ISSN:2348-2079

Volume-5 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

Coupling and cohesion enabled information transaction in cloud computing

[1]Mrs. K.E.Eswari M.C.A., M.Phil., M.E., Associate Professor,
[2]R.Malliga, Final MCA,
Department of M.C.A, Nandha Engineering College (Autonomous), Erode-52.
E-Mail ID: eswari.eswaramoorthi@nandhaengg.org, malligaranganathanr@gmail.com

Abstract - Cloud computing delivers convenient, on-demand access to shared pools of data, applications and hardware over the internet. Cloud computing provides unlimited infrastructure to store and execute customer data and program. Due to this redundancy the data can be easily modified by unauthorized users which can be stored in the database. This leads to loss of data privacy and security to database. Extensive security and performance analysis shows that the proposed scheme ensures that cyclic redundancy check and time-tested practices and technologies for managing trust relationships in traditional enterprise IT environments can be extended to work effectively in both private and public clouds. Those practices include data encryption, strong authentication and fraud detection, etc. Property theft is prevented due to strong authentication mechanism, which includes stronger fraud detection methods.

I 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 service on a huge scale. As a result, users are at the mercy of their cloud service providers for the availability and integrity of their data. On the one hand, although the cloud infrastructures are much more powerful and reliable than personal computing devices, broad range of both internal and external threats for data integrity still exist. Examples of outages and data loss incidents of noteworthy cloud storage services appear from time to time.

On the other hand, since users may not retain a local copy of outsourced data, there exist various incentives for Cloud Service Providers (CSP) to behave unfaithfully towards the cloud users regarding the status of their outsourced data. For example, to increase the profit margin by reducing cost, it is possible for CSP to discard rarely accessed data without being detected in a timely fashion. Similarly, CSP may even attempt to hide data loss incidents so as to maintain a reputation.

In order to achieve the assurances of cloud data integrity and availability and enforce the quality of cloud storage service, efficient methods that enable on-demand data correctness verification on behalf of cloud users have to be designed. However, the In this paper, we propose an effective and flexible distributed storage verification scheme with explicit dynamic data support to ensure the correctness and avail-ability of users' data in the cloud. We rely on erasure-correcting code in the file distribution preparation to pro-vide redundancies and guarantee the data dependability against Byzantine servers, where a storage server may fail in arbitrary ways. 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 of data errors, i.e., the identification of the misbehaving server.

Recently, the importance of ensuring the remote data integrity has been highlighted by the following research works under different system and security models. These techniques, while can be useful to ensure the storage correctness without having users possessing local data, are all focusing on single server scenario. They may be useful for quality of

service testing, but does not guarantee the data availability in case of server failures. We also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third-party auditors and are worry-free to use the cloud storage services. Our work is among the first few ones in this field to consider distributed data storage security in Cloud Computing. Our contribution can be summarized as the following three aspects:

1) Compared to many of its predecessors, which only provide binary results about the storage status across the distributed servers, the proposed scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server.

2) Unlike most prior works for ensuring remote data integrity, the new scheme further supports secure and efficient dynamic operations on data blocks, including: update, delete and append.

3) The experiment results demonstrate the proposed scheme is highly efficient. Extensive security analysis shows our scheme is resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

II PROBLEM STATEMENT

A System Model

Representative network architecture for cloud storage service architecture is illustrated in Figure 1. Three different network entities can be identified as follows:

- User: an entity, who has data to be stored in the cloud and relies on the cloud for data storage and computation, can be either enterprise or individual customers.

- Cloud Server (CS): an entity, which is managed by **cloud service provider** (CSP) to provide data storage service and has significant storage space and computation resources.

- Third Party Auditor (TPA): an optional TPA, who has expertise and capabilities that users may not have, is trusted to assess and expose risk of cloud storage services on behalf of the users upon request.

In cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, cooperated and distributed manner. Data redundancy can be employed with technique of erasure-correcting code to further tolerate faults or server crash as user's data grows in size and importance.

As users no longer possess their data locally, it is of critical importance to ensure users that their data are being correctly stored and maintained. In case that user do not necessarily have the time, feasibility or resources to monitor their data online, they can delegate the data auditing tasks to an optional trusted TPA of their respective choices.

In our model, we assume that the point-to-point communication channels between each cloud server and the user is authenticated and reliable, which can be achieved in practice with little overhead. These authentication handshakes are omitted in the following presentation.

B Adversary Model

From user's perspective, the adversary model has to capture all kinds of threats towards his cloud data integrity. Because cloud data do not reside at user's local site but at CSP's address domain, these threats can come from two different sources: internal and external attacks.

Therefore, we consider the adversary in our model has the following capabilities, which captures both external and internal threats towards the cloud data integrity. Specifically, the adversary is interested in continuously corrupting the user's data files stored on individual servers.

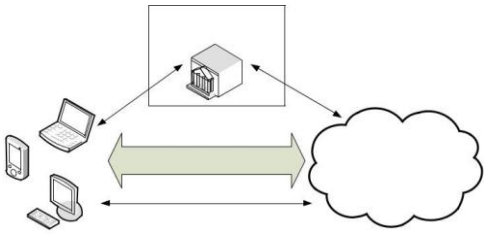

Fig. 1: Cloud storage service architecture

From being retrieved by the user. This corresponds to the threats from external attacks.

