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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 11, November 2014

Secure Distribution of File on Cloud

Niyamat I. Ujloomwale, Ranjana Badre

Dept. of Computer, MIT Academy of Engineering, Alandi, Savitri Phule Pune University, Pune, India

Dept. of Computer, MIT Academy of Engineering, Alandi,Savitri Phule Pune University, Pune, India

ABSTRACT: Cloud computing is worthy of consideration and try to build business systems as a way for businesses in this way can undoubtedly bring about lower costs, higher profits and more choice; for large scale industry, Data security has become the most important issue of cloud computing security. Though many solutions have been proposed, many of them only consider one side of security; this paper proposes the cloud data security must be considered to analyse the data security risk, the data security requirements, deployment of security functions and the data security process through encryption. Distribution of file is done on cloud servers with token generation. The security architecture of the system is designed by using encryption/decryption algorithm, which eliminates the fraud that occurs today with stolen data. There is no danger of any data sent within the system being intercepted, and replaced. The system with encryption is acceptably secure, but that the level of encryption has to be stepped up, as computing power increases. Results in order to be secured the system the communication between modules is encrypted. Since the customer does not have control over data the cloud provider should assure the customer that data is not modified. In this paper a data correctness scheme is proposed in which a Cloud service Provider assures the user that the data stored in the cloud is safe. This scheme also achieves the integration of storage correctness insurance and data error localization i.e., the identification of misbehaving server(s).

KEYWORDS: storage, security, challenge, token, correctness, error localization, misbehaving server

I. INTRODUCTION

Cloud computing is a very young concept and there is no consensus on a formal definition at the time of writing most experts agree that the cloud computing is a buzz which encompasses a variety of services. Other focus on the business model which is typically a pay–as–you-go service.

The following definition approaches cloud computing from a broad conceptual level: Cloud computing represents a broad array of web–based services aimed at allowing users to obtain a wide range of functional capabilities on “pay-as-you-go” basics that previously required tremendous hardware/software investments and professional skills to acquire. Cloud computing is the realization of the earlier ideals of utility computing without the technical complexities or complicated deployment worries. Although most definition do not use such generalized concepts, these generalizations are often implied as a base for other definitions. This makes the definition above highly applicable. as an addendum to the definition above, these key technical concept are often associated with (but not requires of) cloud computing : instantaneous and on-demand resource scalability, parallel and distributed computing ,and virtualization.

There are three general model of cloud computing (IaaS, SaaS, PaaS). Each of these models possesses a different impact on application security where an application is hosted in a cloud, there are two security problems are arise are:
1. How secure is Data?
2. How secure is code?

Security, Availability and Reliability are the major concerns of cloud service users. Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the software as a service (SaaS) computing architecture, are transforming data centres into pools of computing service on a huge scale. The increasing network bandwidth and reliable yet flexible network connections make it even possible that users can now subscribe high quality services from data and software that reside solely on remote data centres. Moving data into the cloud offers great convenience to buyers since they don’t have to care about the complexities of direct hardware management. The pioneer of Cloud Computing

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 11, November 2014

vendors, Amazon Simple Storage Service (S3) and Amazon Elastic Compute Cloud (EC2) are both well known examples.

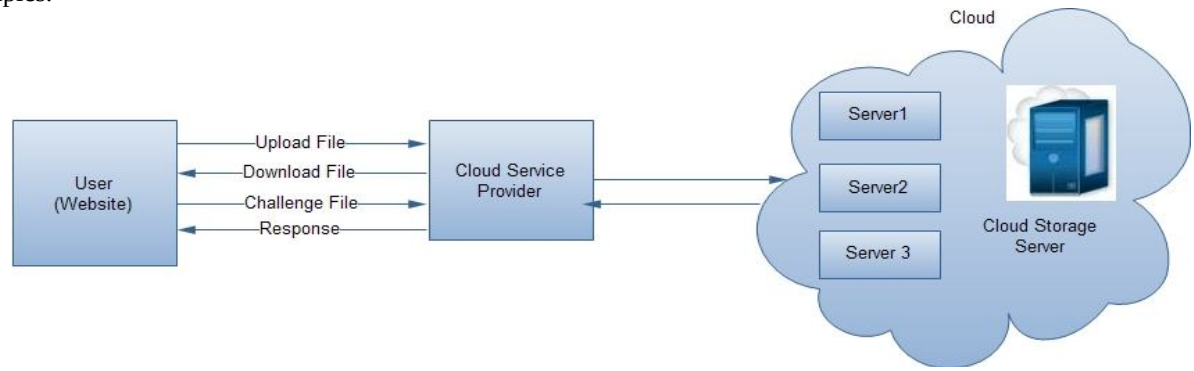

Fig. 1 System architecture

While these internet-based online services do provide huge amounts of storage space and customizable computing resources, this computing platform shift, however, is eliminating the responsibility of local machines for data maintenance at the same time as shown in Fig. 1. As a result, users are at the mercy of their cloud service providers for the availability and integrity of their data. Recent downtime of Amazon's S3 is such an example. To ensure the security and dependability for cloud data storage efficient mechanisms for dynamic data verification and operation goals are designed:

