International Journal of Research in Computer Science
ISSN 2249-8257 Volume 1 Issue 1 (2011) pp. 63-73
© White Globe Publications
www.ijorcs.org

Secure Data Storage In Cloud Computing

B. Shwetha Bindu[1], B. Yadaiah[2]

[1]*Department of CSE, TKR College of Engineering & Technology*
Email: bswetha02@gmail.com

[2]*Department of CSE, TKR College of Engineering & Technology*
Email: yad524.balagoni@gmail.com

Abstract

Cloud computing has gained a lot of hype in the current world of I.T. Cloud computing is said to be the next big thing in the computer world after the internet. Cloud computing is the use of the Internet for the tasks performed on the computer and it is visualized as the next- generation architecture of IT Enterprise. The 'Cloud' represents the internet. Cloud computing is related to several technologies and the convergence of various technologies has emerged to be called cloud computing. In comparison to conventional ways Cloud Computing moves application software and databases to the large data centers, where the data and services will not be fully trustworthy. In this article, I focus on secure data storage in cloud; it is an important aspect of Quality of Service. To ensure the correctness of users' data in the cloud, I propose an effectual and adaptable scheme with salient qualities. This scheme achieves the data storage correctness, allow the authenticated user to access the data and data error localization, i.e., the identification of misbehaving servers.

Keywords: Cloud Computing, enterprise, servers, antagonist model, storage.

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 above mentioned service is the most basic service offered by cloud computing. Cloud is a pool of computing service on large 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.

The Cloud helps enterprises to have a dynamically scalable abstracted computing infrastructure that is available on-demand and on a pay-per-use basis. This model not only saves the IT teams from investing heavily on infrastructure, but also shields them from the intricacies involved in infrastructure setup and

management. Presently, apart from providing the on-demand IT infrastructure, cloud service providers typically provide interfaces for other related IT management services. Cloud based flexible and on-demand infrastructure enables a travel enterprise to offer mobility and social media channels without incurring any fixed cost.

Using a cloud infrastructure, a travel enterprise can start in a small way and grow into these evolving markets with a lower risk and financial strain.

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. At first, traditional cryptographic primitives for the purpose of data security protection can not be directly adopted due to the users' loss control of data under Cloud Computing. Therefore, we require verification of data storage in the cloud. 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 accuracy of data storage in the cloud becomes even more challenging. Secondly, Cloud Computing is not just a third party data warehouse. The stored data in cloud may be frequently revised by the users, including operations like insertion, deletion, modification, affixing, reordering, etc. To ensure storage correctness under dynamic data revise is hence of paramount importance. However, this dynamic feature also makes traditional integrity insurance techniques futile and entails new solutions. The deployment of Cloud Computing is powered by data centers running in a simultaneous, cooperated and distributed manner. Individual user's data is redundantly stored in multiple physical locations to further reduce the data integrity threats.

Ensuring storage correctness without having users possessing data, cannot address all the security threats in cloud data storage, since they are all focusing on single server scenario and most of them do not consider dynamic data operations. This is conquered using distributed protocols for ensuring storage correctness across multiple servers or peers. 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. By utilizing this token with distributed verification of erasure-coded data, our scheme achieves the storage correctness insurance as well as data error localization. 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.
- The security and performance analysis shows the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

Section II introduces the system model, our design goal and details. In section 3 we provide the detailed description of our scheme. Section 4 deals with providing dynamic data operation support, followed by Section 5 which overviews security issues and performance analysis. Finally, Section 6 gives the concluding remark of paper.

II. Problem Statement

A. Model

A descriptive architecture for secure data storage is illustrated in Figure 1. The different network entities can be identified as follows:

- **User:** Users may be a person or an organization who have data to be stored in the cloud and rely on the cloud for data computation.
- **Cloud Service Provider (CSP):** A CSP has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems.

Data storage in a cloud is a process where a user stores his data through a CSP into a set of cloud servers, which are running concurrently, cooperated and in 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 revise, erase, insert and affix.

