International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 3– No.1, July2012 – www.ijais.org

Verifying Data Integrity in Cloud

S. P. Jaikar
Department of Information Technology,
Sinhgad College of Engineering, University of Pune

M. V. Nimbalkar
Department of Information Technology,
Sinhgad College of Engineering, University of Pune

ABSTRACT
Clouds are revolution in computing field which provides on demand access to virtualized resources which are hosted outside of your own data center. In Cloud computing, data and applications are moved to large data centers where management of data is not fully trustworthy. Secure outsourcing of data and applications to a third party service provider is very important. Moving of data in cloud is convenient for users as they don't have to deal with complicated data management and other hardware related issues on their local data centers. But this convenience brings down the user to the mercy at their service providers for correctness and trustworthiness of their data. Since the applications and data are under control of the third party service provider, the users have to rely on the security mechanisms implemented by service provider for availability and integrity of their data. We propose a distributed scheme to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud. We are using erasure correcting code in the file distribution preparation to provide redundancies. We are relaying on challenge response protocol along with pre-computed tokens to verify the storage correctness of user's data & to effectively locate the malfunctioning server when data corruption has been detected. Our scheme maintains the same level of storage correctness assurance even if users modify, delete or append their data files in the cloud.

Keywords
Cloud computing, Distributed data storage, Virtualization, Pervasive Computing, Data security.

1. INTRODUCTION

1.1 Clouds: New Era of computing
Innovations are necessary to ride the inevitable tide of change. Most of enterprises are striving to reduce their computing cost through the means of virtualization. This demand of reducing the computing cost has led to the innovation of Cloud Computing. Cloud computing is a term used to describe a set of IT services that are provided to a customer over a network on a leased basis and with the ability to scale up or down as per their service requirements. Usually cloud computing services are delivered by a third party provider who owns the infrastructure. Cloud Computing has become one of the most talked about technologies in recent times and has got lots of attention from media as well as analysts because of the opportunities it is offering.

A general scenario of cloud computing is shown in Fig.1. Cloud provider provides the resources to the clients in different forms like IaaS, PaaS, and SaaS. Clients subscribes to the service via internet, & they pays as per usage of service to which they have subscribed.

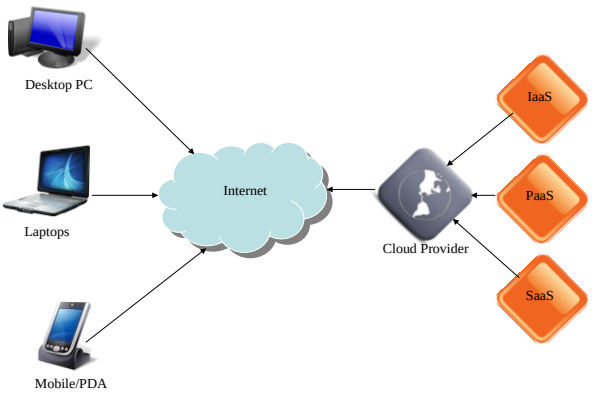

Fig.1. How Cloud Computing works.

Clients need not to be stationary it can be PDA's, Mobiles and Laptops. Clients doesn't need to have in house infrastructure, they can purchase to the service they want on hourly, weekly or monthly reasonable basis. Cloud computing users can avoid capital expenditure on hardware, software, and services when they pay a provider only for what they use. Consumption is usually billed on a utility or subscription basis with little or no upfront cost.

1.2 Cloud Data Storage
Cloud computing, the trend toward loosely coupled networking of computing resources, is unmooring data from local storage platforms. Users today regularly access files without knowing or needing to know on what machines or in what geographical locations their files reside. They may even store files on platforms with unknown owners and operators, particularly in peer-to-peer computing environments. One fundamental aspect of this new computing model is that data is being centralized or outsourced into the cloud. From the data owners perspective, including both individuals and IT enterprises, storing data remotely in a cloud in a flexible on-demand manner brings appealing benefits: relief of the burden of storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, personnel maintenance, and so on although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they still face a broad range of both internal and external threats to data integrity. Outages and security breaches of noteworthy cloud services appear from time to time. Amazon S3's recent downtime [11], Gmail's mass email deletion incident [12] is such examples. For benefits of their own, there are various motivations for CSPs to behave unfaithfully toward cloud customers regarding the status of their outsourced data. Examples include CSPs, for monetary

reasons, reclaiming storage by discarding data that has not been or is rarely accessed or even hiding data loss incidents to maintain a reputation [12].

