International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-1 Issue-8 March-2013

Verification of Data Reliability and Secure Service for Dynamic Data in Cloud Storage

Nithiavathy.R[1], Suresh.J[2]
Department of Computer Science &Engineering, Coimbatore Institute of Engineering and Technology[1,2]

Abstract

Cloud computing has been the genuine solution to the rising storage costs of IT Enterprises. The cost of data storage devices is too high rate at which data is being generated, where the enterprises or individual users to frequently update their hardware or software. The data outsourced to the cloud would help in reducing the maintenance. The user's data are moved from cloud to large data centers, which are located remotely which does not have control over it. Hence there is a security breech which has to be resolved. To address this issue, we propose an effective method to achieve secure and dependable cloud storage by using distributed storage integrity auditing mechanism, which incorporate homomorphic token and distributed erasure-coded data for dynamically storing data. The proposed design allows the user with lightweight communication and computation cost. To maintain reliable cloud storage correctness, and to locate the misbehaving server in which the data are frequently changing in cloud. It is an efficient method for dynamic operation which include erase, append, and block modification and it very effective in fighting against server colluding attacks, byzantine failure, malicious data block modifications.

Keywords

Error localization, data dynamics, Cloud Computing, Data Integrity, storage, Audit.

1. Introduction

Cloud is a large scale pool of computing service. The Cloud helps enterprises are dynamically scalable abstracted computing infrastructure that is available on-demand and on a pay-per-use basis [1][2]. This model saves the IT teams from use its huge capital on infrastructure. The growing network bandwidth allow the users to accesses a reliable yet flexible data and software that reside in remote data center spreads wide in the globe. Now both internal and external threats for data integrity still exist more in cloud [1][2].

To make sure of the correctness of storage without the users possessing their own data, it is difficult to address all data security threats in cloud storage as all concentrated in single server scenario and not consider dynamically changing data and its operation. By using distributed protocols for maintaining storage correctness in the multiple server or peers [3]. We propose a concrete, flexible and effective scheme with explicit dynamic data support to maintain the integrity of the user data in the cloud. We use erasure- correcting code in the distribution of the file in the cloud to avoid redundancies which increases the data dependencies. It overcomes the communication overheads of the traditional replication based techniques of file distribution.

Token utilization is used with distributed verification of the erasure –coded data which ensures the storage correctness and data error localization. The data corruption that has been detected during the verification of the correctness of the stored data is localized, which guarantee the data error localization simultaneously. It identifies the misbehaving server(s) [4].The verification is done without explicit knowledge of the data files. The user can check the integrity of data's in the cloud storage. The main part of the paper can be structured as the following aspects:

- 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 like update append and delete.
- 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. The paper is discussed as follows: Section 2 shows the

detail view of the system and adversary model, our design, goal, and notations. In section 3 we describe our scheme. Section 4 shows how our scheme supports dynamic data operation; Section 5 discusses the security issues and performance analysis. Finally, Section 6 is concluded with remarks of paper.

2. Problem statement

A. System model

The architecture for secure data storage is illustrated in Figure 1. The various network entities can be described as follows:

1. User (owner): Users may be a person or an organization that have data to be stored in the cloud and rely on the cloud for data computation.

2. 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.

3. Third –party Auditor (TPA)(optional): Who is expertise and capabilities that the user does not have and they are trusted to access and expose to the risk of cloud storage service for users. Data storage in a cloud is where the user stores his data through a CSP into a set of cloud servers, which are running concurrently, cooperated and in distributed manner. The redundancy of the data 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[5] [6]. The most general forms of operations we are considering are block revise, erase, insert and affix. The 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 [7]. In our model, the user communication channels between each cloud server and the user is authenticated and reliable, which can be achieved in practice with little overhead.

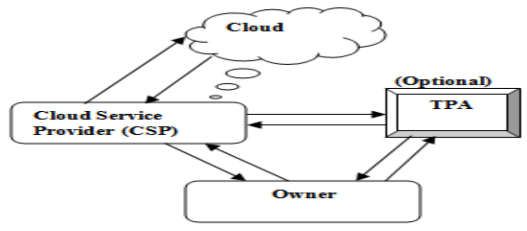

Fig 1: System model

B. Adversary Model

All kinds of threats toward his cloud data integrity are found in the adversary model from the user's view. The 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. For internal attacks, a CSP can be self-interested, untrusted, and possibly malicious; it may also attempt to hide a data loss incident due to management errors, Byzantine failures, and so on. For external attacks, data integrity threats may come from outsiders who are beyond the control domain of CSP, for example, the economically motivated attackers. They may compromise a number of cloud data storage servers in different time intervals and subsequently be able to modify or delete users' data while remaining undetected by CSP.Therefore, we consider the adversary in our model has the following capabilities, the cloud data integrity is maintained by capturing both external and internal threats. Continuously corrupting the user's data files by adversary, in the individual storage servers can cause loss of integrity of data .The server 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. This corresponds to the threats from external attacks. In the worst case scenario, 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 internal attack case where all servers are assumed colluding together from the early stages of application or service deployment to hide a data loss or Corruption incident.

C.Design goal

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.
¢key (.)- Pseudorandom function (PRF).
Ver-version number of individual block.
Sij-seed for PRF which depend on name, block index i, server position j and version number.

3. Secure data storage in cloud

The users store their data in the cloud server which is no longer available locally at the user side. Thus, the integrity 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 theories that are needed in our scheme for file distribution across cloud servers. Then, the homomorphic token is introduced[4]. 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 for file retrieval and error recovery based on erasure-correcting code is outlined.