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 prepared with security means so that they can make continuous correctness assurance of their data stored in Cloud Servers even without the existence of local copies. In case those users do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the tasks to an optional trusted Third Party Auditor of their respective choices. In our model, we assume that the peer-to-peer communication channels between each cloud server and the user is authenticated and reliable, which can be achieved in practice with little overhead.

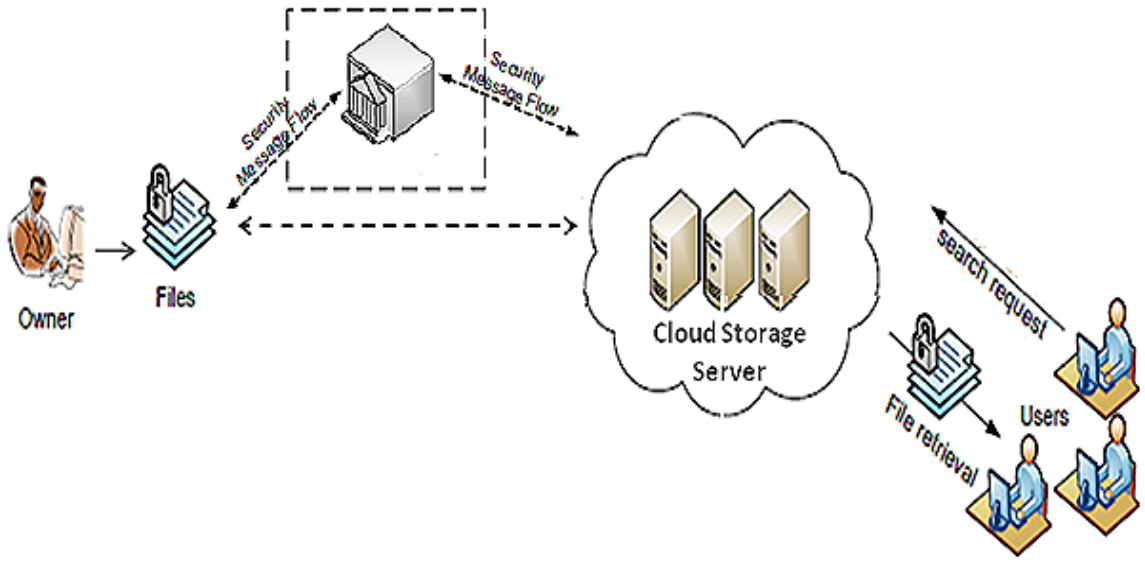

Fig. 1. Secure Data Storage Architecture

B. Antagonist Model

Security intimidation faced by data stored in cloud servers come from two different sources. One hand, a CSP can be self-centered, un-trusted and probably malevolent. Not only it desires to move data that has not been or is rarely accessed to a lower tier of storage than agreed for fiscal 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 antagonist, who has the capability to compromise a number of cloud data storage servers in different time intervals and subsequently is able to modify or erase users' data while remaining undetected by CSPs for a certain period. Specifically, we consider two types of Antagonist with different levels of capability:

Strong Antagonist: This is the worst case scenario, in which we assume that the antagonist 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.

Weak Antagonist: The antagonist is interested in corrupting the user's data files stored on individual servers. Once a server is comprised, an antagonist 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.

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:

i. **Storage accuracy:** to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud.

ii. **Fast localization of data error:** to effectively locate the mal- functioning server when data corruption has been detected.

iii. **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.

iv. **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.

v. **Lightweight:** to enable users to perform storage correctness checks with minimum overhead.

D. Notation and Preliminaries

F – Data file to be stored. **F** is denoted as a matrix of **m** equal-sized data vectors, each consisting of

l blocks.

A – Scattering matrix used for coding.