In short, although outsourcing data into the cloud is economically attractive for the cost and complexity of long-term large scale data storage, it does not offer any guarantee on data integrity and availability. In particular, simply downloading the data for its integrity verification is not a practical solution due to the high cost of input/output and transmission across the network. Besides, it is often insufficient to detect data corruption only when accessing the data, as it does not give correctness assurance for remaining data and might be too late to recover the data loss or damage.

Juels and Kaliski [1] proposed a Proof of Retrievability protocol and provided formal security definitions. POR is a protocol in which a server/archive proves to a client that a target file *F* is intact, in the sense that the client can retrieve all of *F* from the server with high probability. Juels and Kaliski present proofs of Retrievability focusing on static archival storage of large files. Furthermore, the number of queries a client can perform is limited, and fixed a priori. G.Ateniese et al. [2] introduced a model for provable data possession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. Major limitation of this model is the number of challenges a client can perform against the server is limited. Data privacy issues also have not been addressed. The model doesn't provide any support for dynamic operations on data blocks.

G. Ateniese & R. Pietro [3] constructed a highly efficient and provably secure PDP technique based entirely on symmetric key cryptography, while not requiring any bulk encryption. The drawback with this technique is that the number of updates and challenges a client can perform is limited and fixed a priori. Also, one cannot perform block insertions anywhere only append-type insertions are possible. H. Shacham and B. Waters [4] proposed protocols based on the idea of using homomorphic authenticators for file blocks, essentially block integrity values that can be efficiently aggregated to reduce bandwidth in a POR protocol. Due to the use of integrity values for file blocks, this scheme can use a more efficient erasure code to encode the file; the block authenticators transform the erasure code into an error-correcting code. This scheme supports an unlimited number of verifications, but the solution is static.

K. D. Bowers, A. Juels, and A. Oprea [5], introduced High-Availability and Integrity Layer. It is a distributed cryptographic system that permits a set of servers to prove to a client that a stored file is intact and retrievable. C.Wang et al. [6], has proposed an effective and flexible distributed scheme with explicit dynamic data support to ensure the correctness of user's data in the cloud. It relies on erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability. This scheme has limitation on number of challenges user's can perform against the server. User has burden of storing pre-computed tokens locally. Q. Wang et al. [7], proposed a scheme with explicit dynamic data support to ensure the correctness of user's data in the cloud. User can easily verify integrity of his data without much overhead with the help of challenge response protocol. This scheme doesn't address privacy concerns of users. C. Erway et al. [8], come up with A Dynamic Provable Data Possession technique which demonstrates to a client that a server possesses a file *F* in an informal sense, but is weaker than a POR in that it does not guarantee that the client can retrieve the file. Major limitation with this scheme is that, it doesn't support all dynamic operations. Curtmola et al. [9] aim to ensure data possession of multiple replicas across the distributed storage system. They extend the PDP scheme in [2] to cover multiple replicas without encoding each replica separately, providing guarantee that multiple copies of data are actually maintained.

We propose a distributed scheme to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud. We are using erasure correcting code in the file distribution preparation to provide redundancies. We are relaying on challenge response protocol along with pre-computed tokens to verify the storage correctness of user's data & to effectively locate the malfunctioning server when data corruption has been detected. Our scheme maintains the same level of storage correctness assurance even if users modify, delete or append their data files in the cloud. It has no limitation on number of challenges user can perform against the server. Users are having no burden of storing pre-computed tokens locally; all the tokens are stored inside the cloud.

By splitting the file according to the number of server's we are adding extra security to system, so that if an unauthorized user compromises a storage server, he won't get access to all the data. To add furthermore security, we are encrypting the user's data before uploading it to cloud, as we do in traditional system. Another important aspect of our system is that our system is proactive & guarantees to detect every single data modification attack. We are not addressing any load balancing techniques in this research; also we are not doing any work on privacy issues.

