International Journal of Engineering Research & Management Technology ISSN: 2348-4039

Email: editor@ijermt.org November- 2014 Volume 1, Issue-6 www.ijermt.org

Secure Data Storage in Cloud Computing

Chiranjit Dutta, Ranjeet Singh
Faculty of Information Technology
SRM University
NCR Campus Ghaziabad

1. ABSTRACT

Cloud computing is seen as an architecture of next generation IT enterprise. Cloud computing stores and moves the application software and database to the data centers, unlike traditional IT solutions, where all the services provided by them are under physical, logical and personal controls, due to which the data storage may not be fully trustworthy. So here we focus on securing the data storage in cloud computing, which always has been an important aspect for quality of services. To do so we propose an effecting scheme with two features:

- Using the homomorphism token with distributed Verification of erasure correcting code data, our scheme achieves integration of storage correctness insurance and data error localization ie. Identification of misbehaving servers.
- It support secure and effective and dynamic operation on data blocks like update , delete and append.

Extensive security analysis shows our scheme is highly efficient and resilient against Byzantine Failure, modification attack and malicious data

.

Keywords – Cloud Computing, SaaS, PaaS, IaaS, Amazon EC2

2. INTRODUCTION

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 centers into pools of computing 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 centers.

Moving data into the cloud offers great convenience to users since they don't have to care about the complexities of direct hardware management. The pioneer of Cloud Com- puting vendors, Amazon Simple Storage Service (S3) and Amazon Elastic Compute Cloud (EC2) [1] are both well known examples. 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 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

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 for number of reasons. 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. 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.

2.1 Design Goals

To ensure the security and dependability for cloud data storage under the aforementioned adversary model, we aim 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 mal-functioning 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.

2.2 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(2^p) 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.
- $f_{key}(\cdot)$ - pseudorandom function (PRF), which is defined
- as $f : \{0,1\}^* \times key \rightarrow GF(2^p)$.
- ver - a version number bound with the index for individual blocks, which records the times the block has been modified. Initially we assume version is 0 for all data blocks.
- s j - the seed for PRF, which depends on the file name, block index I the server position j as well as the optional block version number ver.

3. ENSURING CLOUD DATA STORAGE

In cloud data storage system users stores their data in the cloud and no longer possess the data locally. The correctness and availability of the data files being stored on the distributed cloud servers must be guaranteed. One of the key issues is to effectively detect any unauthorized data modification and corruption possibly due to server compromise Possibly due to server compromise and/or random Byzantine failures. Besides, in the distributed case when such inconsistencies are successfully detected, to find which server the data error lies in is also of great significance, since it can be the first step to fast recover the storage errors. To address these problems, 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 homomorphism token is introduced. The token computation function we are considering belongs to a family of universal hash function chosen to pre-serve the homomorphism properties, which can be perfectly integrated with the verification of erasure-coded data subsequently; it is also 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 outlined.

3.1 File Distribution Preparation

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. A (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. i.e., the unmodified m data file vectors together with k parity vectors is distributed across m+k different servers. The procedure or Algorithms for the process of the ensuring the data storage security in cloud computing are as:

Algorithm 1 Token Pre-computation

1. Procedure
2.Choose Parameters l,n and function f, and φ;
3.Choose number t of tokens;
4.Choose number r of indices per verification;
5.Generate master key key K_{prp} and challenge $k_{chal;}$
6. for vector $G^{(j)}$, j ← 1, n do;
7. for round i← 1, t do **;**
8. Compute $v^{ij)} = \sum q=1\ \alpha i * G(j)[\varphi m\ K(q)]P\ \ k^{p\ rp.}$
9. end for
10. end for
11. Stores all the v_is locally.
12**.** end procedure

3.2 Challenge Token Precomputation

In order to achieve assurance of data storage correctness and data error localization simultaneously, our scheme entirely relies on the pre-computed verification tokens. Before file distribution the user pre-computes a certain number of short verification tokens on individual vector $G^{(j)}$ (j ∈ {1, . . . , n}), each token covering a random subset of data blocks. Upon receiving challenge, each cloud server computes a short "signature" over the specified blocks and returns them to the user..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∈{1,...,n}), using a PRF f(·), a PRP φ(·), a challenge key k_{chal} and a master permutation key $K_{P\ RP}$. To generate the i^{th} token for server j, the user acts as follows:

1. Derive a random challenge value α_i of GF(2^p) by $\alpha_i = f_k\ \mathbf{chal}$ (i) and a permutation key $k^{pr)p}$ based on $K_{P\ RP}$
2. Compute the set of r randomly-chosen indices:
   $\{I_q \in [1, ..., l] | 1 \le q \le r\}$, where $I_q = \varphi_k$ (i)(q).

