IOSR Journal of Engineering
e-ISSN: 2250-3021, **p**-ISSN: 2278-8719,
Vol. 2, Issue 12 (Dec. 2012) ||V2|| PP 17-21

Ensuring Data Storage Security in Cloud Computing

Rampal Singh, Sawan Kumar, Shani Kumar Agrahari
Department of computer science & engineering Institute of Technology And Management AL-1 Sector-7 Gida Gorakhpur

***Abstract*:- Cloud computing is a new computational paradigm that offers an innovative business model for organization to adopt IT without upfront investments. Despite the potential gain achieved from the cloud computing. It is clearly one of today's most enticing technology area due, at least in part, to its cost-efficiency and flexibility.Cloud computing moves the application software and database to the large data center where the data management and services may not be fully trustworthy. In this article our major discussion on the cloud data storage security. The security is an important aspect of quality of services. To ensures the correctness of user data in cloud. We propose an effective and flexible distribution scheme two way handshakes based on token management. By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the integration of storage correctness insurance and data error localization i.e., the identification of misbehaving server(s).**

Keyword:- *Cloud computing, data security, ,* cloud storage, public auditing, integrity.

I. INTRODUCTION

Today , the 14th largest software company by market capitalization (salesforce.com) operates almost entirely in the cloud, the top five software companies by sales revenue all have major cloud offerings, and the market as a whole is predicted to grow to $160 Billion by 2011 (source: Merrill Lynch). Yet despite the trumpeted business and technical advantages of cloud computing, many potential cloud users have yet to join the cloud, and those major corporations that are cloud users are for the most part putting only their less sensitive data in a cloud. Lack of control in the cloud is the major worry. One aspect of control is transparency in the cloud implementation – somewhat contrary to the original promise of cloud computing in which the cloud implementation is not relevant. Transparency is needed for regulatory. [3]
Cloud computing represents a recent paradigm shift for the provision of computing infrastructure which outsources computation and storage requirements of applications and services to a managed infrastructure. Cloud computing inevitably poses new challenging security threats for number of reasons.

- ➢ Cryptographic primitives for the purpose of data security protection ca not be directly adopted due to the users‟ loss control of data in cloud computing. The problem of verifying correctness of data storage in cloud is becomes even more challenging.
- ➢ Cloud computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, like deleting, modification, insertion, recording, etc.to ensure storage correctness under dynamic data update, this dynamic feature also makes traditional integrity insurance technique futile and entails new solutions.

In this paper, we propose an effective and flexible distributed scheme with explicit dynamic data support to ensure the correctness of users‟ data in the cloud .we rely on erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability. This construction drastically reduces the communication and storage overhead as compared to the traditional replication-based file distribution techniques. By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the storage correctness insurance as well as data error localization: whenever data corruption has been detected during the storage correctness verification, our scheme can almost guarantee the simultaneous localization on data errors, i.e., the identification of the misbehaving server(s) [3].

II. LITERATURE SURVEY

Cloud Computing

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 services.

The following definition approaches cloud computing from a broad conceptual level:
Cloud computing represents a broad array of web –based s services aimed at allowing users to obtain a wide range of functional capabilities on a "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 (bt not requires of) cloud computing : instantaneous and on-demand resource scalability, parallel and distributed computing ,and virtualization.

Uses of Cloud Computing

Various authors have proposed three different tiers of systems employed by cloud service provider []. These tiers make up the different levels of technologies used in cloud computing.

Infrastructure as a Service (IaaS)

This level represents the most computational and storage (e.g., Microsoft, Google, Amazon) manage a vast set of computational and storage resources. Depending on the provider, end users may have direct access of the hardware resources or access to a set of virtual resources. Clouds typically utilize virtual resources and grid applications typically have direct access to hardware. Application and service built upon virtual resources sets are not hardware dependent and can be deployed seamlessly across different cloud platforms. This service is best representing by services like Amazon EC2, a virtual machine platform.

Platform as a Service (PaaS)

At the next level of services are presented to users as a software /application platform instead of hardware. Typically this layer consists of application frameworks that make up the basis of the SaaS layer describe next. The Google APP Engine and Microsoft Azure both offer a large set of programming tools at this level.

Software as a Service (SaaS)

This is the highest level of services provided by cloud platforms. This level provides applications that end users interact with. Examples include Google Docs, Microsoft Office live, Google Maps and Face book.

B. Security issues and challenges

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. [2]

III. PROBLEM STATEMENT

From the perspective of data security, which has always been an important aspect of quality of service, Cloud Computing inevitably poses new challenging security threats.