1. Storage correctness: to ensure users that their data are indeed stored appropriately and kept intact all the time in 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.
4. Dependability: to enhance the data availability against byzantine failures, malicious data modification and server colluding attacks, i.e. minimizing the effect brought by data errors or server failures.

II. RELATED WORK

An effective and flexible distributed scheme with explicit dynamic data support to ensure the correctness of users' data in the cloud was proposed by C. Wang, Q. Wang, K. Ren, and W. Lou in July 2009. C. Wang, Q. Wang, K.Ren, and W. Lou rely on erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability. This construction might drastically reduce the communication and storage overhead as compared to the traditional replication-based file distribution techniques. Their scheme achieves the storage correctness insurance as well as data error localization, that is, whenever data corruption has been detected during the storage correctness verification, their scheme can almost guarantee the simultaneous localization of data errors. Later in May 2011, Cong Wang, Qian Wang, Kui Ren, Wenjing Lou extended their work to allow user to audit the cloud storage with very lightweight communication and computation cost, proposed scheme that is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

A formal "Proof of Retrievability" (POR) model for ensuring the remote data integrity was described by A. Juels and J. Burton S. Kaliski in October 2007. Their scheme combines two methods spot-checking and error- correcting code to ensure both possession and retrievability of files on archive or backup service systems. H. Shacham and B.Waters in 2008 built on this model and constructed a random linear function based homomorphic authenticator which enables unlimited number of queries and requires less communication overhead. An improved framework for POR protocols that generalizes both Juels and Shacham's work was illustrated . All these schemes are focusing on static data. The effectiveness of their schemes rests primarily on the pre-processing steps that the user conducts before outsourcing the data file F. Any change to the contents of F, even few bits, must propagate through the error-correcting code, thus introducing significant computation and communication complexity was proposed by Bowers in 2009. The

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 11, November 2014

"provable data possession" (PDP) model for ensuring possession of file on untrusted storages was defined by Ateniese et al [10]. Their scheme utilized public key based homomorphic tags for auditing the data file, thus providing public verifiability. However, their scheme requires sufficient computation overhead that can be expensive for an entire file. Later in their subsequent work during 2008, described a PDP scheme that uses only symmetric key cryptography. This method has lower-overhead than their previous scheme and allows for block updates, deletions and appends to the stored file, which has also been supported in our work. However, their scheme focuses on single server scenario and does not address small data corruptions, leaving both the distributed scenario and data error recovery issue unexplored.

A new efficient means of polynomial in the size of the input(i.e. key or data) was proposed by M. A. Shah, R. Swaminathan, and M. Baker during the year 2008 in "Privacy Preserving audit and extraction of digital contents". The main threat from the auditor is that it may glean important information from the auditing process that could compromise the privacy guarantees provided by the service. For example, even a few bits from a file containing medical history could reveal whether a customer has a disease. To ensure privacy, there exist different standards for the encrypted data and the encryption key. For the data, the system relies on (1) the strength of the encryption scheme and (2) the zero-knowledge property of the protocol for encryption-key audits.

To ensure file integrity across multiple distributed servers, using erasure-coding and block-level file integrity checks was proposed by T. S. J. Schwarz and E. L. Miller in 2009. However, their scheme only considers static data files.

To verify data integrity using RSA-based hash for data possession in peer-to-peer file sharing networks was defined by D. L. G. Filho and P. S. L. M. Barreto in 2006. However, their proposal requires exponentiation over the entire data file, which is clearly impractical for the server whenever the file is large.

III. PROPOSED ALGORITHM

Files data is divided into equal data vector length. Data vector length=tot. no. of bytes in file/no. of servers. This divides the file into n number of blocks. 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).Multiplication of vector and A is performed on each data vector. Let it be $g_1, g_{2,\ldots}$ and so on.

```
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 Gm [i] =sum;
End for
count = count + 1;
End For
End For
```

A 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.

Step 1. Choose the parameter r for number of indices per verification and pseudorandom function *f*, pseudorandom Permutation function Φ

Step 2. Choose the parameters *t*, the number of tokens to be generated and *l* the length of the data vector.

Step 3. Choose α as a vector to compute the tokens and V to store the tokens.

