ISSN 2319 – 7595

Volume 2, No.5, September - October 2013

International Journal of Information Systems and Computer Sciences

Available Online at http://warse.org/pdfs/ijiscs02252013.pdf

Auditing cloud storage for continuous storage security

P. Prathyusha[1], A. Suresh Babu[2]

[1]M. Tech in Computer Science and Engineering, JNTUACEP, Pulivendula, A.P, India,
prathyusha.cse592@gmail.com

[2]Assistant Professor, Department of Computer Science and Engineering, JNTUACEP, Pulivendula, A.P
asureshjntu@gmail.com

ABSTRACT

Cloud storage permits users to remotely store their knowledge and revel in the on-demand top quality cloud applications while not the burden of native hardware and software package management. Though the advantages are clear, such a service is addition relinquishing users' physical possession of their outsourced knowledge, which necessarily poses new security risks towards the correctness of the knowledge in cloud. In order to deal with this new drawback and additional succeed a secure and dependable cloud storage service, we tend to propose during this paper a versatile distributed storage integrity auditing mechanism, implementing the homomorphic token and distributed erasure-coded information. The planned style permits users to audit the cloud storage with terribly light-weight communication and computation value. The auditing result not solely ensures robust cloud storage correctness guarantee, but at the same time at identical time achieves fast information error localization, i.e., the recognition of misbehaving server. Considering the cloud information are energetic in nature, the proposed design further supports secure and desirable dynamic operations on outsourced information, including block modification, deletion, and append. Analysis shows the planned theme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and continuous server colluding attacks.

Keywords : data integrity, data error localization, data dynamics, dependable distributed storage, cloud computing.

1. INTRODUCTION

Cloud computing is the term used to share the resources globally with less cost .we can also called as "IT ON DEMAND". It provides 3 varieties of services i.e., Infrastructure as a service(IaaS),Platform as a service(PaaS) and package as a service(SaaS)[3]. The ever cheaper and additional powerful processors, along with the software as a service (SaaS) computing design, square measure reworking knowledge centers into pools of computing service on a large scale. End users access the cloud based applications through the web browsers with internet connection. Moving knowledge to clouds makes more convenient and reduce to manage hardware complexities. Knowledge stored at clouds are maintained by Cloud service providers (CSP) with various incentives for different levels of services[2]. However it deletes the responsibility of local machines to maintain knowledge, there is a chance to lose information or it effects from external or internal attacks[1]. Our contribution can be summarized as the following three aspects:

1) Compared to several of its predecessors, that only give binary results concerning the storage standing across the distributed servers, the planned theme achieves the combination of storage correctness insurance and knowledge error localization, i.e., the identification of misbehaving server(s).

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

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

2. PROBLEM STATEMENT

2.1 System model

A representative network architecture for cloud storage service architecture is illustrated in Figure. 1. Three different network entities will be known as follows:

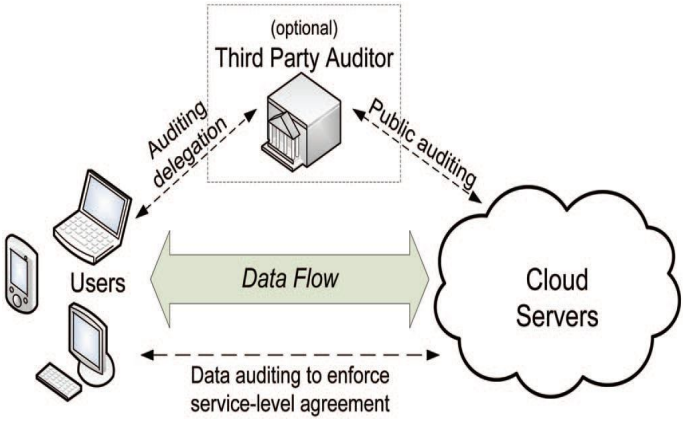

Figure. 1. Cloud storage service architecture.

- User: An entity, who has knowledge to be stored in the cloud and trusts on the cloud for data storage and computation, are often either enterprise or individual customers.
- Cloud Server (CS): An entity, that is ordered by cloud service provider (CSP) to provide knowledge storage service and has significant storage space and computation resources(we will not differentiate CS and CSP hereafter).
- Third-Party Auditor: An alternative 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 knowledge storage, a user stores his knowledge through a CSP into a set of cloud servers that are running in a simultaneous, cooperated, and distributed manner. Knowledge redundant techniques can be employed using erasure correcting code to protect from faults of server crashes.

Users can perform manipulations on stored knowledge like insert update and append entire blocks. Block level updating and deletions are allowed with token verifying. If user has not having enough resources to compute tokens or required hardware support then he can easily delegate the work to a third party auditor known as TPA. User is responsible to generate homomorphic token and stores the token persistently and securely for further verification. In our theme we assume that TPA is secure and responsible to protect from threats, users can pay some incentives to TPA for maintenance.