1. Firstly, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted due to the users loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. Considering various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of verifying correctness of data storage in the cloud becomes even more challenging.
2. Secondly, Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion, deletion, modification, appending, reordering, etc. To ensure storage correctness under dynamic data update is hence of paramount importance.
3. Third is the deployment of cloud computing, it is powered by data centers running in a simultaneous cooperated and distributed manner. Individual user's data is redundantly stored in multiple physical locations to further reduce e data integrity threats. [1]

Notation and 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 (2p) for p = 8 or 16.
- A – The dispersal matrix used for Reed-Solomon coding.
- G – The encoded file matrix, which includes a set of n = m + k vectors, each consisting of l blocks
- f key (•) – pseudorandom function (PRF), which is defined as f : {0, 1}∗ × key → GF (2p).
- φ key (•) – pseudorandom permutation (PRP), which is defined as φ : {0, 1}log2 (ℓ) × key → {0, 1}log2 (ℓ)
- ver – a version number bound with the index for individual blocks, which records the times the block has been modified. Initially we assume ver is 0 for all data blocks.

ENSURING CLOUD DATA STORAGE

In cloud data storage system, users store their data in the cloud and no longer possess the data locally. Thus, the correctness and availability of the data files being stored on the distributed cloud servers must be guaranteed. Our main scheme for ensuring cloud data storage is presented in this section. The first part of the section is devoted to a review of basic tools from coding theory that is needed in our scheme for file distribution across cloud servers. Then, the homomorphic token is introduced. The token computation function we are considering belongs to a family of universal hash function, chosen to preserve the homomorphic properties, which can be perfectly integrated with the verification of erasure-coded data. Subsequently, it is shown how to derive a challenge-response protocol for verifying the storage correctness as well as identifying misbehaving servers. Finally the procedure for file retrieval and error recovery based on erasure- correcting code is also outlined.

File distribution preparation

It is well known that erasure-correcting code may be used to tolerate multiple failures in distributed storage systems. In clodata 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 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 together with k parity vectors is distributed across m + k different servers.

Let **F** = (F1 , F2 , . . . , Fm) and Fi = (f1i , f2i, . . . , fli)T (i ∈ {1, . . . , m}). Here T (shorthand for transpose) de-notes 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 a an m×(m+k) Vandermonde matrix[7] :

1 1 … 1 1 … 1

β1 β2 … βm βm+1 … βn

. . . .

. . . .

β1m-1 β2m-1 … βmm-1 … βm+1m-1 βnm-1

where βj (j ∈ {1, . . . , n}) are distinct elements randomly Picked from GF (2p).

After a sequence of elementary row transformations, the desired matrix A can be written as

1 0 . . . 0 P11 P12 . . . P1k

0 1 . . . 0 P21 P22 . . . Pmk

A = (I|P)= . . . .

. . . .

. . . .

0 0 . . . 1 Pm1 Pm2 . . . Pmk

Where I is a m × m identity matrix and P is the secret parity generation matrix with size m × 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 multiplying **F** by **A**, the user obtains the encoded file:

G = F · A = (G(1) , G(2) , . . . , G(m) , G(m+1) , . . . , G(n))

= (F1, F2 , . . . , Fm , G(m+1) , . . . , G(n)),

Where G(j) = (g1(j), g2(j), . . . , gl(j))T (j ∈ {1,…,n }).

noticed, the multiplication reproduces the original data file vectors of **F** and the remaining part (G(m+1) , . . . , G(n)) are k parity vectors generated based on **F**.

IV. CHALLENGE TOKEN PRE-COMPUTATION

In order to achieve assurance of data storage correctness and data error localization simultaneously, our scheme entirely relies on the pre-computed verification tokens. The main idea is as follows: before file distribution the user pre-computes a certain number of short verification tokens on individual vector G(j) ($j \in \{1, \ldots, n\}$), 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 pre-computed 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 deter- mined by secret matrix P.

Algorithm 1

```
Token Pre_computation
1: **procedure**
2: Choose parameters l, n and function f, φ;
3: Choose the number t of tokens;
4: Choose the number r of indices per verification;
5: Generate master key Kprp and challenge key kchal ;
6: **for** vector G(j) , j ← 1, n **do**
7: **for** round i← 1, t **d**
8: Derive αi = fkchal (i) and kprp from Kprp 9: compute
10 end for
11: end for
12: store all the vis locally
13: end procedure
```

Suppose the user wants to challenge the cloud servers ‟ t times to ensure the correctness of data storage. Then, he must pre-compute t verification tokens for each G(j) ($j \in \{1, \ldots, n\}$), using a PRF f (·), a PRP φ(·), a challenge key kchal and a master permutation key KP RP . Specifically, to generate the ith token for server j, the user acts as follows:

1) Derive a random challenge value αi of GF (2p) by

αi = Kchal (i) and a permutation key KPRP(I) based on KPRP

2) Compute the set of r randomly-chosen indices:

$\{ I_q \in \{ [1, ..., l] | 1 \leq q \leq r\}$, where $I_q = \varphi\ k\ prp\ (i)\ (q)$.

