



Enhanced Dynamic Approach to Data Storage Security in Cloud

Pragati Mishra, Aishwarya Srivastava, Aditi Jain
CSE Department, GCET, Greater Noida,
Uttar Pradesh, India

Abstract— Cloud computing offers infrastructure and various computational services on demand for various customers on a pool of shared resources. Cloud users store their data in the cloud environment on individual cloud servers which is privileged by the service provider. Secure outsourcing of computational services to an untrusted (cloud) service provider is thus, becoming more and more important. This article caters to the need of implementation of security while storing user data on the cloud servers. The traditional techniques of implementing pure cryptographic solutions based on fully homomorphic encryption are promising but they suffer from very high latency. Here, we propose a scheme that reduces the latency of computation by utilizing the homomorphic token with Reed-Solomon erasure correcting code. Thus, by utilizing the concept of homomorphic token, our proposed scheme ensures the containment of faults such that identification of faulty servers becomes efficient and it also verifies the correctness of the data stated on each of the individual cloud servers. The proposed scheme is highly resilient against Byzantine failure and malicious data modification attack. It also supports the application of secure dynamic operations on data blocks. Considerable testing indicates that our proposed scheme is highly cost-effective and provides an effective barrier against various attacks aimed to compromise the cloud servers.

Keywords— Cloud computing, secure outsourcing, latency, byzantine failure, cloud servers.

I. INTRODUCTION

The services offered by cloud can be categorized into three categories that are broadly defined as:

1.1 IAAS (Infrastructure as a Service)

It is the most basic service model. Here, hardware is offered as a service to the organization such as server space. For example: GoogleApps, Salesforce.com.

1.2 SAAS (Software as a Service)

Here, various services are provided to end users from cloud provider through the internet. Customer uses the software on a pay-per use basis. For example: Google AppEngine, Microsoft Azure.

1.3 PAAS (Platform as a Service)

Here, a computing platform is delivered by the cloud service providers to the customers. For example: Amazon EC2, Simple Storage service (S3).

Cloud computing is the technology in which data and programs can be stored centrally and accessed anytime from anywhere through clients. The fact that cloud computing puts data outside the control of the data owner introduces various security threats. This creates the need of secure outsourcing of computation to cloud service provider. Many solutions have been given for handling the security issues in cloud computing. The cryptographic solutions based purely on homomorphic encryption are helpful but suffer from the problem of high latency that is the time delay between a client request and the cloud service provider's response. Since in a cloud environment the workload is larger and less predictable, the issue of latency cannot be ignored. Another security issue involves that the customers should not have any access to other customer's data or network traffic, etc. Thus a system is needed to address this problem by providing strong isolation, storage and network monitoring properties. It is essential to ensure the correctness of data since due to data loss, deletion or modification of data by unauthorized users, the threat of data compromise is increasing day by day. Insufficient authentication, authorization, inconsistent use of encryption and software keys, data center reliability etc are some examples of risks involved in data loss.

In this paper, we propose an effective and efficient scheme for providing advanced security in data storage in cloud. Our proposal can be summarized as follows:

- 1) Compared to traditional applications, we focus on reducing the latency of computation by combining the homomorphic encryption with the token implementation on data where the computation leaks no information and can be verified.
- 2) Error localization is achieved by using trail-reply protocol.
- 3) Proposed scheme supports dynamic operation on data such as insertion, updation, deletion, etc.
- 4) Byzantine failures, malicious data modification and data loss is also taken care of in our work.

II. PROBLEM STATEMENT

A. System Architecture Model:

Three distinct entities that can be identified in the system proposed in this paper are as follows:

- Users: Users may comprise of an individual or an organization who stores data in the cloud and rely on cloud for performing various operations on it.
- Cloud Service provider (C.S.P): CSP maintains and owns cloud servers. Cloud servers contain the resources that may be required by customers for cloud computing.
- Optional mediator: An entity which acts as a link between conflicting parties and make them come to an agreement. It has capabilities to assess and reveal risk of cloud storage services on behalf of the users.

B. Challenger model

There are two sources in which the cloud servers face security intimidations. On the one hand ,a CSP can be malicious and untrusted . It may move data to a lower tier of storage than agreed or may also hide a data loss incident. On the other hand, some economically motivated Challengers may also exist in the system. These Challengers may have the capability to compromise cloud data storage servers. On the basis of capability, there exist two types of Challengers:

- Strong Challenger: This is the worst case scenario in which we assume that the challenger has the capability to compromise all the data storage servers in cloud. In this case, all the servers collude together to hide a data loss or corruption incident.
- Weak Challenger: This Challenger corrupts the user data stored on different individual servers. It may introduce its own fraudulent data to corrupt the original data.