G – Encoded file matrix, which includes a set of

n = m + k vectors, each consisting of l blocks.

f – Function , which is defined as f : {0, 1} × key

III. 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 i^{th} 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

1. **procedure**
2. Choose parameters l, n and function f;
3. Choose the number t of tokens;
4. Choose the number r of indices per verification;
5. Generate master key and challenge key;
6. **for** vector G(j), j ←1, n **do**
7. **for** round i← 1, t **do**
8. Derive i = f(i) and k(i) from master key .
9. Compute v(j)
10. **end for**
11. **end for**
12. Store all the vis locally.
13. **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:

i) The user reveals the i as well as the i^{th} key k (i) to each servers
ii) The server storing vector G aggregates those r rows
iii) Specified by index k(i) into a linear combination R
iv) Upon receiving R is from all the servers, the user takes away values in R.
v) 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

1. **procedure** CHALLENGE(i)
2. Recompute i = fl (i) and k(i) master key ;
3. Send {i, k(i) } to all the cloud servers;
4. Receive from servers R
5. **for** (j ← m + 1, n) **do**
6. R(j) ← R(j)−Prq=1 fkj (sIq,j)·_qi , Iq = _k(i)prp(q)
7. **end for**
8. **if** ((R(1)i , . . . ,R(m)i) ·P==(R(m+1)i , . . . ,R(n)i)) **then**
9. Accept and ready for the next challenge.
10. **else**
11. **for** (j ← 1, n) **do**
12. **if** (R ! =V) **then**
13. **return** server is misbehaving.
14. **end if**
15. **end for**
16. **end if**
17. **end procedure**

IV. 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.

V. Security Issues And Performance Analysis

In this section, we analyze our proposed scheme in terms of security and efficiency. Generally, the checking scheme is secure if (i) there exists no polynomial-time algorithm that can cheat the verifier with non-negligible probability; (ii) there exists a polynomial-time extractor that can recover the original data files by carrying out multiple challenges-responses. Our security analysis focuses on the antagonist

model defined in Section II. We also evaluate the efficiency of our scheme via implementation of both file distribution preparation and verification token pre-computation.

A. Security Strength Against Weak Antagonist

i) Detection Probability against data modification:

In our scheme, servers are required to operate on specified list of tokens. These selected tokens greatly reduce the computational overhead on the server, while maintaining the detection of the data corruption with high probability. Note that if none of the specified r rows in the i^{th} verification process are erased or modified, the antagonist avoids the detection.

ii) Identification Probability for Misbehaving Servers:

We have shown that, if the antagonist modifies the data blocks among any of the data storage servers, our sample checking scheme can successfully detect the attack with high probability.

As long as the data modification is caught, the user will further determine which server is malfunctioning. This can be achieved by comparing the response values R with the pre-stored tokens **v**. The probability for error localization or identifying misbehaving server(s) can be computed in a similar way. It is the product of the matching probability for sampling check and the probability of complementary event for the false negative result.

Next, we consider the fake denial probability that R(j)=**v**(j) when at least one of z blocks are modified. Thus, the identification probability for misbehaving server(s) is predicted.

B. Security Strength against Strong Antagonist

We analyze the security strength of our schemes against server colluding attack and explain why blinding the parity blocks can help improve the security strength of our proposed scheme.

Redundancy parity vectors are calculated via multiplying the file matrix **F** by **P**, where **P** is the secret parity generation matrix we later rely on for storage correctness assurance. If we disperse all the generated vectors directly after token pre-computation, i.e., without blinding, malicious servers that collaborate can reconstruct the secret P matrix easily: they can pick blocks from the same rows among the data and parity vectors to establish a set of $m \cdot k$ linear equations and solve for the $m \cdot k$ entries of the parity generation matrix **P**. Once they have the knowledge of **P**, those malicious servers can consequently modify any part of the data blocks and calculate the corresponding parity blocks, and vice versa, making their codeword relationship always consistent. Therefore, our storage correctness challenge scheme would be damaged even if those modified blocks are covered by the specified rows, the storage correctness check equation would always hold. To prevent colluding servers from recovering **P** and making up consistently-related data and parity blocks, we utilize the technique of adding random perturbations to the encoded file matrix and hence hide the secret matrix **P**. We make use of a keyed pseudorandom function f with key k, both of which has been introduced previously.

