© 2016 IJSRSET | Volume 2 | Issue 2 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Ensuring Data Storage Security in Cloud Computing Using Homomorphism Token

Loheswaran K, Deepan R
Department of Coputer Science and Technology, Sasurie College of Engineering, Vijayamangalam, Tamil Nadu

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. The security is an important aspect of quality of services. To ensures the correctness of user data in cloud. In the cloud many services are provided to the client by cloud. Data store is main future that cloud service provides to the companies to store huge amount of storage capacity. But still many companies are not ready to implement cloud computing technology due to lack of proper security control policy and weakness in protection which lead to many challenge in cloud computing. The main objectives of this paper are, 1) To prevent Data access from unauthorized access, it propose a distributed scheme to provide security of the data in cloud .This could be achieved by using homomorphism token with distributed verification of erasure-coded data. 2) Proposed scheme perfectly stores the data and identifies the any tamper at the cloud server. 3) And also performs some of the tasks like data updating, deleting, appending.

Keywords: Cloud Computing, Internet Computing, Homomorphism, Data Storage Security, TPA,CSP

I. INTRODUCTION

Cloud computing, to put it simply, means internet computing. The internet is commonly visualized as clouds; hence the term "cloud computing" for computation done through the internet. With cloud computing users can access database resources via the internet from anywhere, for as long as they need, without worrying about any maintenance or management of actual resources. Besides, databases in cloud are very dynamic and scalable. Cloud Computing is unlike grid computing, utility computing, or autonomic computing. In fact, it is a very independent platform in terms of computing. The best example of cloud computing is Google apps where any application can be accessed using a browser and it can be deployed on thousands of computer through the internet. It also provides facilities for users to develop, deploy and manage their applications on the cloud, which entails virtualization of resources that maintains and manages itself.

Our proposed scheme enables the data owner to delegate tasks of data file re-encryption and user secret key update to cloud servers without disclosing data contents or user access privilege information. We achieve this goal by exploiting and uniquely combining techniques and algorithms (Correctness Verification and Error Localization, traditional replication-based file distribution, adding random perturbations). 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 homomorphism 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(s).

Our work is among the first few ones in this field to consider distributed data storage 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 state across the distributed servers, the challenge-response protocol in our work further provides the localization of data error. 2) Unlike most prior works for ensuring remote data integrity, the new scheme supports secure and efficient dynamic operations on data blocks, including: update, delete and append. 3) Extensive security and performance analysis shows that the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

II. METHODS AND MATERIAL

Achieved Problem statement

A. System Model

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

- User: users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations.
- Cloud Service Provider (CSP): a CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems.
- 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. Thereafter, for application purposes, the user interacts with the cloud servers via CSP to access or retrieve his data. In some cases, the user may need to perform block level operations on his data.

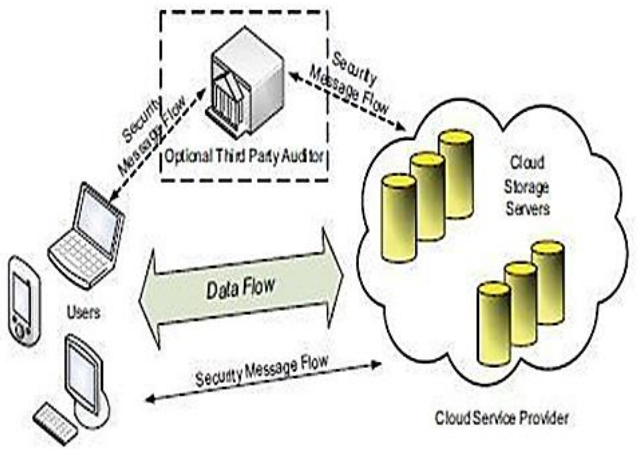

Figure 1 : Cloud data storage architecture

The most general forms of these operations we are considering are block update, delete, insert and append. As users no longer possess their data locally, it is of critical importance to assure users that their data are being correctly stored and maintained. That is, users should be equipped with security means so that they can make continuous correctness assurance of their stored data even without the existence of local copies. In case that user do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the 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. Note that we don't address the issue of data privacy in this paper, as in Cloud Computing, data privacy is orthogonal to the problem we study here.