C. Design Goals:

Under the aforementioned Challenger model, we aim to design the efficient mechanisms for ensuring Cloud data storage. Following are the goals achieved in our proposed system:

- Accuracy of Stored Data: to ensure user that the data provided to them is appropriate, undamaged and kept intact all the time in Cloud.
- Fast Data Error Localization: To effectively locate the mal-functioning server as soon as data corruption has been detected.
- Data Dependability: To enhance the availability of data against malicious data modification .It minimizes the effect brought by data errors due to Byzantine failure, server colluding attack.
- Minimum Latency: To enable users to perform checks on storage correctness with mini mum expenses that is to reduce the time delay between requests and their responses.

D. Notation and Preliminaries:

- F = data to be stored in the form of matrix with m -sized data vectors, consisting of '1' blocks.
- A =Dispersal Matrix
- V = Encoded File Matrix with ' $n=m+k$ ' vectors with '1' block
- $fkey(.)$ =pseudo random function(PRF) such as
- $f : \{0, 1\} * key \rightarrow GF(2^p)$
- $\emptyset key ()$ = pseudo random permutation (PRP) such as
- $\emptyset : \{0, 1\} \log_2 l * key \rightarrow \{0, 1\} \log_2 l$
- k =No. Of times user challenges cloud
- P = Parity generation matrix of size $m*k$.

III. PROPOSED SYSTEM

In cloud data storage systems, the data is stored in cloud so users do not possess the data locally. Thus, it is important to guarantee the correctness and availability of user's data. In order to address the problems of data storage, our main scheme for data storage is given in this section. Fig 1 shows the descriptive architecture of the proposed system.

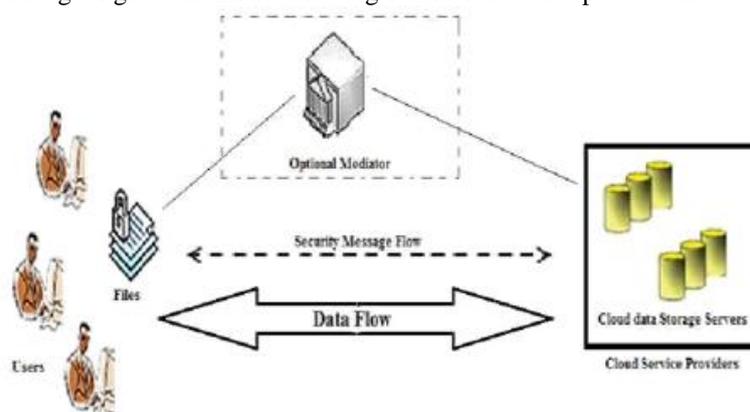


Fig. 1 System Architecture Model

A. File distribution

The original data file F_l is distributed redundantly across a set of $n=m+k$ servers. A $(m+k, k)$ Reed Solomon erasure correcting code is used to create k parity vectors from the ' m ' vectors so that m vectors can be constructed from any ' m ' out of ' $m+k$ ' vectors.

As they are kept on individual servers, they can survive fault of any ' k ' of ' $m+k$ ' servers without any loss of data.

Let $F_l = (F_1, \dots, F_m)$ and

$$F_i = (f_{1i}, f_{2i}, \dots, f_{li})^T$$

$$i \rightarrow \{1, \dots, m\}, l \leq (2^p - 1);$$

such that the elemental block units are elements of Galois Field (2^p) .

The parity vectors are maintained systematically in the form of dispersal matrix A , so that ' m ' out of ' $m+k$ ' columns are invertible.

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

Where I is an $(m \times m)$ identity matrix and P is a parity generation matrix of size $(m \times k)$

Encoded File Generation

$$V = F_l \cdot A = (V^{(1)}, V^{(2)}, \dots, V^{(n)})$$

$$= (F_1, F_2, \dots, F_m)$$

Where $V^{(j)} = (v_1^{(j)}, v_2^{(j)}, \dots, v_l^{(j)})^T; j \in \{1 \dots n\}$

This procedure to generate the encoded file V minimizes the original file vectors of F_l .

Algorithm 1 Trial Token Pre-computation

User pre-computes verification tokens for individual vectors $V^j; j \in \{1 \dots n\}$, distributes the file on individual servers, and challenges the cloud servers periodically based on the trial request.

Each token is associated with a randomized subset ' r ' of data blocks.

For ensuring the overall correctness of blocks of data, user makes trials to the cloud server. The trials are nothing but challenges to the cloud server in order to ensure correctness of our data. When the trial is received by the cloud; the cloud generates the 'identity' for the specified block and returns it as a reply to user. If the identity matches the token, there is no fault of data integrity at that server. Also, all the other servers work upon the same set of indexes and the reply for integrity check should be a plausible codeword determined by secret matrix P .

If the user wants to challenge cloud servers ' k ' times, then ' k ' verification tokens must be computed for each $v[j]$, using challenge key (k_{chal}), master permutation key (k_{master}), PRF $f(\cdot)$ and PRP $\emptyset(\cdot)$.