3) Calculate the token as:

$V_i(j) = \Sigma_{q=1}^{r} \alpha_i\ q * G(j) [I_q]$, where $G[I_q] = g_{iq}(j)$

After token generation, the user stores token locally to obviate the need for encryption and lower the bandwidth during dynamic data operation.

V. CORRECTNESS VERIFICATION AND ERROR LOCALIZATION

Our scheme outperforms those by integrating the correctness verification and error localization in our challenge-response protocol: the response values from servers for each challenge not only determine the correctness of the distributed storage, but also contain information to locate potential data error.

Specifically, the procedure of the i-th challenge-response for a cross-check over the n servers is described as follows:

1) The user reveals the αi as well as the i-th permutation key kprp(i) to each servers.

2) The server storing vector G(j) ($j \in \{1, \ldots, n\}$) aggregates those r rows specified by index kprp(i) into a linear combination

$R\ i(j) = \Sigma_{q=1}^{r} \alpha_{iq} * G(J) [\phi\ K_{prp}(i)\ (q)]$.

3)Upon receving Ri(j) s from all the servers, the users takes away blind values in R(j) ($j \in \{ m=1,\ldots,n\}$) by

$R_i (j) \leftarrow R_i(j)) -\Sigma_{q=1}^{r} f_{k\ j} (sI_q , j) \bullet \alpha_{iq} , I_q = \phi_{k(i)prp} (q)$.

4) Then the user verifies whether the received values remain a valid code word determined by secret matrix P:

$(R_i(1) , \ldots , R_i(m)).P = (R_i(m+1) , \ldots , R_i(n))$.

Because all the servers operate over the same subset of indices, the linear aggregation of these r specified rows(Ri1, . . . , Ri(n)) has to be a codeword in the encoded file matrix.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 Ri(j) is compute exactly in the same way as token vi(j) ,thus the user can simply find which server is misbehaving.[10]

Algorithm 2 Correctness Verification and Error Localiza- tion

```
1: procedure CHA L L E NG E (i)
2:   Recompute $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}$ from KPRP
3:   Send { $\alpha_i, k^{(i)}_{prp}$ } to all the cloud server
4:   Receive from servers:
     $\{R_i^{(j)} = \Sigma_{q=1} \alpha_i * G^{(j)} [\varphi (i) (q)] | 1 \le j \le n\}$
5:   for (j ← m + 1, n) do
6:     $R^{(j)} \leftarrow R^{(j)} - \Sigma_{q=1} f_{k_j}(s_{I_q,j}) \cdot \alpha_i , I_q = \varphi_{k_{prp}}(i) (q)$
7:   end for
8:   if (( Ri ( 1 ) ,…, Ri ( m ) ) • P = = ( Ri ( m+1 ) , . . ., Ri ( n ) )) then
9:     Accept and ready for the next challenge.
10:  else
11:    for ( j ← 1, n) do
12:      if (Ri ( j ) ! = vi( j ) then
13:        return server j is misbehaving.
14:      end if
15:    end for
16:  end if
17: end procedure
```

REFERENCES

1). C. Wang, Q. Wang, K.Ren, and W. Lou, “Ensuring data storage security in cloud computing,” in Proc. of IWQoS‟ 09, July 2009

2). Amazon.com, “Amazon web services (aws),” Online at http://aws.amazon.com/, 2009.

3). 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.

4). K. D. Bowers, A. Juels, and A. Oprea, “Proofs of retrievability: Theory and implementation,” in *Proc. of ACM workshop on Cloud Computing security (CCSW’09)*, 2009, pp. 43–54.

5). Q. Wang, K. Ren, W. Lou, and Y. Zhang, “Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance,” Proc. of IEEE INFOCOM, 2009.

6). G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, “Scalable and Efficient Provable Data Possession,” Proc. of SecureComm ‟ 08, pp. 1– 10, 2008.

7). Q. Wang, K. Ren, W. Lou, and Y. Zhang, “Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance,” Proc. of IEEE INFOCOM, 2009.

8). K. D. Bowers, A. Juels, and A. Oprea, “HAIL: A High-Availability and Integrity Layer for Cloud Storage,” Cryptology ePrint Archive, Report 2008/489, 2008, http://eprint.iacr.org/.

9). L. Carter and M. Wegman, “Universal Hash Functions,” Journal of Computer and System Sciences, vol. 18, no. 2, pp. 143–154, 1979.

10). Q. Wang, K. Ren, W. Lou, and Y. Zhang, “Dependable and secure sensor data storage with dynamic integrity assurance,” in Proc. Of IEEE INFOCOM‟09, Rio de Janeiro, Brazil, Appril 2009