B. Adversary Model

Security threats faced by cloud data storage can come from two different sources. On the one hand, a CSP can be self-interested, un trusted and possibly malicious. Not only does it desire to move data that has not been or is rarely accessed to a lower tier of storage than agreed for monetary reasons, but it may also attempt to hide a data loss incident due to management errors, Byzantine failures and so on. On the other hand, there may also exist an economically motivated adversary, who has the capability to compromise a number of cloud data storage servers in different time intervals and subsequently is able to modify or delete users' data while remaining undetected by CSPs for a certain period. Specifically, we consider two types of adversary with different levels of capability in this paper: Weak Adversary: The adversary is interested in corrupting the user's data files stored on individual servers. Once a server is comprised, an adversary can pollute the original data files by modifying or introducing its own fraudulent data to prevent the original data from being retrieved by the user. Strong Adversary: This is the worst case scenario, in which we assume that the adversary can compromise all the storage servers so that he can intentionally modify the data files as long as they are internally consistent. In fact, this is equivalent to the case where all servers are colluding together to hide a data loss or corruption incident.

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:

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.

D. 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.
- fkey (·) – pseudorandom function (PRF), which is defined as f : {0,1}
- ∗× key → GF(2p).
- 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.
- sver ij – 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.

Secure Data Storage in Cloud

In cloud storage system, companies stores their data in the remotely located data server. Accordingly, correctness of the data is assured. Even though sometimes unauthorized person may modify or delete the data which leads to server compromise and/or random Byzantine failures. Because it can be the first step for fast recovery of the storage errors. The cloud storage systems propose an effective and flexible distributed scheme with explicit dynamic data support for file distribution across cloud servers.

By computing homomorphic token using universal hash function which can be perfectly integrated with the verification of erasure-coded data. As well as it identifies misbehaving servers.

Finally, the procedure for file retrieval and error recovery based on erasure- correcting code is outlined.

Token Correctness

It achieves assurance for data storage correctness and data error localization, using pre-computed token. Before sharing file distribution using pre-computes a certain number of shortest verification token are generated that will ensure security for a block of data in a file in cloud storage. 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. After getting assurance of the user it again asks for authentication by which the user is confirmed to be the authenticated user. Upon receiving assurance, 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. All servers operate over the same subset of the indices, the requested response values for integrit y check must also be a valid codeword determined by a secret matrix. Suppose the user wants to challenge the cloud server's t times to make sure the correctness of data storage. Then, he must pre-compute t verification tokens for each function, a challenge key and a master key are used. To generate the token for server j, the user acts as follows the details of token Generations are shown in Algorithm 1.

- Derive an arbitrary value i and a permutation key based on master permutation key.
- Calculate the set of randomly-chosen index.
- Calculate the token using encoded file and the arbitrary value derived.

Algorithm 1 Token Pre-computation

```
Block of data is represented as l;

No. of .blocks is denoted as n;

Let f be the function and t be the token; Index
per proof is denoted as r;
Generate $M_k$ and $C_k$;
For point G (j); j->1, n execute
    /*j server position*/
    For round i->1, t execute
        /*i block index*/
        Derive i = f (i) and k (i) from master
        key. Compute v(j)
    End for
End for
Store all the views locally. End procedures
```

Correctness Verification and Error

Error localization is a key requirement for eradicate errors in storage systems. However, many previous schemes do not explicitly consider the problem of data error localization.

The challenges response protocol in our work future provides the localization of data error. Which only provides binary results about the storage state across the distributed service in predecessors? 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(s).

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

- The client reveals the i as well as the with key k (i) to each servers
- The server storing vector G aggregates those r rows
- Specified by index k(i) into a linear combination R
- Upon receiving R is from all the servers, the user takes away values in R.
- Then the user verifies whether the received values remain a valid codeword determined by secret matrix.

