DATA STORAGE SECURITY IN CLOUD COMPUTING

Freshka Kumari[1], Asst Prof. Mr. Sandeep Kumar[2]
M.Tech-CSE IV Semester (Multimedia Technology),
Department of Computer Science & Engineering, Kalinga University, Naya Raipur (C.G)

ABSTRACT: Cloud Computing has been envisioned as the next- generation architecture of IT Enterprise . In contrast to traditional solutions , where the IT services are under proper physical , logical and personnel controls , Cloud Computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy . This unique attribute , however , poses many new security challenges which have not been well understood . In this article , we focus on cloud data storage security , which has always been an important aspect of quality of service .To ensure the correctness of users' data in the cloud, we propose an effective and flexible distributed scheme with two salient features, opposing to its predecessors. 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. Unlike most prior works, the new scheme further supports secure and efficient dynamic operations on data blocks, including : data update, delete and append . 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.

Keywords: SaaS, PaaS , Iaas , homomorphic.

I. INTRODUCTION

Cloud computing is the most demanded advanced technology throughout the world. As cloud computing is an Internet based computer technology. Some of the major firms like Amazon, Microsoft and Google have implemented the "CLOUD" and have been using it to speed up their business. Cloud computing has given a new dimension to the complete outsourcing arena (SaaS, PaaS and IaaS) and they provide ever cheaper powerful processor with these computing architecture. The simplest thing that a computer does is to store in the available space and retrieve information whenever requested by the authenticated user. We can store any kind of data that we use in our day to day life from simple photographs, favorite songs, or even save movies to huge bulk amounts of data which is confidential. 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. In this paper, we propose an effective and flexible 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. Error Localization is the data corruption that 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). This is among first few ones in this field to consider distributed data storage in Cloud Computing. The main contribution can be recapitulated as the following aspects:
When compared to its predecessors they only provide binary results about the data storage status across the distributed servers, the protocol used in our work provides point of data error (i.e. Error Localization).
We provide secure and efficient dynamic operations on data blocks.

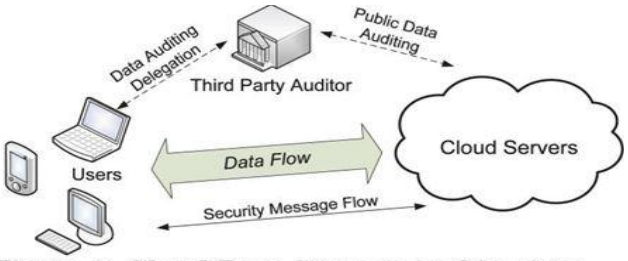

Figure 1: Cloud Data Storage Architecture

II. CLOUD STORAGE MODELS

There are models for cloud storage that allow users to maintain control over their data. Cloud storage [2] has evolved into three categories, one of which permits the merging of two categories for a cost-efficient and secure option. Public cloud storage providers, which present storage infrastructure as a leasable commodity (both in terms of long- term or short-term storage and the networking bandwidth used within the infrastructure). Private clouds use the concepts of public cloud storage but in a form that can be securely embedded within a user's firewall. Finally, hybrid cloud storage permits the two models to merge, allowing policies to define which data must be maintained privately and which can be secured within public clouds.

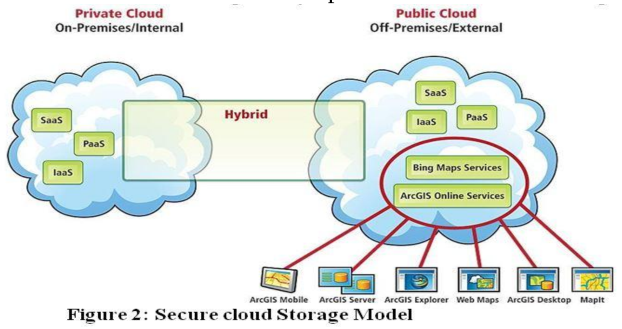

Figure 2: Secure cloud Storage Model

III. 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 simulta-neous, 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. 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 users 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, untrusted 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 make sure the security and dependability for data storage in cloud under the aforementioned antagonist model, we aim to design efficient mechanisms for dynamic data verification and operation and achieve the following goals:
- Storage accuracy: 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 mal- functioning server when data corruption has been detected.
- Dynamic data support: to maintain the same level of storage correctness assurance even if users modify, erase or affix their data files in the cloud.
- 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.
- 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).

φkey(·) – 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.

sij^{ver} – 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.

IV. SECURE DATA STORAGE IN CLOUD

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. 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 are 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 also shown how to derive a challenge response protocol for verifying the storage correctness as well as identifying misbehaving servers. Finally, the procedure forfile retrieval and error recovery based on erasure-correcting code is outlined.

A. Token exactness

In order to achieve assurance of data storage correctness and data error localization, our scheme entirely relies on the pre-computed verification tokens. The main idea is before file distribution the user pre-computes a certain number of short verification tokens on individual; each token covers 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. 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. 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 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 ith token for server j, the user acts as follows:
I. Derive a arbitrary value i and a permutation key based on master permutation key.
II. Compute the set of randomly-chosen indices:
III. Calculate the token using encoded file and the arbitrary value derived.

Algorithm 1 Token Pre-computation

```
• Procedure
• Choose parameters l, n and function f;
• Choose the number t of tokens;
• Choose the number r of indices per verification; 5. Generate master key and challenge key;
• for vector G(j), j ←1, n do
• for round i← 1, t do
• Derive i = f(i) and k(i) from master key .
• Compute v(j)
• end for
• end for
• Store all the vis locally.
• end procedure
```

B. Correctness Verification and Error Localization

Error localization is a key requirement for eradicating errors in storage systems. However, many previous schemes do not explicitly consider the problem of data error localization. Thus it only provides binary results for the storage verification. Our scheme provides 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(s).

