IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 2, February 2017

An Enhanced Reliable and Protective Method for flexible distributed Storage in Cloud Computing

B. Durgalakshmi[1], S. Rakoth Kandan[2]

Research Scholar, School of Computing Science, VIT University, Chennai, Tamil Nadu, India [1]

Research Scholar, Dept. of CSE, Annamalai University, Chidambaram, Tamil Nadu, India[2]

Abstract: Cloud applications can utilize the file management system without the trouble of fixings. We are detaching some guarantee contingency against the regularity and availability in the cloud environment. We propose in this paper to enhance the flexible distributed storage with very lightweight communication and computation cost. It does not only ensure strong cloud storage, but also achieves fast data error localization. The proposed design further supports secure and efficient dynamic operations on outsourced data including block modification, deletion, and append[3][4]. Analysis shows the proposed scheme is highly efficient and flexible against Byzantine failure, malicious data modification attack, and even server colluding attacks.

Keywords: Trustworthy, Cloud storage, Key generation, Token Generation.

I. INTRODUCTION

We investigate the problem of data security in cloud data storage, which is essentially a distributed storage system. Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagram (figure 1). Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing is basically a resource that you can utilize online to maintain your business' platform while you worry about other basic assets about your business. It basically changes how online developers meet their criteria for their IT software. Cloud computing is always advancing and changing each year so that it will work better for personal uses and business uses.

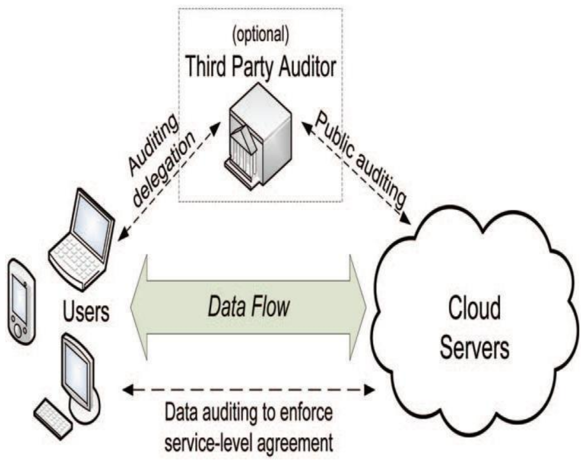

Fig. 1. Cloud storage service architecture

We are having the Third-Party Auditor (TPA) for the purpose of authentication between the users, cloud servers and Service Level Agreement (SLA) that is a negotiated agreement between two parties, where one is the user and the other is the service provider. This can be a legally binding formal or an informal contract (Ex: Internal department relationships). Contracts between the service provider and other third-party auditors are often called SLAs – because the level of service has been set by the customer, there will be no "agreement" between third parties these agreements we can say simply a "contract." Our scheme supports an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append operations.

II. PROBLEM STATEMENT

2.1 System Model

We use three different network entities as illustrated in the Figure 1

User: Data to be stored in the cloud environment and used for data storage and computation can be enterprise by a user.

Cloud Server (CS): An entity to provide data storage service and has sufficient storage space and computation resources. It is managed by cloud service provider.

Third-Party Auditor: A TPA, who has capabilities that users may not have, is hope to apprise and expose risk of cloud storage services on behalf of the users upon request.

As users it is difficult to maintain their data for long time, it is important to ensure that their data are correctly stored and maintained. By implementing the service-level agreement in cloud environment we can ensure that data are maintained securely. Some time the user don't have time to maintain the data so it may be hacked by any other person to avoid that we are using here (TPA) Third-Party Auditor for security purposes.

1) In existing systems it only provide binary form of results about storage status across the distributed servers we propose in this paper a flexible distributed storage integrity auditing mechanism, utilizing the homomorphic token[7] and distributed erasure-coded data[6][8].
2) The proposed design allows users to audit the cloud storage with very lightweight communication and computation cost.
3) The new scheme further supports secure and efficient dynamic operations on data blocks, including: update, delete, and append.
4) Our scheme is resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

2.2 Impugn Model

From user's aspect, the impugn model has to find out all kinds of threats toward his cloud data honestness. Because here we are dealing with the two types of attackers: Internal and External attacks. For internal attacks, a CSP will be self-interested, untrusted, and possibly malicious, usually this internal attackers are hacking at the time of they are dealing with that data this type of attacking is entirely depends on internal persons who are all working in one organization and also it may also attempt to keep secrecy of a data loss incident due to management errors, Byzantine failures, and some other things. For external attacks, data loss will be entirely come from the outsiders who are all having the capability of control the cloud service provider for example, some attackers will consider about the money. They may be reduce the total capacity of accessing the storage of cloud servers and also modifying users data but this cannot be identified by cloud service provider.

