc. verifying the correctness of the data during dynamic operations is also of great importance.

In recent years, encryption mechanisms are used to provide the privacy of the data to resolve the above problem. However, they do not take into account the system performance impact for their protection mechanism, when their methods are implemented that may cause significant performance overhead. Nevertheless, it makes users' data dangerous in sometimes that disclose its content. No one proposes a mechanism that protects data and considers the performance overhead at the same time. Thus, we want to propose a scheme that secure users' data confidentiality in the cloud storage and maintain cloud system performance without much extra system overhead.

Table 1. Exiting Systems

Sr. No	Title of Paper	Existing Model	Drawbacks
1.	An Effective Privacy Protection Scheme for Cloud Computing[1]	Mainly uses of the encryption algorithm	Not described about the dynamic data operations & server misbehavior
2.	Ensuring Data Storage Security in Cloud Computing[2]	It works on the file distribution and token pre computation	Does not provide the insertion operation as inserting of block corresponds to lot of shifting of block.
3.	Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing[3]	Allows third party auditor (TPA), on behalf of the cloud client, Merkle Hash Tree (MHT) is used for the construction of block tag authentication.	Third party can be unreliable or not be able to commit necessary computation resources performing continuous verifications.
4.	Privacy-Preserving Public Auditing for Secure Cloud Storage[4]	Enables the TPA to perform audits for multiple users. Uses the HLA based solution.	Independent to data Encryption is the problem we are going to tackle in this paper.
5.	HAIL: A High-Availability and Integrity Layer for Cloud Storage[5]	Manages file integrity and availability across a collection of servers or independent storage services. It makes use of PORs as building blocks.	Not consider the secrecy.
6.	Auditing to Keep Online Storage Services Honest[6]	Describe approaches and system hooks that support both internal and external auditing of online storage services.	Cryptographic scheme is not sufficient to keep privacy.
7.	Scalable and Efficient Provable Data Possession[7]	Based entirely on symmetric key cryptography.	Not supported to insertion operation.
8.	On Data Replication and Storage Security over Cloud Computing: Are we getting what we are paying for?[8]	Stores multiple copies of customers' data	Storage overhead at server side increases.
9.	Data Security Model for Cloud Computing[9]	Mathematical model of data security is given.	Single point of failure if file corrupts.
10.	Implementing Digital Signature with RSA Encryption Algorithm to Enhance the Data Security of Cloud in Cloud Computing [10]	Use of RSA encryption algorithm	In case of server misbehavior file cannot be retrieved.

3. PROPOSED MODEL OF DATA SECURITY

3.1 System Architecture

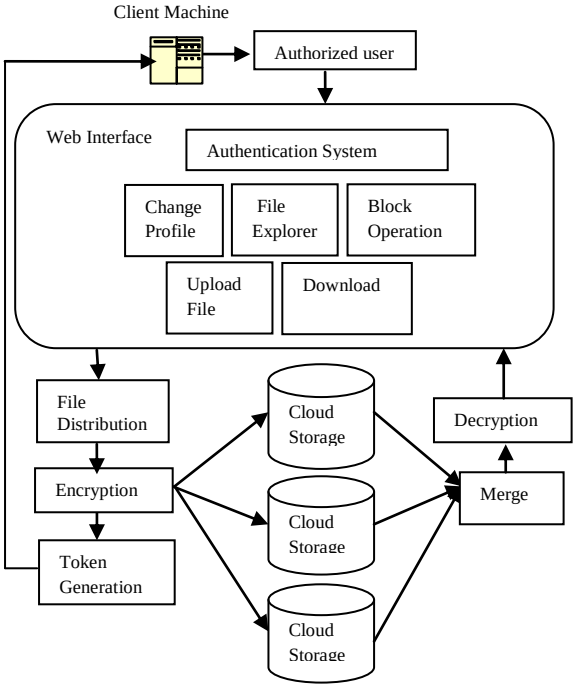

Figure 1. System Architecture for Cloud Data Storage

Cloud data storage is a distributed data storage works as a single system in concurrent, coordinated and distributed manner. User can store his data on cloud servers through web application. Cloud service provider coordinates set of cloud servers. 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.