A. Preparation of file distribution

To tolerate multiple failures in the distributed storage system we use erasure –correcting code. The technique is to dispraise the data file F redundantly across a set of n=m+k distributed server. Reed Solomon erasure code is used for creating k redundancy parity vectors from m data reconstructed from any m out of m+k data[5] .It can handle the failure without any data loss. Vandermonde matrix is used for dispersal matrix A is derive from m x (m+k), this is the layout of parity vector. The A matrix is written after the transformation as (I/P), where I is identity matrix and P is the secret parity vector. By multiplying F by A, the user obtains the encoded file, where F is the files to be stored in the cloud. The multiplication reproduces the original data file vectors of F and the remaining are k parity Vectors generated based on F.

B. Token exactness

Verification of tokens is done in order to achieve data storage correctness and data error localization, the pre-computed verification tokens for each data files that stored in cloud. 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 or the third party auditor makes sure the storage correctness for the data in the cloud, they challenge 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 an 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
```

Blinding the each parity block is important after the token precomputing; this is done for protection of secret matrix P. we use HMAC for parity blinding .The encode vector is dispersed in the cloud server. This can be done by the user.

C. Correctness verification and error localization

The key requirement of Error localization is to eradicating errors in storage systems. Previous schemes do not explicitly consider the problem of data error localization effectively.

Thus it only provides binary results for the storage verification. Integration of all correctness verifications, error localization in our challenge-response protocol is done. The procedure of the ith challenge-response for a cross-check over the n servers is described as follows:

i) The user reveals the i as well as the ith 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

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
```

D. File retrieval and error recovery

Spot-checking is randomly done for storage correction assurance. The user can always ask servers to send back blocks of the r rows specified in the challenge and regenerate the correct blocks by erasure correction. The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.

Algorithm 3. Error Recovery.

```
1: procedure
% Assume the block corruptions have been detected
Among % the specified r rows; % Assume s <= k
servers have been identified misbehaving
2: Download r rows of blocks from servers;
3: Treat s servers as erasures and recover the blocks.
4: Resend the recovered blocks to corresponding
servers.
5: end procedure.
```

E. Third Party Auditing

Our protocol can support privacy-preserving Third party auditing (optional). Encoding procedure in file distribution after blinding data vector, then the storage verification task can be successfully delegated to third party auditing in a privacy-preserving manner[8][9].The following is the protocol. The user blinds each file block data before file distribution k is the secret key for data vector is generated. Based on the blinded data vector, the User generates k parity vector via the secret matrix P. The user calculates the ith token for server j as previous scheme The user sends the token secret matrix P, permutation and challenge key $K_{\text{master key}}$, and k_{chal} to TPA for auditing delegation. The blinding values in the servers are not taken by TPA response of the server are verified directly. As TPA does not know the secret blinding key there is no way for TPA to learn the data content information during auditing process. Thus the privacy-preserving third party auditing is achieved with slight modification [9].

4. 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.

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; this is 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 where the original data blocks can be replaced with zeros or some predetermined special blocks.

C. Append Operation

Adding blocks at the end of the data file when user like to extend, 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.*Update Operation*

The user may need to modify some data block(s) stored in the cloud, after modification the block is updated. The random blinding information on parity blocks by subtracting the old and newly updated parity blocks. As a result, the secret matrix P is still being well protected, and the guarantee of storage correctness remains.

5. 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. We also evaluate the efficiency of our scheme via implementation of both file distribution preparation and verification token pre-computation.

i) 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.

ii) 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 ith verification process are erased or modified, the antagonist avoids the detection. Next, we consider the fake denial probability that $R(j)=\mathbf{v}(j)$ when at least one of z blocks are modified. Thus, the identification probability for misbehaving server(s) is predicted.

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 relies 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.

A. Performance evaluation

File Distribution Preparation is implemented for the generation of parity vectors for our scheme. We use Hmac for parity blinding which improves cost. 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 evens our scheme supports dynamic data operation while are 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.

6. 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, reliable and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks. It also an optional for the user to allows Third Party Auditor to audit the cloud data storage without demanding users' time, probability. 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.

References

[1] Cloudcomputing,http://en.wikipedia.org/wiki/Cloud_computing, Accessed: 24/08/2012.

[2] Cloud Computing**,** http://www.techno-pulse.com/ Cloud Computing for Beginners, Accessed: 24/08/2011.

[3] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing," Proc.IEEE INFOCOM, Mar. 2010.

[4] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," Proc. 17th Int'l Workshop Quality of Service (IWQoS '09), pp. 1-9, July 2009.

[5] Correction to the 1997 Tutorial on Reed-Solomon CodingJames S. Plank Ying DingUniversity of Tennessee Knoxville, TN 37996.

[6] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions,"http://www.techcrunch.com/2006/12/28/gmail-disasterreportsof-Mass-email-deletions, Dec. 2006.

[7] Privacy-Preserving Public Auditing for Secure Cloud Storage Cong Wang, Student Member, IEEE, Sherman S.M. Chow, Qian Wang, Student Member, IEEE,Kui Ren, Senior Member, IEEE, and Wenjing Lou, Senior Member, IEEE.

[8] Data storage auditing service in cloud computing: challenges, methods and opportunities Kan Yang · Xiaohua Jia.

[9] 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, July/Aug. 2010.

Nithiavathy.R has received under graduate degree in the field of Computer Science & Engineering, from Anna University Chennai. She is currently pursuing Post graduate from the same university. Her research interests include network security and privacy and cloud computing security.