To generate the i^{th} token for server j user follows the following steps:-

Method

1. Based on k_{master} , derive the $k_{prp} [i]$ and
2. Derive the challenge value A_i of GF (2^p) by $(A_i = f_{kchal}(i))$
3. Calculate the set of r indices (random)
 $\{I_q \in [1, \dots, l] | [1 \leq q \leq r]\}$
 $I_q = \emptyset_{kprp(i)}(q)$
4. Compute token as
 $t_i^{(j)} = \sum_{q=1}^r A_i^q * V^{(j)}[I_q]$ where $V^{(j)}[I_q] = v_{I_q}^{(j)}$
5. If (operations require very less latency) Token stored locally
 Else

Token stored on cloud servers after encryption

{

The token $t_i^{(j)}$ is an element of Galois Field (2^p) i.e. $G.F(2^p)$, is the reply user expects the server j would send to it after the user challenges the server for a specified data block.

Here we store the token using encryption.

Bind it to the each parity vector $v_i^{(j)}$ in $(V^{(m+1)}, \dots, V^{(n)})$, given by $(v_i^{(j)} = v_i^{(j)} + f_{k_j}(s_{ij}))$. The user then distributes the file to different servers S_1, S_2, \dots, S_n .

}

Algorithm 2 Correctness Verification and Error Localization

- 1: **Method** TRIAL (i)
- 2: Recompute $A_j = f_{kchal}(i)$ and $k_{prp}^{(i)}$ from K_{master} ;
- 3: Send $\{A_i, k_{prp}^{(i)}\}$ to all the cloud servers;
- 4: Receive identity reply from servers.
 $\{L_i^{(j)} = \sum_{q=1}^r A_i^q * V^j[\emptyset_{kprp}^{(i)}(q)] | 1 \leq j \leq n\}$
- 5: **for loop 1** ($j \leftarrow m+1, n$)
- 6: $L^j \leftarrow L^j - \sum_{q=1}^r \int k_j(s_{iq,j}) \cdot A_i^q, I_q = \emptyset_{kprp}^{(i)}(q)$
- 7: **end for loop 1**
- 8: **if** (1) $(L_i^{(1)}, \dots, L_i^{(m)}) \cdot P = (L_i^{(m+1)}, \dots, L_i^{(n)})$
- 9: Agree and prepare for the next trial.
- 10: **Else**

```
11:   for loop 2 (j ← 1, n)
12:       if (2) ( $L_i^{(j)} \neq t_i^{(j)}$ ) then
13:           return server j is at fault
14:       end if(1)
15:   end for loop(2)
16:   end if condition(2)
17: end method
```

B. File Reconstruction and recovery

As the file is maintained in matrix format, the user can reconstruct the original one by using data vectors. Whenever data corruption is detected, the pre-computed tokens and the received identity response comparison can identify the faulty servers.

IV. DYNAMIC OPERATIONS ON DATA

In this segment, we will show how our scheme can efficiently handle dynamic operations on data for data storage on cloud.

A. Update Operation

In cloud data storage, sometimes the user may need to modify some data blocks stored in the cloud, from its current value to a new one. This operation is referred to as data update. Due to the linear property of Reed-Solomon code, user can perform the update operation and generate the updated parity blocks, without involving any other unchanged blocks. The user can also generate the update information for both the data vectors and parity vectors.

B. Delete Operation

Sometimes, certain data blocks may need to be deleted. User can replace the data block with zero or some special reserved data symbol. Thus, the delete operation is actually a special case of the data 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 at one time secret matrix P, the user can directly calculate the append blocks for each parity server.

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 corresponds to shifting all blocks by one slot. An insert operation may affect many rows in the logical data file matrix, and 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.

V. EXPERIMENT RESULTS

In this section, we have analyzed our proposed scheme in terms of security and efficiency.

A. Strength against Weak Challenger

- High detection probability against modification of data: Since servers are required to operate on selected rows, computational overhead is really reduced.
- High Probability of Error Localization: A matching check is performed where it is checked whether the rows chosen by user matches the rows modified by the Challenger and then matching probability is calculated that one of the rows selected by user matches one of the rows modified by challenger. The probability of identifying misbehaving servers is as follows:
 $P(\text{Misbehaving Servers}) = P(\text{matching probability}) * P(\text{false negative result for complementary event})$

B. Strength against Strong Challenger

If the parity vectors F.P are dispersed directly, it will become very easier for a challenger to identify the P matrix and then recover the Data Files. So, in order to ensure the storage correctness of data, we have blinded the parity blocks using random permutations. Keyed pseudorandom function fk_j

(•) with key k_j , introduced previously are used for this purpose. Thus, the malicious servers are not able to recover the secret matrix P.