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

3.2 Design Goals

Proposed scheme is to build efficient data security model which supports for i) Data Integrity – to ensure the user's data is correct and stored in cloud server. ii) Effectively locate the server on which data has been modified by unauthorized user. iii) Support for the dynamic data like append, delete, insert, update while retaining the same storage correctness.

3.3 Data Division

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.

Algorithm 1: File Distribution Preparation

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.

Step 2. 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.

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

If *file_length* mod *n* < > *0* then

datavector_length = datavector_length+1 (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 Vander monde matrix for the Matrix_Multiply () function. Construct *A* using Vander monde matrix defined over *GF* (2^w). For e.g. if we take 3x3 Vander monde matrix defined over *GF* (2^4)

$$A = \begin{bmatrix} 1^0 & 2^0 & 3^0 \\ 1^1 & 2^1 & 3^1 \\ 1^2 & 2^2 & 3^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 5 \end{bmatrix}$$

Step 3. Perform *Matrix_multiply (vector, A)* operation on each data vector. Use g_1, g_2, g3…g_m as data vector names.

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

Step 4. 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.

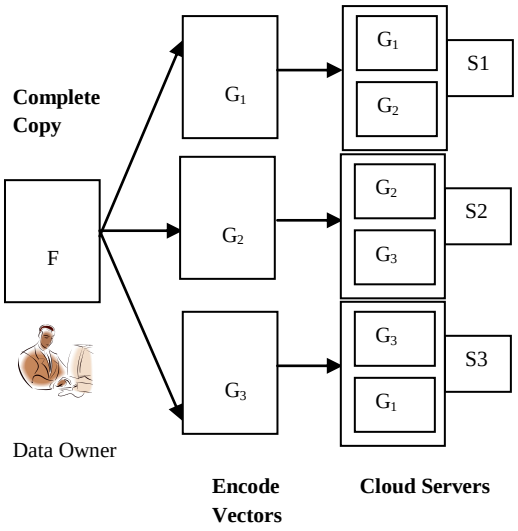

Figure 2.Distribution of File Stored on Cloud Storage

3.4 Token Pre-computation

Before storing the data vectors on cloud servers, user pre-computes the verification tokens. These tokens are used to check the integrity of data stored on cloud servers. Also these tokens are used to locate the cloud server on which data has been modified by the hacker. Before data division and dispersing file user generates tokens on individual data vectors. When user wants to check the correctness of the data, he sends the file identifier 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 is misbehaving. The details of token generation are shown in Algorithm 2.

Algorithm 2. Token Generation

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 e.g. (dividing byte array in 3parts)

$$P(x) = 1x^8 + 1x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 1x^2 + 1x^1 + 1x^0$$

For j←1 to n Do
  For i ← 1 to t Do
    A[i] = *f* (i)
    sum = 0
    For q ← 1to r Do
      a←g_j [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
  V_j ←Add V1, V2, V3
Step 5: Store the token into Data storage

Generated tokens can be stored locally on user's machine. User can keep these tokens on cloud server also in encrypted form. But storing the tokens on cloud server increases communication cost. Tokens require less storage i.e. for 8GB data, it requires 1MB storage. In our case, the user stores them locally to prevent the need for encryption and lower the bandwidth overhead during dynamic data operation [2]. Figure 3, describes token pre-computation before dispersing the file to the cloud servers.