C. Performance Evaluation

File Distribution Preparation is implemented for the generation of parity vectors for our scheme. This experiment is conducted using JAVA on a system with an Intel Core 2 processor running at 1.86 GHz, 2048 MB of RAM and 250 GB Serial ATA drive. Thus the cost decreases when more data vectors are involved. The performance of our scheme is comparable and even our scheme supports dynamic data operation while is for static data only.

Challenge Token Pre-computation: In our scheme we use fixed number of verification token t that are determined before file distribution, we can overcome this issue by choosing sufficient large t in practice.

VI. 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.

VII. References

[1] Robert Gellman and World Privacy Forum , "Privacy in the Clouds: Risks to Privacy and Confidentiality from Cloud Computing", February 23, 2009.

[2] Amazon.com, "Amazon Web Services (AWS)," Online at http://aws. amazon.com, Version 2010-01.

[3] Weiss, Aaron. Computing in the clouds. netWorker 11, 4 (Dec. 2007), <http://doi.acm.org/10.1145/1327512.1 327513>.

[4] Eric A. Marks, Bob Lozano "Executive's Guide to Cloud computing", John Wiley & Sons, Inc.

[5] Theart of Service, "A Complete Guide to Cloud Computing", http://theartofservice.com.

[6] Tim Mather, Subra Kumaraswamy, and Shahed Latif, "Cloud Security and Privacy", Published by O'Reilly Media, Inc.,- 2009

[7] Brian J.S. Chee and Curtis Franklin, Jr., "Cloud Computing, Technologies and Strategies of the Ubiquitous Data Center", CRC Press 2010 by Taylor and Francis Group, LLC.

[8] N.Gohring, "Amazon's S3 down for several hours," Online at http://www.pcworld.com/businesscenter/article/142549/amazons s3 down for several hours.html, 2008.

[9] Juels and J. Burton S. Kaliski, “PORs: Proofs of Retrievability for Large Files,” Proc. of CCS ’07, 2007.
[10] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, “Scalable and Efficient Provable Data Possession,” Proc. of SecureComm ’2008.
[11] John W. Rittinghouse, James F. Ransome, ” Cloud Computing Implementation, Management, and Security”, CRC Press 2010 by Taylor and Francis Group, LLC.
[12] Journal of Theoretical and Applied Information Technology, “CLOUD COMPUTING”, www.jatit.org, 2005 – 2009.
[13] Hand, Eric. “Head in the Clouds.”Nature. 25;(2007 Oct).
[14] “Privacy in the Clouds: Risks to Privacy and Confidentiality from Cloud Computing”, Prepared by Robert Gellman for the World Privacy Forum February 23, 2009.
[15] “Advancing cloud computing: What to do now?, Priorities for Industry and Governments”, World Economic Forum in partnership with Accenture – 2011.
[16] Security of Cloud Computing Providers Study Sponsored by CA Technologies Independently conducted by Ponemon Institute LLC Publication Date: April 2011
[17] National Institute of Standards and Technology, “The NIST Definition of Cloud Computing,” document posted October 2009, http://csrc.nist.gov/groups/SNS/cloud-computing/.
[18] James Walden, Northern Kentucky University, The OWASP Foundation, http://www.owasp.org, “Cloud Computing Security”, February 22nd, 2011.
[19] Cloud Security Alliance(CSA), “Top Threats to Cloud Computing V1.0”, March 2010, http://www.cloudsecurityalliance.org/topthreats.

How to cite

B. Shwetha Bindu, "Secure Data Storage In Cloud Computing". *International Journal of Research in Computer Science*, 1 (1): pp. 63-73, September 2011. doi:10.7815/ijorcs.11.2011.006