3. Calculate the token as:
   $v^{ij)} = \alpha^i * G^{(j)}[I_q]$, where $G^{(j)}[I_q] = g^{Ij)}q$

Algorithm 2 Correctness Verification and Error Localization

1.Procedure
2.. Recompute $\alpha_i = f_k\mathbf{chal}$ (i) and $k^{pr)p}$ from K_P
3. Send $\{\alpha_i, k^{pr)p}\}$ to all the cloud servers;
4. $\{R^{ij)} = \sum q=1\ \alpha iv * Gr\ \ (j)[\varphi_k(i)prp$

(q)]|1 ≤ j ≤ n}
5. **for** R(j) ← R(j) −∑**do** q=1 fkj (sIq ,j)·αi ,
Iq = φk
6. end for
7.if((R^i1), . . . , R^im)). P== (R^im+1), . ., R^in)) then
8. Acceptr and ready for the next challenge.
9. else
10. for (j ← 1, n) do
11. if (R^ij)! =v^ij)) then
12. return server j is misbehaving
13 end if
14. end for
15. end if
16.end procedure

3.3 Correctness Verification and Error Localization

Error localization is a key prerequisite for eliminating errors in storage systems. However, many previous schemes do not explicitly consider the problem of data error localization. Our scheme outperforms those by integrating the correctness verification and error localization in our challenge-response not only determines the correctness of the distributed storage, but also contains information to locate potential data error(s).
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 α_i as well as the i-th permutation key $k^{pr)p}$ to each servers.
2.The server storing vector $G^{(j)}$ aggregates those r rows specified by index $k^{pr)p}$ into a linear combination.

$$R^{ij)} = \sum_{q=1} \alpha^i * G^{(j)}[\varphi_k(i)\ (q)].$$

3. Upon receiving $R^{ij)}$s from all the servers, the user takes way blind values in $R^{(j)}$ (j ∈ {m + 1, . . . , n}) by $R_i^{(j)} \leftarrow R_i^{(j)} - \sum^{r}_{Q=1} fkj\ (sIq,j) \cdot \_qi$,where Iq = _k(i)prp(q).

4.Then the user verifies whether the received values remain a valid codeword determined by secret matrix P:
$(R^{i1)},\ldots,R^{im)}) \cdot P =^{?} (R^{im+1)}, \ldots, R^{in)})$.

3.4 File Retrieval and Error Recovery

Since our layout of file matrix is systematic, 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. However by choosing system parameters (e.g., r, l, t) appropriately and conducting enough times of verification, we can guarantee the successful file retrieval with high probability. 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 3, 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.

4. Providing Dynamic Data Support Operation

So far, we assumed that F represents static

Algorithm 3 Error Recovery

1. Procedure
   % Assume the block corruptions have been
   detected among
   % the specified r rows;
   % Assume s ≤ k servers have been identified
   misbehaving
2. Download r rows of blocks from servers ;
3. Treat S servers as erasure and recover blocks.
4. Resend the recovered blocks to corresponding
   Servers.
5. end procedure

Archived data this model may fit some application scenarios, such as libraries and scientific datasets. 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 this section, we will show how our scheme can explicitly and efficiently handle dynamic data operations for cloud data storage.

A. Update Operation

In cloud data storage, sometimes the user may need to modify some data block(s) stored in the cloud, from its current value fij to a new one, fij + fij . We refer this operation as data update. Due to the linear property of Reed-Solomon code, a user can perform the update operation and generate the updated parity blocks by using fij only, without involving any other unchanged blocks. Specifically, the user can construct a general update matrix for all the unused tokens, the user needs to exclude every occurrence of the old data block and replace it with the new one.