Specifically, the procedure of the i^{th} challenge-response for a cross-check over the n servers is described as follows:
The user reveals the i as well as the i^{th} 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. Note that each response R(j) i is computed exactly in the same way as token v(j) i , thus the user can simply find which server is misbehaving by verifying the following n equations:
Algorithm 2 gives the details of correctness verification and error localization.

Algorithm 2
Correctness Verification and Error Localization

```
• procedure CHALLENGE(i)
• Recompute i = fl (i) and k(i) master key ;
• Send {i, k(i) } to all the cloud servers;
• Receive from servers R
• for (j ← m + 1, n) do
• R(j) ← R(j)−Prq=1 fkj (sIq,j)·_qi , Iq = _k(i)prp(q)
• end for
• if ((R(1)i , . . . ,R(m)i ) ·P==(R(m+1)i , . . .
• ,R(n)i )) then
• Accept and ready for the next challenge.
• else
• for (j ← 1, n) do
• if (R ! =V ) then
• return server is misbehaving.
```

- end if
- end for
- end if
- end procedure

V. PROVIDING DYNAMIC DATA OPERATION SUPPORT

So far, we assumed that F represents archived data. 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 revise, erase and affix to modify the data file while maintaining the storage correctness assurance. The straightforward and insignificant 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 unambiguously and efficiently handle dynamic data operations for cloud data storage.

A. Revise Operation

In cloud data storage, sometimes the user may need to modify some data block(s) stored in the cloud, from its current value f to a new one. We refer to this operation as data revise.

B. Erase Operation

Sometimes, after being stored in the cloud, certain data blocks may need to be erased. The erase 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 erase operation is actually a special case of the data revise 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 (not a single block) at one time.

D. Affix Operation

An affix operation to the data file refers to an affix operation at the desired index position while maintaining the same data block structure for the whole data file, i.e., inserting a block F corresponds to shifting all blocks starting with index j + 1 by one slot. An affix 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 affix operation is difficult to support and thus we leave it for our future work.

VI. RELATED WORK

Jules [2] described a formal “proof of retrievability” (POR) model for ensuring the remote dat a integrity. Their scheme combines spot-checking and error correcting code to ensure both possession and retrievability of files on archive service systems. Shacham [3] built on this model and constructed a random linear function based homomorphic 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. 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.

VII. CONCLUSION

In this paper, we studied the problem of data security in data storage in cloud servers. To guarantee the correctness of users’ data in cloud data storage, we proposed an effectual and flexible scheme with explicit dynamic data support, including block revise, erase, and affix. We use erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. Our scheme accomplishes the integration of storage correctness insurance and data corruption has been detected during the storage correctness verification across the distributed servers. Our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks. We believe that data storage security in Cloud Computing, an area full of challenges and of dominant significance, is still in its infancy to be identified. We envision several possible directions for future research on this area. It allows Third Parity Auditor to audit the cloud data storage without demanding users’ time, probability.

REFERENCES

[1] K. D. Bowers, A. Juels, and A. Oprea, “Proofs of Retrievability: Theory and Implementation,” Cryptology ePrint Archive, Report 2008/175, 2008, http://eprint.iacr.org/.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable Data Possession at Untrusted Stores,” Proc. of CCS ’07, pp. 598–609, 2007.

[3] 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.

[4] T. S. J. Schwarz and E. L. Miller, “Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage,” Proc. of ICDCS ’06, pp. 12–12, 2006.

[5] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, “A Cooperative Internet Backup Scheme,” Proc. of the 2003 USENIX Annual Technical Conference (General Track), pp. 29–41, 2003.

[6] 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/.

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

[8] 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.

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

[10] 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.

[11] 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.

[12] Juels and J. Burton S. Kaliski, “PORs: Proofs of Retrievability for Large Files,” Proc. of CCS ’07, 2007.

[13] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, “Scalable and Efficient Provable Data Possession,” Proc. of SecureComm ’2008.

[14] John W. Rittinghouse, James F. Ransome, ” Cloud Computing Implementation, Management, and Security”, CRC Press 2010 by Taylor and Francis Group, LLC.

[15] Journal of Theoretical and Applied Information Technology, “CLOUD COMPUTING”, www.jatit.org, 2005 – 2009.

[16] Hand, Eric. “Head in the Clouds.”Nature. 25;(2007 Oct).

[17] “Privacy in the Clouds: Risks to Privacy and Confidentiality from Cloud Computing”, Prepared by Robert Gellman for the World Privacy Forum February 23, 2009.

[18] “Advancing cloud computing: What to do now?, Priorities for Industry and Governments”, World Economic Forum in partnership with Accenture – 2011.

[19] Security of Cloud Computing Providers Study Sponsored by CA Technologies Independently conducted by Ponemon Institute LLC Publication Date: April 2011

[20] National Institute of Standards and Technology, “The NIST Definition of Cloud Computing,” document posted October 2009, http://csrc.nist.gov/groups/SNS/cloud-computing/.

[21] James Walden, Northern Kentucky University, The OWASP Foundation, http://www.owasp.org, “Cloud Computing Security”, February 22nd, 2011.

[22] Cloud Security Alliance(CSA), “Top Threats to Cloud Computing V1.0”, March 2010, http://www.cloudsecurityalliance.org/topthreats.