2. CLOUD STORAGE ARCHITECTURE

The general architecture of cloud storage system is illustrated in Fig.2. Generally two different network entities can be identified. We have assumed that user's have direct peer to peer connection between them & cloud.

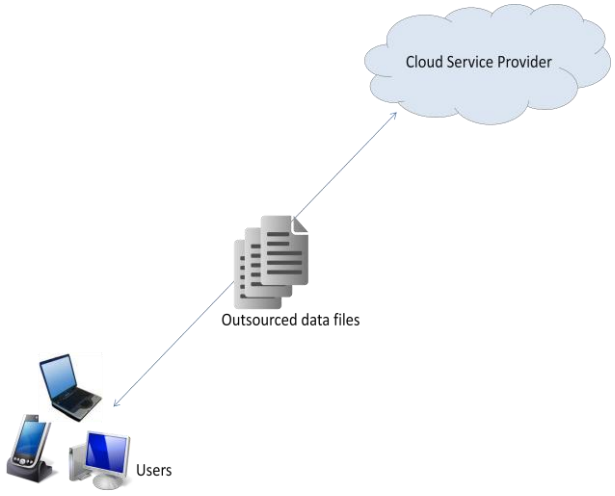

Fig.2 Storage Architecture for Cloud

Different network entities are mentioned below:

- User: users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations.
- Cloud Service Provider (CSP): CSP is who has the capabilities to host data & applications of users. They have huge resources that they can provide dynamically for satisfying various user needs. CSP having expertise in building & managing cloud servers, having their own data centers for hosting user's data.

Fig.2 shows how the data is outsourced in cloud and users have no control over it. This also gives perception of the problem with the storage and to ensure the integrity of the data in the cloud. In cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, cooperated and distributed manner. Data redundancy can be employed with technique of erasure-correcting code to further tolerate faults or server crash as user's data grows in size and importance. Thereafter, for application purposes, the user interacts with the cloud servers via CSP to access or retrieve his data. In some cases, the user may need to perform operations on his data.The most general forms of these operations we are considering are update, delete, insert and append. As users no longer possess their data locally, it is of critical importance to assure users, that their data are being correctly stored and maintained. That is, users should be equipped with security means so that they can make continuous correctness assurance of their stored data even without the existence of local copies.

In case those users do not necessarily have the time, feasibility or resources to monitor their data, user's can delegate the tasks to an optional trusted TPA of their respective choices. But users need to pay to the Third Party Auditors for that. This is not our aim, what we want is to give freedom to users to ensure intactness of their data in cloud. In our scheme, we assume that the point-to-point communication channels between each cloud server and the user is authenticated and reliable. Security threats faced by cloud data storage can come from two different sources. On the one hand, a CSP can be self-interested, untrusted and possibly malicious. It may also attempt to hide a data loss incident due to management errors, Byzantine failures and so on. On the other hand, there may also exist an economically motivated adversary, who has the capability to compromise a number of cloud data storage servers in different time intervals and subsequently is able to modify or delete user's data while remaining undetected by CSPs for a certain period. So we have attackers with different purposes in different context & we need to classify them as per the severity of damage they can do to storage.

To ensure the security for cloud data storage under the aforementioned adversary model, we aim to design efficient mechanisms for dynamic data verification and operation.

3. SECURING DATA STORAGE

3.1 Notation & Preliminaries

- $F-$ 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 $GF(2^w)$ for $w = 4, 8, 16$.
- $R-$ The dispersal matrix used for Reed-Solomon coding.
- $D-$ Data matrix constructed over data vectors.
- $C-$ The encoded file matrix, which includes a set of $n = m + k$ vectors, each consisting of l blocks.
- $PRF-$ Pseudorandom function.
- $PRP-$ Pseudorandom permutation.