Because all the servers operate over the same subset of indices, the linear aggregation of these r specified rows (R (1) i , . . . ,R(n)i) 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.

III. RELATED WORKS

"Proof of irretrievability" (POR) model for ensuring the remote data integrity. Their scheme combines spot-checking and error correcting code to ensure both possession and irretrievability of files on archive service systems. Shacham [3] built on this model and constructed a random linear function based homomorphism authenticator which enables unlimited number of challenges and requires less communication overhead due to its usage of relatively small size of BLS signature. In their subsequent work, Ateniese [4] described a PDP scheme that uses only symmetric key based cryptography. This method has lower-overhead than their previous scheme and allows for block updates, deletions and appends to the stored file, which has also been supported in our work. However, their scheme focuses on single server scenario and does not provide data availability guarantee against server failures, leaving both the distributed scenario and data error recovery issue unexplored.

The explicit support of data dynamics has further been studied in the two recent works [5] and [6].

The incremental cryptography work done by Bellare [10] also provides a set of cryptographic building blocks such as hash, MAC, and signature functions that may be employed for storage integrity verification while supporting dynamic operations on data. However, this branch of work falls into the traditional data integrity protection mechanism, where local copy of data has to be maintained for the verification.

It is not yet clear how the work can be adapted to cloud storage scenario where users no longer have the data at local sites but still need to ensure the storage correctness efficiently in the cloud. In other related work, Curtmola [9] aimed to ensure data possession of multiple replicas across the distributed storage system. They extended the PDP scheme to cover multiple replicas without encoding each replica separately, providing guarantees that multiple copies of data are actually maintained. Very recently, C. Wang [8] gave a study on many existing solutions on remote data integrity checking, and discussed their pros and cons under different design scenarios of secure cloud storage services. Portions of the work presented in this paper have previously appeared as an extended abstract in [7].

We have revised the article a lot and add more technical details as compared to [7].The primary improvements are as follows: Firstly, we provide the protocol extension for privacy-preserving third-party auditing, and discuss the application scenarios for cloud storage service. Secondly, we add correctness analysis of proposed storage verification design. Thirdly, we completely redo all the experiments in our performance evaluation part, which achieves significantly improved result as compared to [7].We also add detailed discussion on the strength of our bounded usage for protocol verifications and its comparison with state-of-the-art.

IV. 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.

By utilizing the homomorphism token with distributed verification of erasure-coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s).Considering the time, computation resources, and even the related online burden of users, 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 be worry-free to use the cloud storage services. Through detailed security and extensive experiment results, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks.

V. REFERENCES

[1] S.Sajithabanu and Dr.E.George Prakash Raj, "Data Storage Security in Cloud" IJCST Vol. 2, Issue 4, Oct. - Dec. 2011

[2] A. Jules and J. Burton S. Kaliski, "Pors: Proofs of retrievability for large files," in Proc. of CCS’07, Alexandria, VA, October 2007, pp.584–597.

[3] H. Shacham and B. Waters, "Compact proofs of retrievability," in Proc. of Asiacrypt’08, volume 5350 of LNCS, 2008, pp. 90–107.

[4] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data

possession," in Proc. of Secure Comm'08,2008, pp. 1–10.

[5] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing,"in Proc. of ESORICS'09, volume 5789 of LNCS. Springer-Verlag, Sep.2009, pp. 355–370.

[6] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proc. of CCS'09, 2009, pp. 213–222.

[7] 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.

[8] 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.

[9] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "Mr-pdp:Multiple-replica provable data possession," in Proc. of ICDCS'08. IEEE Computer Society, 2008, pp. 411–420.

[10] M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental cryptography: The case of hashing and signing," in Proc. Of CRYPTO' 94, volume 839 of LNCS. Springer-Verlag, 1994, pp. 216–233.

[11] Amazon.com, "Amazon Web Services (AWS)," Online at http://aws.amazon.com, 2008.