2.3 Design Goals

In cloud data storage for assuring the safety measure and regularity for under the previous impugn model, we focus to design 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.
4. Dependability: To enhance data availability against Byzantine failures, malicious data modification and server colluding attacks, i.e., minimizing the effect brought by data errors or server failures.
5. Lightweight: To enable users to perform storage correctness checks with minimum overhead.

III. SYSTEM FORMATION

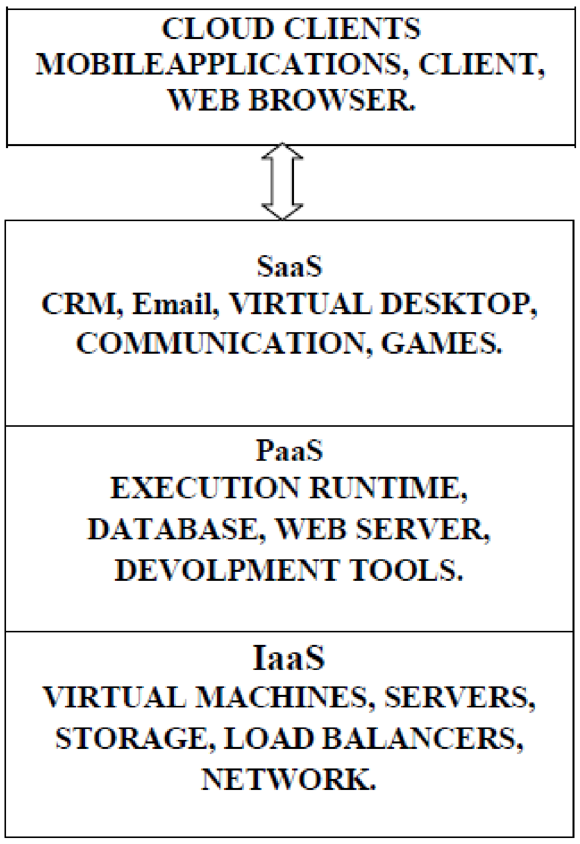
CLOUD CLIENTS
MOBILEAPPLICATIONS, CLIENT, WEB BROWSER.
SaaS
CRM, Email, VIRTUAL DESKTOP, COMMUNICATION, GAMES.
PaaS
EXECUTION RUNTIME, DATABASE, WEB SERVER, DEVOLPMENT TOOLS.
IaaS
VIRTUAL MACHINES, SERVERS, STORAGE, LOAD BALANCERS, NETWORK.

Fig. 2. AaaS, SaaS, PaaS, and IaaS

3.1 Putting Server Access Restore Point and Time Management

To avoid server failure in every data inclusion or any other activity by unauthorized person or any internal and external attack coming into the CSP address domain, one server access point or restore point is given to the cloud server when the client is doing some delete, modification, and append operations in his will. This is a new technique that we have introduced in our system design. The reason for which we want to do this restore point is that if there is any server failure and crash then, this restore access point helps a lot to recover everything that we have lost before that crash time itself. This process is proceeded by fixing one access point to server database (it is done with the help of CSP, because we don't have any copy of our outsourced data) when we finally finish our data exchange in cloud server. Also since we don't have any physical possession of our data in cloud server we can't have any separate login access using cryptographic key [1] for that. It is a major drawback to our cloud server to maintain our individual or group data. In the days to come, it can be rectified with the help of CSP. Here one time management is fixed in cloud database management with the help of CSP"s access or automatically in server failure when user stores, deletes, modifies and appends the data and this time management is used to know the restore access point easily by CSP and clients when servers meet such failure.

3.2 File Distribution Preparation

3.2.1 File Encode

It is well known that erasure-correcting code may be used to tolerate multiple failures in distributed storage systems. 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. An (m,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 m+k. For support of efficient sequential I/O to the original file, our file layout is systematic, i.e., the unmodified m data file vectors together with k parity vectors is distributed across m+k different servers.

Let **F**=(F1,F2,…..,Fm) and Fi= (F1i, F2i, ….,Fli)^T Here, T (shorthand for transpose) denotes that each Fi is represented as a column vector, and l denotes data vector size in blocks. All these blocks are elements of GF(2p). The systematic layout with parity vectors is achieved with the information dispersal matrix A, derived from an m × (m+k) vandermonde matrix.