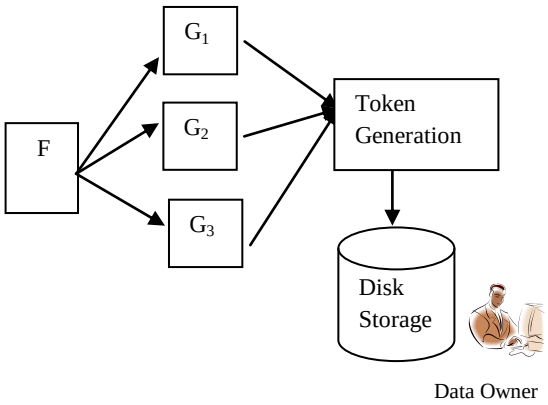

Figure3. Token Pre-computation

3.5 Verifying Storage Correctness and Locating Misbehavior of Server

Previous Scheme does not explicitly consider the problem of locating server on which error occurred. 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.

Algorithm 3: Challenges and Error Localization

Step 1. The user reveals the i^{th} challenge as well as the i^{th} permutation key to each server.
Step 2. 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:

$$(R_{i1}, \cdots, R_{im}).T \stackrel{?}{\Leftrightarrow} V_i \quad (3)$$

If the above equation holds, the challenge is passed. Otherwise, it indicates that among those specified rows, there exist file block corruptions.

Once the inconsistency among the storage has been successfully detected, we can rely on the pre-computed verification tokens to further determine where the potential

data error(s) lies in. Note that each response R_i is computed exactly in the same way as token V_i, thus the user can simply find which server is misbehaving by verifying the following n equations

$$R_i^{(j)} \stackrel{?}{\Leftrightarrow} V_i^{(j)}, j \in \{1, \dots, n\} \quad (4)$$

3.6 Repossession of file and Correcting Data Blocks

User can regain the original file from cloud servers. Cloud service provider downloads the data vectors from first m servers and reconstructs the complete file. Assuming that cloud service provider retrieve the correct data blocks from m servers. 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, shown in Algorithm 4 as long as there are at most *k* misbehaving servers are identified. The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.

Algorithm 4: Retrieval and Recovery

Step 1. The block corruption is detected in specified r rows using Algorithm 3.

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

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.

4. 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 [3] 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.

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. Most of the times, append operation includes large number of blocks in cloud storage. With this facility user can upload large number of blocks to its data file. In file distribution preparation, appending blocks towards the end of a data file is simply the concatenation of data vectors in the file *F*.

5. PERFORMANCE EVALUATION

Performance evaluation measures we have considered are

i) Time required generating the tokens.
ii) Communication time to upload and download time

We implemented our model on four systems with Intel core 2 duo running at 2.13 GHz, 2GB of RAM. We consider one system for client and other three as cloud service provider, and cloud servers. Four systems are connected to each other with LAN such that network latency is negligible.

To ensure the data reliability in distributed storage systems, various data redundancy techniques can be employed, such as replication, erasure codes, and network coding. Replication is the most straightforward way of adding data redundancy where each of *n* storage servers stores a complete copy of the original data. Data users can retrieve the original data by accessing any one of the storage servers, and the corrupted server can be repaired by simply copying the entire data from a healthy server.

In our system, given the same level of redundancy, the erasure codes based distributed storage system is more reliable Data users can recover the entire *m* original packets by retrieving the same number of encoded packets from any *k*-combination of *n* servers, and therefore every server only needs to store *m/k* encoded packets which is regarded as the property of optimal redundancy-reliability tradeoff.

Our system reduces the maintenance and communication cost because there is no need to store complete file on each of n storage servers. 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 without disturbing others. 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. In figure 4[2], the data availability is the probability that data users could recover original data from any *k*-combination of *n* storage servers.

Figure 5 shows the number of blocks accessed by the data owner. If any unauthorized user modifies the data, changes will be reflected even for small modification. This is the graph for same file. Figure 6 can detect the modifications in the original file. For 1 MB file the results are illustrated in table 2. It can be better, depends on the bandwidth of the network. This result is for m=3, where m is the primary cloud servers and k=3, where k is replication servers.