2.2 Adversary Model

From user's perspective, the adversary model needs to capture all kinds of threats toward his cloud knowledge integrity. Because cloud knowledge does not reside at user's local site but at CSP's address domain[2], these threats can come from two distinctive sources: internal and external attacks.

- Internal attacks: For internal attacks, a CSP can be self-interested, entrusted, and possibly malicious. Not solely does it desire to move data that has not been or is never accessed to a lower tier of storage than agreed for financial reasons, but it may also attempt to hide a data loss incident due to management defects, Byzantine failures, and so on.
- External attacks: For external attacks, data integrity threats can come from outsiders who are beyond the control domain of CSP, considering example, the economically motivated attackers. They may compromise a variety of cloud data storage servers in several time intervals and subsequently be able to adjust or delete users' information whereas remaining undetected by CSP.

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

2.3 Design goals

To ensure the security and responsibility for cloud data storage under the aforementioned opponent model[4], we aim to design economical mechanisms for dynamic data verification and operation and achieve the following goals:

- Storage correctness: to confirm users that their data are absolutely stored appropriately and kept intact all the time in the cloud.
- Fast localization of data error: to impressively locate the dead server when data corruption has been detected.
- Dynamic data support: to maintain the identical level of storage correctness assurance even though users modify, delete, or append their data files with in the cloud.
- Dependability: to enhance knowledge availability against Byzantine failures, malicious data modification and server colluding attacks, i.e., minimizing the effect brought by information errors or server failures.
- Lightweight: to enable users to achieve storage correctness checks with minimal overhead.

3. ENSURING CLOUD DATA STORAGE

In cloud data storage system, users store their data remotely i,e., on clouds, in order that the correctness and availability of data files should be sure to be identical. Our aim is to discover the servers that behaves differently and may leads to internal and external threats. In this project, we explore the technique used to detect the modified blocks easily with very less overhead using homomorphic token precomputation technique[1][7], later we can use erasure coded technique to acquire the desired blocks from different servers.

3.1 Challenge token free computation

In order to realize assurance of data storage correctness and information error localization at the same time, our theme entirely depends on the precomputed verification tokens. The main plan is as follows: before file separation the user precomputes an explicit variety of short verification tokens on individual vector, each token covering a random subset of knowledge blocks. Later, when the user wants to create certain the storage correctness for the data in the cloud[6], he challenges the cloud servers with a group of randomly generated block indices. Upon receiving challenge, each cloud server computes a short "signature" over the specified blocks and comebacks them to the user[5]. The values of those signatures should match the corresponding tokens precomputed by the user. Meantime, as all servers operate over the identical subset of the indices, the requested response values for integrity check should even be a correct codeword determined by the key matrix.

After token generation, the user has the selection of either keeping the precomputed tokens approximately or storing them in encrypted form on the cloud servers. In our case, the user stores them regionally to obviate the requirement for encryption and lower the bandwidth overhead throughout dynamic data operation which is able to be discussed shortly.

The information of token generation is shown in Algorithm1.

Algorithm 1. Token Precomputation.

1: **procedure**
2: Choose parameters l, n and function f, ϕ;
3: Choose the number t of tokens;
4: Choose the number r of indices per verification;
5: Generate master key K_{PRP} and challenge key k_{chal};
6: **for** vector $G^{(j)}, j \leftarrow 1, n$ **do**
7: **for** round $i \leftarrow 1, t$ **do**
8: Derive $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP}.
9: Compute $v_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)]$
10: **end for**
11: **end for**
12: Store all the v_i's locally.
13: **end procedure**

3.2 Correctness Verification and Error Localization

Error localization may be a key necessity for eliminating errors in storage systems. It is also of critical importance to detect potential threats from external attacks. However, many previous schemes don't expressly think about the problem of data error localization, therefore solely providing binary results for the storage verification[7]. Our theme outperforms those by integration the correctness verification and error localization in our challenge response protocol: the response values from servers for every challenge not solely verify the correctness of the distributed storage, but also include information to locate potential data error(s).

Algorithm 2. Correctness Verification and Error Localization.

1: **procedure** CHALLENGE(i)
2: Recompute $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP};
3: Send $\{\alpha_i, k_{prp}^{(i)}\}$ to all the cloud servers;
4: Receive from servers:
$\{R_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)] | 1 \le j \le n\}$
5: **for** $(j \leftarrow m+1, n)$ **do**
6: $R^{(j)} \leftarrow R^{(j)} - \sum_{q=1}^{r} f_{k_j}(s_{I_q,j}) \cdot \alpha_i^q$, $I_q = \phi_{k_{prp}^{(i)}}(q)$
7: **end for**
8: **if** $((R_i^{(1)}, \ldots, R_i^{(m)}) \cdot \mathbf{P} == (R_i^{(m+1)}, \ldots, R_i^{(n)}))$ **than**
9: Accept and ready for the next challenge.
10: **else**
11: **for** $(j \leftarrow 1, n)$ **do**
12: **if** $(R_i^{(j)} != v_i^{(j)})$ **than**
13: **return** server j is misbehaving.
14: **end if**
15: **end for**
16: **end if**
17: **end procedure**

3.3 File Retrieval and Error Recovery

Since our layout of file matrix is systematic, the user will reconstruct the original file by downloading the data vectors from the primary m servers, assuming that they return the correct response values. Notice that our verification theme is based on random spot-checking[5]; therefore the storage correctness assurance is a probabilistic one.