$$\begin{pmatrix} 1 & 1 & \cdot\ \cdot & 1 & 1 & \cdot\ \cdot & 1 \\ \beta_1 & \beta_2 & \cdot\ \cdot & \beta_m & \beta_{m+1} & \cdot\ \cdot & \beta_n \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot & \cdot & \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot & \cdot & \cdot\ \cdot & \cdot \\ \beta_1^{m-1} & \beta_2^{m-1} & \cdot\ \cdot & \beta_m^{m-1} & \beta_{m+1}^{m-1} & \cdot\ \cdot & \beta_n^{m-1} \end{pmatrix}$$

where βj (j∈ {1,,,,n}}) are distinct elements randomly picked from GF(2Þ). After a sequence of elementary row transformations, the desired matrix A can be derived as

$$\mathbf{A= (I|P)=} \begin{pmatrix} 1 & 0 & \dots & 0 & p_{11} & p_{12} & \dots & p_{1k} \\ 0 & 1 & \dots & 0 & p_{21} & p_{22} & \dots & p_{2k} \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ 0 & 0 & \dots & 1 & p_{m1} & p_{m2} & \dots & p_{mn} \end{pmatrix}$$

Where, I is a m x m identity matrix and P is the secret parity generation matrix with size m x k. Note that A is derived from a Vandermonde matrix, thus it has the property that any m out of the m + k columns form an invertible matrix. By using F and P, the user obtains the encoded file

R = F X P, Where R is a matrix of order m x k that is derived by multiplying F and P.
G = (F1, F2,….. Fm, R1, R2, … Rk)

Where, Fi denotes data blocks, $1 \le i \le m$
Rj denotes jth column of matrix R, $1 \le j \le k$

3.2.2 Token Precomputation

In order to achieve assurance of data storage correctness and data error localization simultaneously, our scheme entirely relies on the precomputed verification tokens. The main idea is as follows: Before file distribution the user precomputes a certain number of short verification tokens on individual vector, G(j) (j∈ {1…n)}) where, j = 1 to n , each token covering a random subset of data blocks. Later, when the user wants to make sure the storage correctness for the data in the cloud, he challenges the cloud servers with a set of randomly generated block indices. Upon receiving challenge, each cloud server computes a short "signature" over the specified blocks and returns them to the user. The values of these signatures should match the corresponding tokens precomputed by the user. Meanwhile, as all servers operate over the same subset of the indices, the requested response values for integrity check must also be a valid codeword determined by the secret matrix P.

Suppose the user wants to challenge the CS for 't' times to ensure the correctness of data storage, then, he must compute t verification tokens for each G(j) (j∈ {1…n)}), using a PRF f(.) , a PRP Ø(.), a challenge key kchal, and a master permutation key KPRP. Specifically, to generate the ith token for the server j, the user acts as follows:

1. Derive a random challenge value αi of GF(2p) by αi= f kchal (i) and a permutation key kprp(i) based on KPRP.
2. Compute the set of r randomly-chosen indices

{Iq∈ [1,,,,,,,, l]|$1 \le q \le r$} , where Iq = Øk pr p(i) (q)

3. Calculate the token as

$$vi\ (j) = \sum_{q=1}^{r} q\alpha i + G(j)\ [Iq],$$

Where G(j) [Iq] = gIq(j)

Note that $v_i^{(j)}$, which is an element of GF(2^p) with small size, is the response the user expects to receive from server j when he challenges it on the specified data blocks. After token generation, the user has the choice of either keeping the precomputed tokens locally or storing them in encrypted form on the cloud servers. In our case here, the user stores them locally to obviate the need for encryption and lower the bandwidth overhead during dynamic data operation which will be discussed shortly. The details of token generation are shown in the following algorithm.

```
ALGORITHM: Token_Precomputation (G)
{
/*
INPUT:
G – Encoded File.
Kchal – A challenge key.
Kprp – Key for a master permutation.
n – Number of distributed servers.
t – Number of tokens.

FUNCTIONS:
f () – Pseudorandom Function (PRF)
Ø () – Pseudorandom Permutation (PRP)
OUTPUT:
Vi – Tokens.
*/
for j=1 to n do // for each vector
{
for i=1 to t do //for each round
                {
  αi = f kchal (i) and kprp(i)
r
vi (j) = Σ qαi + G(j) [Øk pr p(i) (q)]
q=1
        }
}
  }
```

Once all tokens are computed, the final step before file distribution is to blind each parity block $gi^{(j)}$ in ($G^{(m+1)}$,……., $G^{(n)}$) by

$gi^{(j)} \leftarrow gi^{(j)} + f\ kj\ (sij)$, $i \in \{1….. l\}$,

Where kj is the secret key for parity vector G(j),($j \in \{m+1,….., n\}$). This is for protection of the secret matrix P. After blinding the parity information, the user disperses all the n encoded vectors G(j) across the cloud servers S1,S2, . . . ,Sn.

For verifying Correctness and localizing errors the algorithm CHALLENGE () that is proposed in [1] can be used.

3.2.3 File Renovation

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. That our verification scheme is based on random spot-checking, so the storage correctness assurance is a probabilistic one.

IV. DATA BLOCK DYNAMIC PROCESS

In our proposed system we can ensure our data will be protected by doing the following operations such as append, deletion, and update. These are some operations performed by an user in the cloud environment.