C. Performance Evaluation

- File Distribution: As the number of data servers increases, the length of data vectors on each server will decrease. Therefore only few calls are made to the encoder. This results in decrease in cost of parity generation. It is seen that the performance of our proposed scheme is comparable to that of [17]. Our scheme supports dynamic operations on data whereas [17] is for static data only.

- Trial Token Pre-computation: The process adopted for the token computation is faster than hash function based token precomputation. Also extra storage for token is very less and thus can be neglected.

VI. CONCLUSIONS

Then this paper, we explored the problem of data storage security.

In the distributed environment of the cloud system. To ensure the integrity of the users' data in cloud storage environment, we proposed a potent and flexible distributed strategy with explicit dynamic data support, including update, delete, and append operations on the data block. We rely on erasure-correcting code in the file distribution in order to provide redundancy parity vectors and guarantee the data dependability.

By exploiting the concept of homomorphism token with distributed verification of erasure-coded data, our proposed scheme achieves integrity of stored data and error localization in the data, i.e., whenever data corruption is detected during the process of verification of storage correctness across the distributed servers, the simultaneous identification of the misbehaving servers can be assured.

Security and performance analysis exhibit that our scheme is highly resilient to Byzantine failure, malicious data alteration attack, and even server colluding attacks. We deduce that data storage security in Cloud Computing is an area full of disputes and of paramount importance. It is still in its initial stages now and many research problems are yet to be recognized.

REFERENCES

- [1] F. Zhang, J. Wang, K. Sun, A. Stavrou, HyperCheck: a hardware-assisted integrity monitor, *IEEE Trans. Dependable Sec. Comput.* (2013);<http://dx.doi.org/10.1109/TDSC.2013.53>.M. Ali et al. / *Information Sciences* 305 (2015) 357–383383.
- [2] D. Zissis and D. Lekkas, “Addressing Cloud Computing Security Issues,” *Future Generation Computer Systems*, Vol. 28, No. 3, 2012, pp. <http://www.sciencedirect.com/science/article/pii/S0167739X10002554>
- [3] F. Zhang, J. Chen, H. Chen, B. Zang, Cloudvisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization, in: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, 2011, pp. 203–216.
- [4] Monitoring Editor: Gunther Eysenbach, Reviewed by Peter Mork, Eizen Kimura, Carl Reynolds, and Feipei Lai (2011); <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3222190/>
- [5] James Walden, Northern Kentucky University, The OWASP Foundation, <http://www.owasp.org>, “Cloud Computing Security”, February 22nd, 2011.
- [6] Q. Wang, C. Wang, K. Ren W. Lou, and J. Li, “Enabling public verifiability and data dynamics for storage security in cloud computing,” *IEEE Trans. Parallel and Distributed Computing*.VOL.22, NO.5, May 2011, pp.847-859.
- [7] S. Ramgovind, M. Elo and E. Smith, “The Management of Security in Cloud Computing,” *Information Security for South Africa*, Sandton, 2-4 August 2010, pp. 1-7.
- [8] Cloud Security Alliance(CSA), “Top Threats to Cloud Computing V1.0”, March 2010, <http://www.cloudsecurityalliance.org/topthreats>.
- [9] A. F. Barsoum and M. A. Hasan, “Provable possession and replication of data over cloud servers,” *Centre For Applied Cryptographic Research (CACR), University of Waterloo, Report 2010/32, 2010*, <http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-32.pdf>.
- [10] *Journal of Theoretical and Applied Information Technology*, “CLOUD COMPUTING”, www.jatit.org, 2005 – 2009.
- [11] Q. Wang, K. Ren, W. Lou, and Y. Zhang, “Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance,” *Proc. of IEEE INFOCOM*, 2009.
- [12] Avery P. *IT Business Edge*. 2009. Aug 26. *webcite Research Indicates Increase in Cloud Computing*; <http://www.itbusinessedge.com/cm/community/kn/blog/research-indicates-increase-in-cloud-computing/?cs=35256>.
- [13] M.A. Shah, R. Swaminathan, and M. Baker, “Privacy-Preserving Audit and Extraction of Digital Contents,” *Cryptology ePrint Archive*, Report 2008/186, <http://eprint.iacr.org>, 2008.
- [14] R. Curtmola, O. Khan, and R. Burns. “Robust remote data checking”. In: *Proc. of StorageSS '08*, 2008, pp.63-68, Virginia, USA.
- [15] *Business Wire The Free Library*. 2008. *webcite DiskAgent Launches New Remote Backup and Loss Protection Software as a Service Offering* [http://www.thefreelibrary.com/DiskAgent\(TM\) Launches New Remote Backup and Loss Protection Software...-a0182194404](http://www.thefreelibrary.com/DiskAgent(TM) Launches New Remote Backup and Loss Protection Software...-a0182194404).
- [16] 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>