It is well known that erasure-correcting code may be used to tolerate multiple failures in distributed storage systems [18]. In cloud data storage, we rely on this technique to disperse the data file F redundantly across a set of $n = m + k$ distributed servers. $R(m + k, k)$ Reed-Solomon erasure-correcting code is used to create k redundancy parity vectors from m data vectors in such a way that the original m data vectors can be reconstructed from any m out of the $m + k$ data and parity vectors. By placing each of the $m + k$ vectors on a different server, the original data file can survive the failure of any k of the $m + k$ servers without any data loss, with a space overhead of k/m. For support of efficient sequential I/O to the original file, our file layout is systematic, i.e., the unmodified m data file vectors are distributed across $m + k$ different servers. We are using Reed Solomon Algorithm to disperse the file redundantly over m storage devices. In Reed Solomon Algorithm, Given n data devices and m checksum devices, the RS-Raid algorithm for making them fault-tolerant to up to n failures is as follows.

1. Choose a value of w such that $2^w > n + m$. It is easiest to choose $w = 4\ or\ w = 8\ or\ w = 16$, as words then fall directly on byte boundaries. $(w - word\ size)$.
2. Set up the table's $gflog$ and $gfilog$. These tables are used to perform multiplication over Galois Fields.
3. Set up the matrix P to be the $m \times n$ matrix: $p_{i,j} = j^{i-1}$ where multiplication is performed over $GF(2^w)$.
4. Use the matrix P to calculate and maintain each word of the checksum devices from the words of the data devices. Again, all addition and multiplication is performed over $GF(2^w)$. Create the matrix D as actual data matrix & Calculate C by equation $PD = C$.
5. If any number of devices up to m fails, then they can be restored in the following manner. Choose any n of the remaining devices, and construct A' and a vector E'. Then solve for D in $A'D = E'$. This enables the data devices to be restored. Once the data devices are restored, the failed checksum devices may be recalculated using the matrix F.

So, as per RS Raid algorithm, we divide the input file to the n data vectors, where n is number of storage devices present in the system. The data vectors that are generated are of equal size, so the load will be distributed equally to all the storage devices. We create $m \times n$ matrix D & store all the data vectors in matrix D. In next step we create a Reed Solomon matrix R which is generated over Galois field, i. e. $GF(2^w)$. In our case we have assumed word

size $w = 4$. After this stage, we perform matrix multiplication to generate checksum matrix C. We multiply data matrix D with Reed Solomon matrix R. The resultant matrix is the redundant matrix which contains original data from data matrix D & parity vectors added by Reed Solomon matrix. It means matrix D will be stored redundantly across the different storage devices & it will be used for token computation as well as data recovery purpose.

3.2 Token Pre-computation

To verify the correctness of user's data & to locate the errors, we entirely rely on the pre-computed verification tokens. These tokens are calculated before file distribution & they are very short. We are computing the tokens by pseudorandom function PRF & pseudorandom permutation function PRP. We pre-computes short verification tokens on individual vector, each token covering a random subset of data blocks. We have assumed block size as 256 bits & r as 8 number of verification per indices. We have three data devices and three checksum devices. Then $n = 3$ and $m = 3$. We choose $w = 4$, since $2^w > n + m$. Next, we set $gflog$ and $gfilog$ table's. $gflog$ and $gfilog$ tables are shown in Table 1.

We construct P to be a 3×3 matrix, defined over $GF(2^4)$.

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

Now, we can calculate each word of each checksum device using $PD = C$

Later, when the user wants to make sure the storage correctness for the data in the cloud, he challenges the cloud servers. Upon receiving challenge, cloud server computes the new value of tokens, which is compared with previously calculated tokens. It gives clear idea about integrity of user's data.

Algorithm: TOKEN PRE-COMPUTATION

1. Begin
2. Choose file F to upload & encrypt the file using AES.
3. Generate $n \times m$ Vector Matrix D on file F.
4. Create Reed Solomon Matrix P over Galois Field $GF(2^w)$. where $w = 4$.
5. Generate Matrix $C = D \times P$. It is Checksum Matrix created for fault tolerance.
6. Compute Token over Matrix C i.e. $ComputeToken\ (C, l, t, r)$, where $l -$ block size, $t -$ no. of tokens, $r -$ indices per verification. Compute the tokens by pseudorandom function PRF & pseudorandom permutation function PRP.
7. Store these precomputed tokens on the main cloud server.
8. Disperse the file over the Cloud. i.e. $DisperseFile(Matrix\ D)$
9. End.