B. Delete Operation

Sometimes, after being stored in the cloud, certain data blocks may need to be deleted. The delete operation we are considering is a general one, in which user replaces the data block with zero or some special reserved data symbol. From this point of view, the delete operation is actually a special case of the data update operation, where the original data blocks can be replaced with zeros or some predetermined special blocks. Therefore, we can rely on the update procedure to support delete operation, i.e., by setting fij in F to be −fij . Also, all the affected tokens have to be modified and the updated parity information has to be blinded using the same method specified in update operation.

C. Append Operation

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. We anticipate that the most frequent append operation in cloud data storage is bulk Append, in which the user needs to upload a large number of blocks (not a single block) at one time. To support block append operation, we need a slight modification to our token pre-computation. Specifically, we require the user to expect the maximum size in blocks.

D. Insert Operation

An insert operation to the data file refers to an append operation at the desired index position while maintaining the same data block structure for the whole data file, i.e., inserting a block F[j] corresponds to shifting all blocks starting with index j + 1 by one slot. An insert operation may affect many rows in the logical data file matrix F, and a substantial number of computations are required to renumber all the subsequent blocks as well as re-compute the challenge-response tokens. Therefore, an efficient insert operation is difficult to support and thus we leave it for our future work.

5 FUTURE SCOPES

With arrival of cloud computing the conventional way of computing has gone for a sea change. And this new addition in the computing is not a flash in the pan as it is going to rule the roost in the future. As per some expert opinions, it is going to be the face of future cloud computing. And hence, the future of cloud computing seems very promising.

1. **Presence of Internet will boost its future**: The cloud computing will becomes all the more important with the omnipresence of high-speed, broadband Internet. Slowly but steadily we are getting closer. Even airlines are offering satellite based wi-fi services in flights. In a mass drive to connect every village with Internet wireless Internet services are offered through the help of satellite, although speed is a bit slow.
2. **No more software updates:** Most of the computer professionals loose lots of their time and efforts downloading different versions of software so that they can access the various programs and data with little efforts. Most of the softwares are on the cloud servers so you don't need to down and install for little use. So, whether you want to access emails or go through spreadsheet, it has become fun with the arrival of cloud computing.
3. **Hardware optional:** With the arrival of cloud computing it is no longer necessary to purchase hard drives with large storage capacity, as it can be stored on cloud. So keep the fear of loosing your data away. All your data with complete back up can be stored on the cloud.

6. CONCLUSION

In this paper, we investigated the problem of data security in cloud data storage, which is essentially a distributed storage system. To ensure the correctness of users' data in cloud data storage, we proposed an effective and flexible

distributed scheme with explicit dynamic data support, including block update, delete, and append Through detailed security and performance analysis, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colliding attacks.

REFERENCE

1. Amazon.com "amazon web services(AWS)", 2008.
2. N.Gohring, Amazon's "S3 down for several hours",online at http:\\www.pcworld.com , 2008
3. K.D Bowers, A.Juels and A.Oprea, HAIL:"A high- availability and Integrity Layer for cloud data storage"
4. G. Ateniese , R.D Burns, R curtmola, J.herring "provable data possession at untrusted stores", 2007
5. J.S Plank and Y.Ding, "Note: correction to the 1997 tutorial on Reed- Solomon Coding , 2003
6. Q.Wang, K.Ren, W.Lou and Y.Zhang "Dependable and secure sensor data storage with dynamic integrity assurance", 2009.
7. Asprise.com
8. Google.com
9. M.A Shah , M.Baker , J.C. Bogul and R.Swaminathan "Auditing to keep online storage Service honest".
10. M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard "A Cooperative internet backup schemes"
11. G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik"Proc of the secure Comm"08.
12. T.S.J Schwarz and E.L Miller"Store, Forget and Check using allgebric signatures to check remotely tely administered storage".
13. L.Carter and M. Wegman"Universal Hash Functions"
14. R.Curtmola,O.Khan"MR-PDP Multiple Replica Provable Data Possession"