4.1 Append Operation in Server Block

In this operation we assume that a user having some GB of memory space allotted by a service provider for the user application purposes. CSP will not restrict a user to access for particular applications and user can access any kind of applications also, The service provider is to provide all kind of access to a user. The user can include some data with existing data, change some data or remove some data because of this we can give assurance to the data integrity[5]. It is very efficient method in our proposed design compared to any other such type.

4.2 Deletion Operation in Server Block

In this deletion operation, the user has to compare his data with the existing servers' data. After this process, user may perform the deletion operation which is shown in Figure 3. If there is number of servers for data selection while deleting the data that are stored in the particular storage server is considered for this. The required data can be deleted[1].

4.3 Update Operation in Server Block

In cloud data storage, a user may need to modify some data block(s) stored in the cloud, from its current value fij to a new one, fij + Δfij. This operation is referred as data update. Figure 3 gives the high level logical representation of data block update. Due to the linear property of Reed-Solomon code, a user can perform the update operation and generate the updated parity blocks by usingΔ fij alone, without involving any other unchanged blocks.

V. EFFICIENCY ANALYSIS

5.1 Efficiency of File Encoding

The content of file F is encoded and stored in G by adding the first 'm' columns of F to G and the next 'k' columns are multiplied with the matrix P and appended in G.
Time taken to multiply Fm+1..n with P is calculated as M(m)

M(m) – Number of multiplications needed to multiply $F_{m+1..n}$ by P.

$$M(m) = \sum_{h=1}^{m} \sum_{i=1}^{m} \sum_{j=1}^{k} 1 = m^2k$$

$M(m) \in O(m^2)$ for small values of k.

$M(m) \in O(m^3)$ for k which is equal to m.

5.2 Efficiency of Token Precomputation

To distribute the file in cloud environment 'n' vectors are generated and 't' tokens are generated for each vector. We propose a method to increase the efficiency of the algorithm that is proposed in [1] for token precomputation. In general, since the multiplication is performed as repeated addition the time taken for multiplication operation is more than the addition operation.

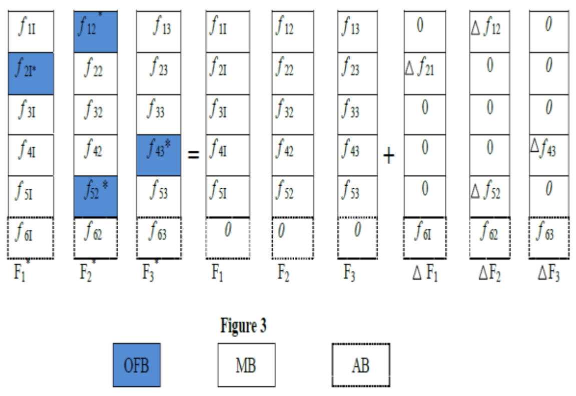

Figure 3

Similarly exponentiation is performed as repeated multiplication, so the time taken by exponentiation is more than multiplication. We use multiplication instead of exponentiation and addition instead of multiplication to increase the efficiency. Definitely this change will not affect the security aspect of the data and also correctness will be maintained.

VI. CONCLUSION

In this paper, we analyze the problem of data security in cloud data storage, which is essentially a distributed storage system. To achieve the assurances of cloud data integrity and availability and enforce the quality of dependable cloud storage service for users, we propose an enhanced effective and flexible scheme with unambiguous dynamic data support, including block update, delete, and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data reliability. 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.

ACKNOWLEDGMENT

The authors gratefully acknowledge the authorities of Annamalai University and VIT University for the facilities offered to carry out this work.

REFERENCES

[1] C.Wang, Q.Wang, K.Ren, Ning Cao, and W.Lou, "Toward Secure and Dependable Storage Services in Cloud Computing" IEEE Services Computing, Vol. 5, No. 2, April-June 2012.

[2] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," Proc. 17th Int'l Workshop Quality of Service (IWQoS '09), pp. 1-9, July 2009.

[3] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf.Security and in Comm. Netowrks (SecureComm '08), pp. 1-10, 2008.

[4] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. 14th European Conf. Research in Computer Security (ESORICS '09), pp. 355-370, 2009.

[5] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proc. 16th ACM Conf. Computer and Comm. Security (CCS '09), pp. 213-222, 2009.

[6] T. Schwarz and E.L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '06), pp. 12-12, 2006.

[7] L. Carter and M. Wegman, "Universal Hash Functions," J.Computer and System Sciences, vol. 18, no. 2, pp. 143-154, 1979.

[8] J. Hendricks, G. Ganger, and M. Reiter, "Verifying Distributed Erasure-Coded Data," Proc. 26th ACM Symp. Principles of Distributed Computing, pp. 139-146, 2007.