Step 4. 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)

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 11, November 2014

$P(x)=1x^8+1x^7+0x^6+0x^5+0x^4+0x^3+1x^2+1x^1+1x^0$
For j=1 to n Do
For i =1 to t Do
A[i] = *f* (i)
sum = 0
For q =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= 2 Then V3 [i] = V3 [i] + sum
End For
Vj =Add V1, V2, V3
Step 5: Store the token into Data storage If j= 1 Then V2 [i] = V2 [i] + sum

B Correctness verification 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.
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. 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.

C Error recovery

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.

Step 1. The block corruption is detected in specified r rows using above algorithm.
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.

D Dynamic data support

In today's scenario, dynamic data required for many applications like electronic documents, log files, medical data etc. Therefore it is crucial to consider the dynamic case, where user can do the various block level operations like update, append to modify the data file, simultaneously it should maintain the storage correctness.

Suppose user wants to update the data block *mi*. After update the data block *mi* becomes *mi'*. 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 *mi* block are erased and replaced with new tokens generated for *mi'*. This can be done with the homomorphic tokens.

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 11, November 2014

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*.

IV. PERFORMANCE ANALYSIS

Its measure considered are:

i) Time required for generating the tokens.

ii) Communication time (auditing) to upload and download a file.

Firstly implemented proposed 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 were connected to each other with LAN. Then hosted on aws.amazon.in.

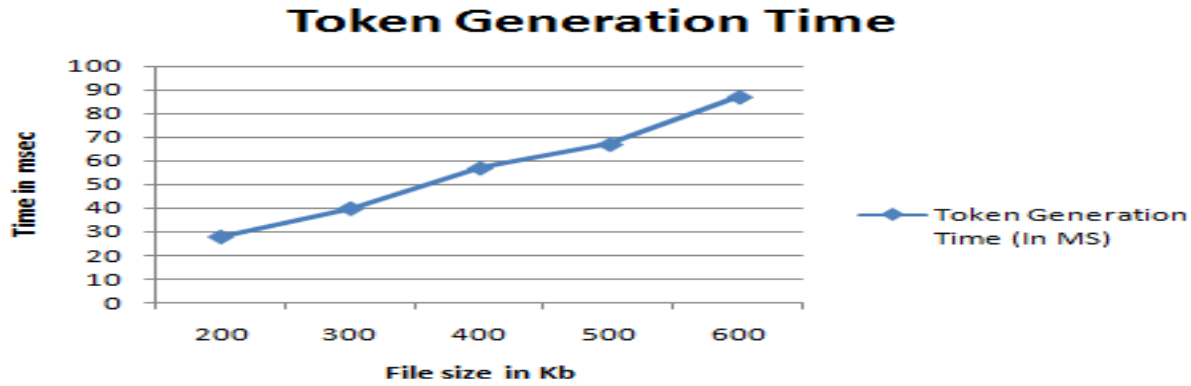

Fig.2 Graph for time required for generating token

The proposed system reduces the maintenance and communication cost because there is no need to store complete file on each of 3 storage servers. Token generation time is given in ms as shown in Fig. 2 .For e.g. if the file size is 200 kb, the time required to generate the token is 30 ms.

To prevent data dropout or corruption, the integrity of data stored in each server needs to be periodically checked. User 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.