3.3 Correctness Verification

To eliminate the errors in storage systems key prerequisite is to locate the errors. However, many previous schemes do not explicitly consider the problem of data error localization, thus only provide binary results for the storage verification. In our scheme we integrate the correctness verification and error localization in our challenge-response protocol. The newly computed tokens from servers for each challenge are compared with pre-computed tokens to determine the correctness of the distributed storage. This also gives information to locate potential data errors.

Algorithm: CORRECTNESS VERIFICATION

1. Begin Challenge i, for $i = (i = 1\ to\ n)$, where $n -$ total number of cloud servers.
2. Get $TokenArr()$ // Getting precomputed tokens from main cloud server.
3. $HandleChallenge()$ // Reading file blocks from all cloud servers for calculating new tokens.
4. Generate Vector Matrix D on all file blocks that are read in step 3.
5. Create Reed Solomon Matrix P
6. Generate Matrix $C = D \times P$. On this matrix, new tokens will be computed.
7. Compute token on Matrix C. $ComputeToken\ (C, l, t, r)$
8. If $((Precomputed\ token == newly\ computed\ token)$ then,

   Data is intact Else

   Data is Corrupt. For that i, initiate the recovery.
9. End

3.4 Error Recovery

Once the data corruption is detected, next important step is to recover the corrupted data and bring data storage back to consistent state. The comparison of pre-computed tokens and received response values can guarantee the identification of misbehaving server. Therefore user can recover the corrupted data. Our system recovers data from backup server & distributes all data vectors to corresponding servers. This will results in successful recovery of corrupted data. But due to file splitting we made at the time of file distribution, user's need to recover file from all the servers. Error localization is limited to misbehaving servers only, i.e. servers giving false assurance of posing user's data.

Algorithm: Error Recovery

1. Begin (Assume that the data corruptions have been detected & $s \leq k$ servers have been identified misbehaving.)
2. Download consistent data blocks from backup server.
3. Create the data vectors as per number of cloud storage servers.
4. Distribute the consistent data blocks to corresponding servers & recover the data.
5. End.

Table 1: gflog and gfilog tables for $GF(2^4)$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$gflog[i]$	−	0	1	4	2	8	5	10	3	14	9	7	6	13	11	12
$gfilog[i]$	1	2	4	8	3	6	12	11	5	10	7	14	15	13	9	−

3.5 Dynamic Operations

In cloud data storage, there are many potential scenarios where data stored in the cloud is dynamic, like electronic documents, photos, or log files etc. Therefore, it is crucial to consider the dynamic case, where a user may wish to perform various operations of update, delete and append to modify the data file while maintaining the storage correctness assurance. The straightforward and trivial way to support these operations is for user to download all the data from the cloud servers and re-compute the whole parity blocks as well as verification tokens. This would clearly be highly inefficient. In cloud data storage, sometimes the user may need to modify some data stored in the cloud, from its current value to a new one. We refer this operation as data update. To perform update operation on particular data block client need to recalculate the verification token on updated data. Also client need to update this value of newly calculated token to all the replicas of file in storage cloud. When user want to perform update operation, the splitted file from all storage servers is merged and given to the user to perform data updates. Once user has finished with the updating the data, new tokens are calculated on whole file and they are stored on main cloud server. After this, updated file is splitted back and dispersed onto corresponding cloud storage servers. Update operations include modifying file, inserting data, as well as deleting data from file.

Sometimes, after being stored in the cloud, certain data may need to be deleted. The delete operation we are considering is a general one. When user wants to delete some file, he can simply delete it. In delete operation, file blocks that are distributed among cloud storage servers are all deleted. Once file is deleted, we cannot perform any recovery of deleted files as there won't be any backup available in main cloud server. In some cases, the user may want to increase the size of his stored data in file by adding data at the end of the data file, which we refer as data append operation. So in case of append operation whenever user append data to his file, new verification tokens are calculated & stored on main cloud server & file is splitted as before and dispersed among the cloud storage servers.

4. IMPLEMENTATION & RESULTS

We have implemented our system with the help of web services. The functionalities of cloud servers are provided through web services. We have used .net framework 4.0. At the back end side we used MySQL server.