Algorithm 3. Error Recovery.

1: **procedure**
% Assume the block corruptions have been detected among
% the specified r rows;
% Assume $s \le k$ 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**

3.4 Toward Third Party Auditing

The user does not have the time, feasibleness or resources to perform the storage correctness verification; he can optionally delegate this task to an independent third party auditor, creating the cloud storage publicly verifiable and securely introduce an efficient TPA[1][2], the auditing process should bring g in no new vulnerabilities towards user data privacy. Namely, TPA should not learn user's knowledge content through the delegated data auditing. We tend to show that with only slight modification, our protocol can help privacy-preserving third party auditing. The new style is predicated on the observation of linear property of the parity vector bright process.

4. PROVIDING DYNAMIC DATA OPERATION SUPPORT

So far, we assumed that F represents static or archived data. This model may fit some application situations, like as libraries and scientific data sets. However, in cloud information storage, there are many potential scenarios where data stored in the cloud is dynamic, like electronic documents, photos, log files, etc. Therefore, it is crucial to consider the dynamic case, where a user may wish to perform different block level operations[7][8] of update, delete, and append to switch the modify the data file while maintaining the storage correctness assurance.

Update Operation

In cloud data storage, sometimes the user may need to modify some data block(s) stored in the cloud; we have a tendency to refer this operation as information update. In other words, for all the unused tokens, the user must exclude every occurrence of the old data block and replace it with the new one.

Delete Operation

Sometimes, after being kept within the cloud, certain data blocks may need to be deleted. The delete operation we are observing is a general one, during which user replaces the data block with zero or some special reserved information symbol. From this point of view, the delete operation is actually a special case of the data update operation, where the original data blocks will be replaced with zeros or some predetermined special blocks. Therefore, we are able to believe the update procedure to support delete operation. Also, all the affected tokens need to be changed and the updated parity information has to be blinded using the same method specified in an update operation.

Append Operation

In some cases, the user might want to increase the size of his stored data by adding blocks at the end of the information file, which we refer as data append. We anticipate that the foremost frequent append operation in cloud data storage is bulk append, during which the user needs to upload a large number of blocks (not a single block) at one time.

Insert Operation

An insert operation to the data file refers to an append operation at the desired index position while maintaining the same data block structure for the whole data file, i.e., inserting a block F[j] corresponds to shifting all blocks starting with index j+1 by one slot.

5. CONCLUSION

In this paper, we investigate the problem of data security in cloud data storage, that is actually a distributed storage system. To achieve the assurances of cloud knowledge integrity and availability and enforce the quality of dependable cloud storage service for users, we propose an efficient and flexible distributed scheme with explicit dynamic knowledge support, including block update, delete, and append[1]. Our coming direction is to implement a novel explanation mechanism for the problem of having high false intruders can also be resolved by one such approach that uses a high rate of accuracy in the case of any business method with human- understandable can also improve the efficiency for secure storage analyzing the complex dataset.

REFERENCES

[1] **"Towards Secure and Dependable Storage Services in Cloud Computing"** Cong Wand Student Member IEEE Qian Wang Kui Ran Ning Cao Student Members IEEE and Wenjing Lou Senior Member.

[2] C. Wang, Q. Wang, W. Lou, and K. Ren, "**Ensuring data storage security in cloud computing,**" in Proc. of IWQoS'09, July 2009, pp. 1–9.

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

[4] Sun Microsystems, Inc., "**Building customer trust in cloud computing with transparent security,**" Online https://www.sun.com/ offers/details/suntransparency.xml, Nov 2009.

[5] R. Burns, R. Curtmola, G. Ateniese, L. Kissner, J. Herring, Z. Peterson, and D. Song, "**Provable data possession at untrusted stores,**" in Proc. of CCS'07, VA, Alexandria, October 2007, pp. 598–609.

[6] Amazon.com, "Amazon s3 availability event: jul2008,"Online http://status.aws.amazon.com/s3-20080720.html, July 2008.

[7] J. Li, K. Ren, W. Lou (2009), Q. Wang, C. Wang, "**Enabling public verifiability and data dynamics for storage security in cloud computing**", in Proc. of ESORICS'09, Saint Malo.

[8] K. Ren, and W. Lou, C. Wang, S.S.M. Chow, Q. Wang, "**Privacy-Preserving Public Auditing for Secure Cloud Storage,**" IEEE Trans. Computers, preprint, 2012, doi:10.1109/TC.2011.245.