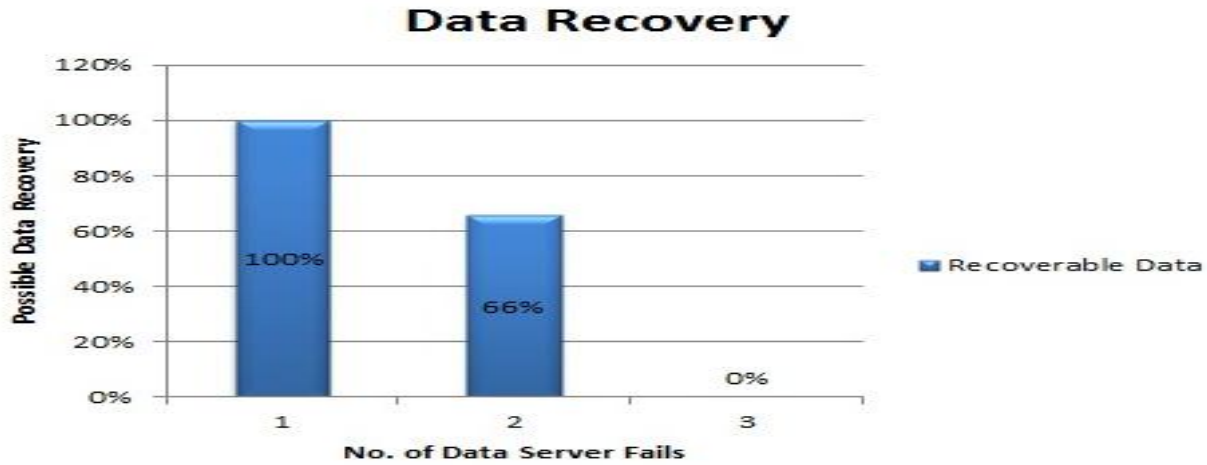

Fig. 3 Graph for data recovery

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

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 11, November 2014

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. To ensure the data reliability in distributed storage systems, various data redundancy techniques can be employed, such as replication, erasure codes, and network coding. In Fig. 3, the data availability is the probability that data users could recover original data from any *k*-combination of *n* storage servers. For e.g. On server one, file part1 and file part2 are stored. On server two, file part2 and file part3 and on server three, file part3 and file part1 are stored. In the mentioned scenario, even if one server misbehaves, 100% data can be recovered.

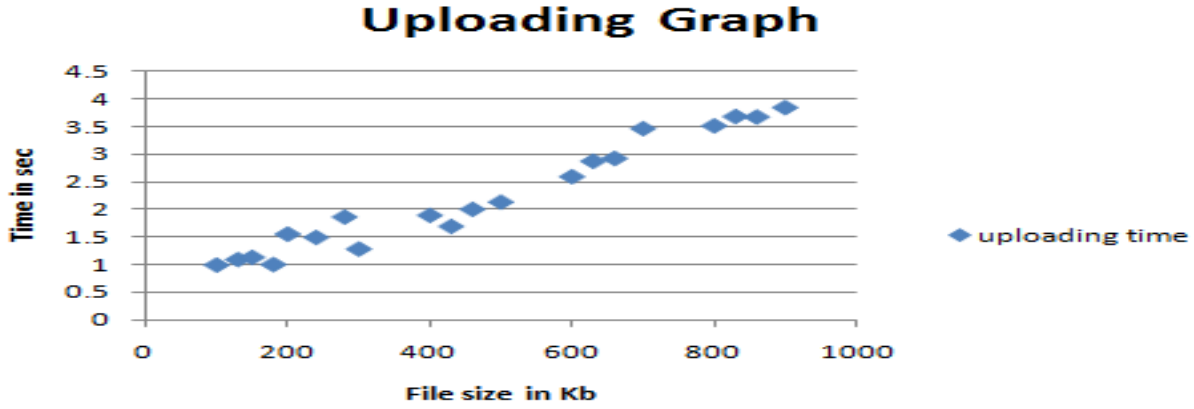

Fig.4 Graph for uploading file on cloud

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 required data from a healthy server. Fig.4 shows time required to upload a file on cloud. If file size is less, uploading time required will also be less.

V. CONCLUSION AND FUTURE WORK

The implemented flexible and distributed scheme provides the method for data security, which is the major parameter of the quality of service. The 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.

The scheme provides the block operations like update and append for dynamic data. This also ensures the data availability in case of communication link failure or particular server misbehaves. The proposed system is restricted to text files. In future will work for images, video, audio files.

REFERENCES

1. C. Wang, Q. Wang, K. Ren, and W. Lou, 'Ensuring data storage security in cloud computing', in Proc. of IWQoS, pp.1–9, July 2009.
2. A. Juels and J. Burton S. Kaliski, 'Pors: Proofs of retrievability for large files', in Proc. of CCS, Alexandria, VA, pp.584–597, October 2007.
3. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson,and D. Song, 'Provable data possession at untrusted stores', in Proc. of CCS, Alexandria, VA, pp. 598–609, October 2007.
4. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, 'Enabling public verifiability and data dynamics for storage security in cloud computing,', in Proc. of ESORICS, volume 5789 of LNCS. Springer-Verlag, pp. 355–370, Sep. 2009.
5. R. Curtmola, O. Khan, R. Burns, and G. Ateniese, 'Mr-pdp: Multiple-replica provable data possession', in Proc. of ICDCS, IEEE Computer Society, pp. 411–420, 2008.
6. K. D. Bowers, A. Juels, and A. Oprea, 'Hail: A high-availability and integrity layer for cloud storage', in Proc. of CCS, pp. 187–198, 2009
7. C. Wang, K. Ren, W. Lou, and J. Li, 'Towards publicly auditable secure cloud data storage services', IEEE Network Magazine,vol. 24, no. 4, pp. 19–24, 2010.
8. G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. of SecureComm, pp. 1–10, 2008.
9. H. Shacham and B. Waters, 'Compact proofs of retrievability', in Proc. of Asiacrypt, volume 5350 of LNCS, pp. 90–107, 2008.