Our model is systematic & it guarantees detection of every single data modification attacks. So there is no probability of detection as previous work rather there is guarantee of detection of each modification attack. We have evaluated the performance of our system in following cases:

1. Token pre-computation time.
2. File distribution time.
3. Server Token computation time.
4. Server response time.

Our experiment is conducted using C#.Net on a system with an Intel core 2 duo processor running at 2.10 GHz, 4 GB of RAM, and a 7200 RPM Western Digital 320 GB Serial ATA drive. We have tested our system under upload speed of 1Mbps & varied file size up to 10MB. The token pre-computation is long process. It includes encrypting user's file then converting user's file into data vector matrix D. Next step is to create Reed Solomon matrix P. Later we generate checksum matrix C on which we computes our short tokens. After that the file is distributed to all cloud storage servers. Cost of token pre-computation & file distribution is shown in Table 2. By looking at the Fig.5 & Fig.6 the time taken for token pre-computation & file distribution is very small & it increases gradually as file size grows.

As we have mentioned, dynamic operations like append, update, delete can be performed on .doc, .rtf, and .txt file formats only. When user performs any dynamic operation, the new token values are calculated and stored back into cloud. Modified file is divided again & dispersed to all cloud servers to maintain consistency. To demonstrate data modification attacks, we have provided a hacker account with access to all the uploaded files by all users. Hacker performs the dynamic operations on files like an authorized user. Our system detects every single data modification done by an unauthorized user. Data modification is shown in Fig.3.Token pre-computation technique helps us to find out any modification to users data. It guarantees that no single unauthorized data modification is left undetected. Fig.4 shows a screenshot where data modification has been detected. After detection of an unauthorized data modification, our system initiates recovery. Sometimes storage servers try to hide an unauthorized data modification & give false result about the intactness of user's data, i.e. Storage server doesn't possess the user's data but it gives false assurance about the data. Our system guarantees detection of such misbehaving servers. Our scheme supports dynamic operations on .txt, .rtf &.doc data formats only. Support for other data formats like, .xls, .pdf has not been provided. Also in case of .doc data formats, we are not able to edit the data which contains images inside it. Data storage security in Cloud Computing is an area full of challenges and of paramount importance and many research problems are yet to be identified. We have envisioned several possible directions for future research on this area.

Fig.3 Data modification snapshot

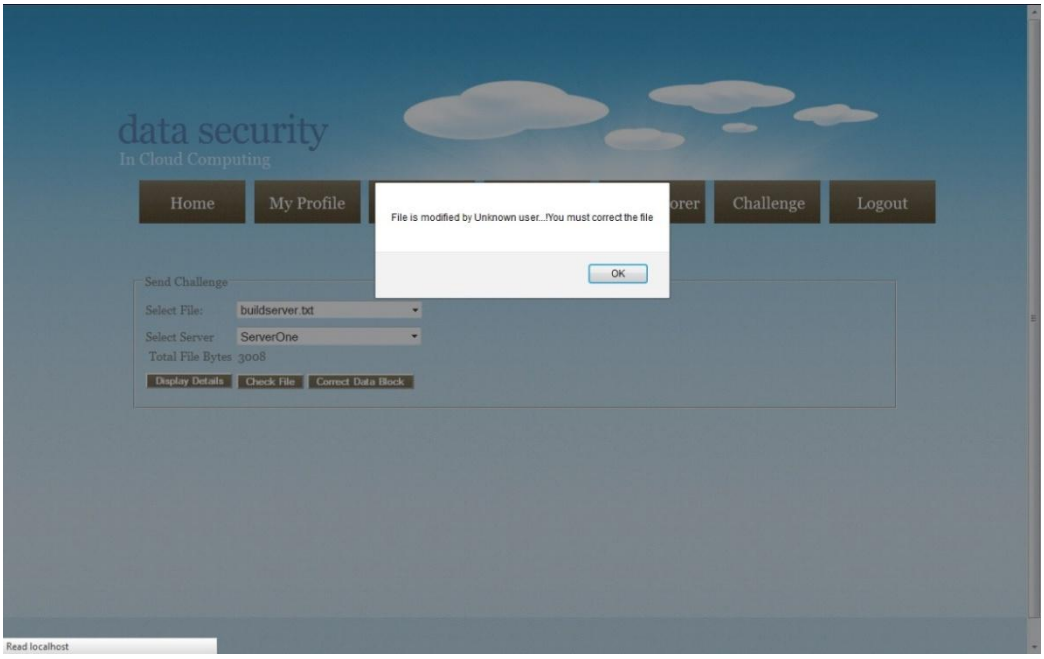

Fig. 4 Data modification detected snapshot.