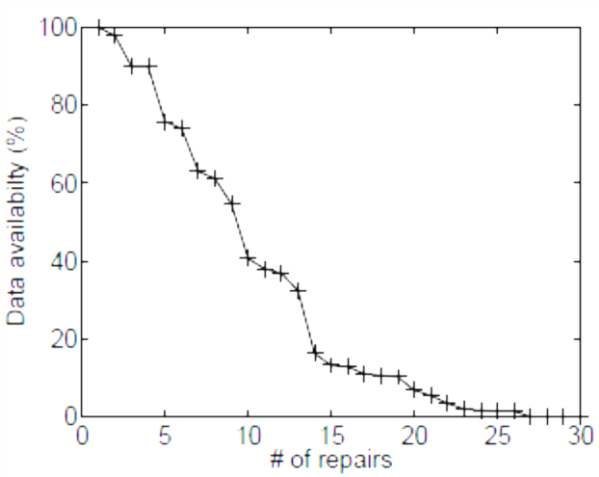

Figure 4. Data availability after functional repairs

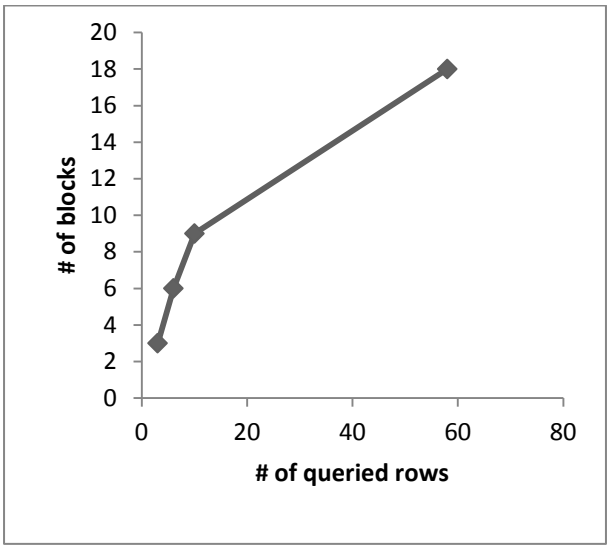

Figure 5. Number of queried rows by data owners

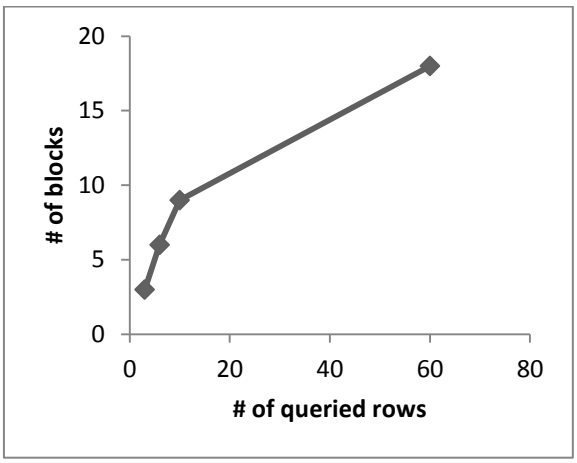

Figure 6. Detection against data modification

Table 2. Upload, download and token generation time.

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

6. CONCLUSION

We describe the method for data security, which is the major parameter of the quality of service. Our contributions are 1) data division – if data is divided into smaller and smaller parts, it is difficult for hacker to get the complete data. 2) server misbehavior – locating the server on which modifications has been done by unauthorized user. 3) Checking integrity of data with the help of token pre-computation. We have provided the block operations for dynamic data operations. Most of the previous work does not support for dynamic insertion. But our scheme supports for the insertion operation. This also ensures the data availability in case of communication link failure or particular server misbehaves.

7. 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

[10] Uma Somani,Kanika Lakhani,Manish Mundra, "Implementing Digital Signature with RSA Encryption Algorithm to Enhance the Data Security of Cloud in Cloud Computing" , 2010 1st International Conference on Parallel, Distributed and Grid Computing (PDGC - 2010) Pages 211-216

[11] A. Juels and J. Burton S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. of CCS '07, pp. 584–597, 2007.

[12] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Cryptology ePrint Archive, Report 2008/175, 2008, http://eprint.iacr.org/