C 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: Storage correctness: to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud. Fast localization of data error: to effectively locate the malfunctioning server when data corruption has been detected.

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.

III 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. One of the key issues is to effectively detect any unauthorized data modification and corruption, possibly due to server compromise and/or random Byzantine failures.

A 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.

B 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.

To identify potential threats from external attacks. How-ever, many previous schemes, do not explicitly consider the problem of data error localization, thus only providing binary results for the storage verification.

C Correctness Verification and Error Localization

Error localization is a key prerequisite for eliminating errors in storage systems. It is also of critical importance within our error occurred in this verified application to correctness.The user sends the token, secret matrix P, permutation and challenge key. $K_{P\ RP}$ and k_{chal} to TPA for auditing delegation. The correctness validation and misbehaving server identification for TPA is just similar to the previous scheme. There is no way for TPA to learn the data content information during auditing process. Therefore, the privacy-preserving third party auditing is achieved. Note that compared to previous scheme, we only change the sequence of file encoding, token pre-computation, and blinding. Thus, the overall computation overhead and communication overhead remains roughly the same.

IV PROVIDING DYNAMIC DATA OPERATION SUPPORT

So far, we assumed that F represents static or archived data. This model may fit some application scenarios, such as libraries and scientific datasets. However, in cloud data storage, there are many potential scenarios where data stored in the cloud is dynamic, like electronic documents, photos, or log files etc. Therefore, it is crucial to consider the dynamic case, where a user may wish to perform various block-level operations of update, delete and append to modify the data file while maintaining the storage correctness assurance.

Since data do not reside at users' local site but at cloud service provider's address domain, supporting dynamic data operation can be quite challenging. On the one hand, CSP needs to process the data dynamics request without knowing the secret keying material. On the other hand, users need to ensure that the entire dynamic data operation request has been faithfully processed by CSP. To address this problem, we briefly explain our approach methodology here and provide the details later. For any data dynamic operation, the user must first generate the corresponding resulted file blocks and parities. This part of operation has to be carried out by the user, since only he knows the secret matrix P. Besides, to ensure the changes of data blocks correctly reflected in the cloud

address domain, the user also needs to modify the corresponding storage verification tokens to accommodate the changes on data blocks.

A Update Operation

In cloud data storage, a user may need to modify some data block(s) stored in the cloud, from its current value f_{ij} to a new one, $f_{ij} + f_{ij}$. We refer this operation as data update. Figure 2 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 using f_{ij} only, without involving any other unchanged blocks. Specifically, the user can construct a

Note that we use zero elements in F to denote the unchanged blocks and thus F should only be a sparse matrix most of the time (we assume for certain time epoch, the user only updates a relatively small part of file **F**). To maintain the corresponding parity vectors as well as be consistent with the original file layout, the user can multiply **F** by A and thus generate the update information for both the data vectors and parity vectors as follows:

$$F \cdot A = (\Delta G^{(1)}, \ldots, G^{(m)}, G^{(m+1)}, \ldots, G^{(n)}) = (\Delta F_1, \ldots, F_m, G^{(m+1)}, \ldots, G^{(n)}),$$

where $(j \in \{m + 1, \ldots, n\})$ denotes the update information for the parity vector $G^{(j)}$.

We use a new seed s^{ver}_{ij} for the PRF. The version number ver functions like a counter which helps the user to keep track of the blind information on the specific parity blocks. After blinding, the user sends update information to the cloud servers, which perform the update operation as $G^{(j)} \leftarrow G^{(j)} + G^{(j)}$, $(j \in \{1, \ldots, n\})$.

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.

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.

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. Thus, 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.

V CONCLUSION

In this paper, we investigate 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 effective and flexible distributed scheme with explicit 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 dependability.

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, pp.1-9.

[2]Amazon.com,"Amazon web services,"Online at http://aws.amazon.com/,2009.

[3] M.Arrington,"Gmail disaster: Reports of mess email deletions," Online at http://www.techcrush.com/2006/12/28/December/2009.

[4] J.Kincaid, "MediaMax/TheLinkup Closes Its Doors, Online at http://www.techcrush.com/2008/07/10/July 2008.

[5] A.Jules and J.Burton S.Kaliski,"Pors;Proofs of retriveability for large files," in Proc. Of CCS'07, Alexandria, v/a Octobar 2007.

[6] G.Atenies , R.D.Pietro, L.V.Mancini, and G.Tsudik,"Scalable and efficient provable data possession,"in Proc. Of SecureComm'08,2008.

[7] M.A.Shah, R.Swaminathan, and M.Baker,"Privacy-preserving audit and extraction of digital contents," Cryptography ePrint Archieve, Report 2008.

[8] M.A.Shah, M.Baker, J.C.Mogul, and R.Swaminathan,"Auditing to keep online storage services honest,"in Proc. OfHotOS'07 Berkely,CA,USA:2007.

[9] C.Erway, A.Kupcu, C.Paoamanthou, and R.Tamassia,"Dynamic provable data possession,"in Proc. Of CCS'09,2009

.

[10] K.D.Bowers, A.Juels, and A.Oprea,"Proofs of retrievability: Theory and implementation,"in Proc. Of ACM workshop on Cloud Computing security,2009.