Table 2: Cost of token pre-computation & file distribution.

File Size	Token Pre-computation Time (Sec)	File Distribution Time (Sec)
1 MB	7	9
2 MB	15	16
3 MB	22	24
4 MB	30	33
5 MB	42	44
6 MB	47	50
7 MB	54	59
8 MB	60	67
9 MB	71	77
10 MB	90	100

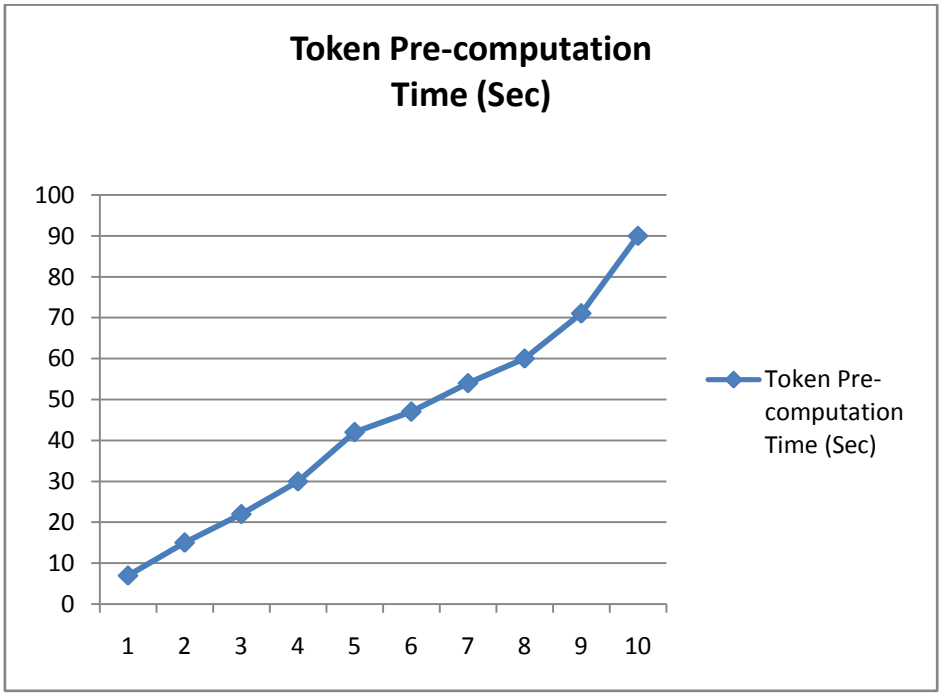

Fig. 5 Graph of token pre-computation time v/s file size

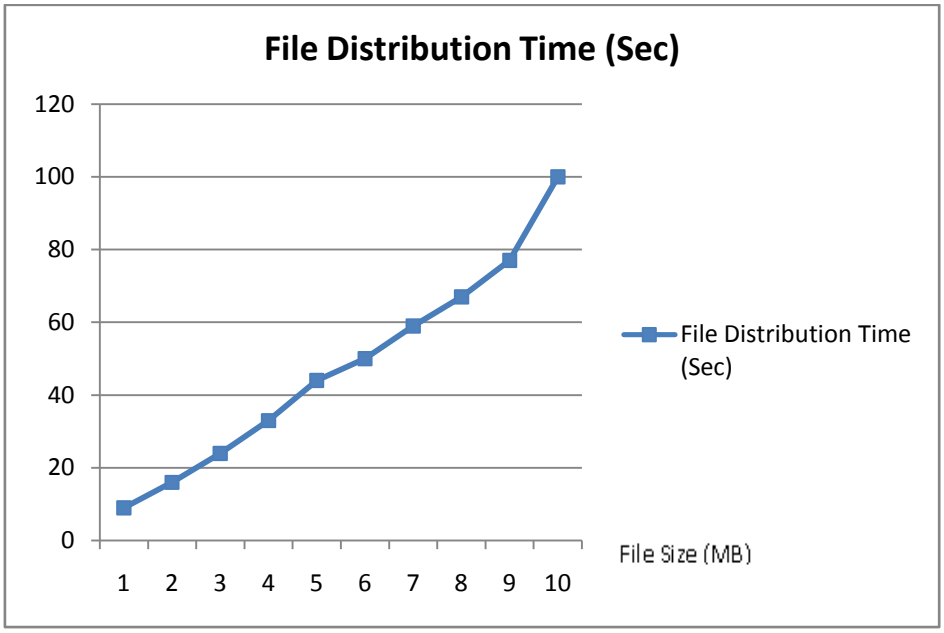

Fig. 6 Graph of File distribution time v/s file size

5. CONCLUSION

We have analyzed the data security concerns in cloud data storage, which is a distributed storage system. We proposed a distributed scheme to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud. To provide redundancy we used erasure correcting code in the file distribution preparation. As we all know cloud is not just a third party data warehouse. So providing support for dynamic operations is very important. Our scheme maintains the same level of storage correctness assurance even if users modify, delete or append their data files in the cloud. Challenge response protocol along with pre-computed token is used to verify the storage correctness of user's data & to effectively locate the malfunctioning server when data corruption has been detected. Through detailed performance analysis, we show that our scheme is having very low communication overhead & guarantees to detect every single unauthorized data modification. Our scheme has no limitation on number of pre-computed tokens used for challenging the cloud servers. Unlimited number of challenges can be made. We removed burden of calculating pre-computed tokens & storing the locally from the users. By splitting the file according to the number of server's we are added extra security to system. But we still believe that data storage security in Cloud computing is an area full of challenges and of paramount importance.

6. REFERENCES

[1] A. Juels, J. Burton, and S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. ACM CCS '07, Oct. 2007, pp. 584–97.

[2] G.Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. ACM CCS '07, Oct. 2007, pp. 598–609.

[3] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. SecureComm '08, Sept. 2008.

[4] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. Asia-Crypt '08, LNCS, vol. 5350, Dec. 2008, pp. 90–107.

[5] K. D. Bowers, A. Juels, and A. Oprea, "Hail: A High-Availability and Integrity Layer for Cloud Storage," Proc. ACM CCS '09, Nov. 2009, pp. 187–98.

[6] C.Wang, Qian Wang, Kui Ren, Wenjing Lou, "Ensuring Data Storage Security in Cloud Computing," Proc. IWQoS '09, July 2009, pp. 1–9.

[7] Q. Wang, C.Wang, Wenjing Lou, Jin Li, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. ESORICS '09, Sept. 2009, pp. 355–70.

[8] C. Erway, Alptekin, Charalampos Papamanthou, Roberto Tamassia, "Dynamic Provable Data Possession," Proc. ACM CCS '09, Nov. 2009, pp. 213–22.

[9] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-replica provable data possession," in Proc. of ICDCS'08. IEEE Computer Society, 2008, pp. 411–420.

[10] T. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in Proc. of ICDCS'06, 2006.

[11] N. Gohring, "Amazon's S3 down for several hours," Online at http://www.pcworld.com/businesscenter/article/142549/, amazons_s3_down_for_several hours.html, 2008.

[12] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions," Dec. 2006; http://www.techcrunch.com/2006/12/28/gmail-disaster-reports-of-massemail-deletions/

[13] Peter Mell, Tim Grance, "The NIST Definition of Cloud Computing", http://www.nist.gov/itl/cloud/upload-def-v15.pdf.

[14] 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/.

[15] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows and M. Isard, "A Cooperative Internet Backup Scheme*", Proc. of the 2003, USENIX Annual Technical Conference (General Track)*, pp. 29–41, 2003.

[16] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing" , IEEE Transactions on Parallel & Distributed Systems, Volume: 22, Issue: 5, pages: 847-859.

[17] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for storage security in cloud computing", in *Proc. of IEEE INFOCOM'10*, San Diego, CA, USA, March 2010.

[18] J. S. Plank and Y. Ding, "Note: Correction to the 1997 Tutorial on Reed-Solomon Coding," University of Tennessee, Tech. Rep. CS-03- 504, 2003.