ISSN No. 0976-5697

Volume 4, No. 5, May 2013 (Special Issue)

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

Privacy - Preserving Audit of Secure Data Storage Services in Cloud Computing

M.Kanchana
M.Tech (CSE),
Viswodaya Engineering College,
Kavali, Nellore.
Kanchana1238@gmail.com

M. Kiran Kumar,
Assistant Professor, CSE,
Jagan's college of Engineering & Technology,
Nellore.
madirikiran@gmail.com

Sk. Nazar hussain,
Assistant Professor, CSE,
Brahmas College of Engg &Tech, Nellore.
nazarhussain76@gmail.com

C. Praveen,
Associate Professor, CSE,
Viswodaya Engineering College,
Kavali, Nellore, madirikiran@gmail.com

Abstract: Cloud computing is the delivery of computing as a service rather than a product. It provides shared resources, software, and information to computers and other devices over a network. 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. We can store and retrieve the data as we like using cloud computing. we propose in this paper a flexible distributed storage integrity auditing mechanism, utilizing the homomorphic token and distributed erasure-coded data. The proposed design allows users to audit the cloud storage with very lightweight communication and computation cost. The auditing result not only ensures strong cloud storage correctness guarantee, but also simultaneously achieves fast data error localization, i.e., the identification of misbehaving server. Considering the cloud data are dynamic in nature, the proposed design further supports secure and efficient dynamic operations on outsourced data, including block modification, deletion, and append.

Index Terms— Cloud Storage Services, data integrity, dependable distributed storage, data dynamics, 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 three types of services i.e., Infrastructure as a service(IaaS) , Platform as a service(PaaS) and Software as a service(SaaS). The ever cheaper and more powerful processors, together with the software as a service (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. End users access the cloud based applications through the web browsers with internet connection. Moving data to clouds makes more convenient and reduce to manage hardware complexities.

Data stored at clouds are maintained by Cloud service providers (CSP) with various incentives for different levels of services.

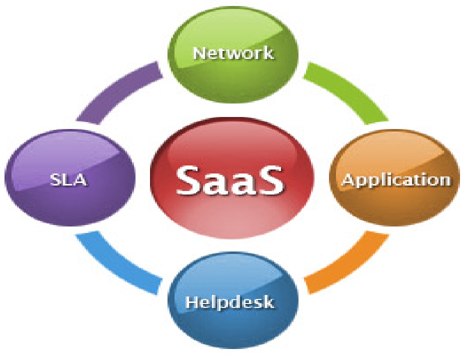

Fig 1.SaaS Structure

End users access the cloud based applications through the web browsers with internet connection. Moving data to clouds makes more convenient and reduce to manage hardware complexities. Data stored at clouds are maintained by Cloud service providers (CSP) with various incentives for different levels of services. However it eliminates the responsibility of local machines to maintain data, there is a chance to lose data or it effects from external or internal attacks. To maintain the data integrity and data availability many people proposed several algorithms and methods that enable on demand data correctness and verification. So Cloud servers are not only used to store data like a ware house , it also provides frequent updates on data by the users with different operations like insert, delete , update and append.

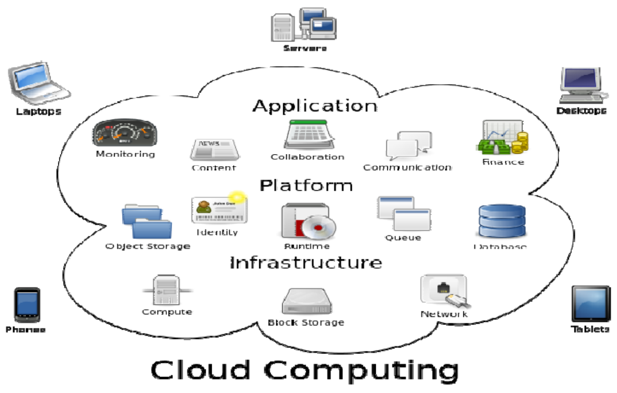

Fig 2: Architecture of cloud computing

II. PROBLEM STATEMENT

2.1 System Model

The cloud storage system architecture consists of following network entities

User: An entity, which performs data storage and retrieval operations without knowing the internal issues.

Cloud Server (CS): An entity, which provides data storage space and resources, required for computations, cloud servers are managed by cloud service providers.

Third Party Auditor (TPA): An optional Entity, but here we use TPA as Trusted party and to perform some computations instead of users.

Working:

In cloud data storage system, user can upload or stores the data into cloud or use services from the cloud (Here we focused on file storage and retrieval operations). User stores data into set of cloud servers which are running in a distributed and cooperated manner. Data redundant techniques can be employed using erasure correcting code to protect from faults or server crashes.

Users can perform manipulations on stored data like insert update and append through blocks. Block level updating and deletions are allowed with token checking. 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 called as TPA. He is responsible to generate homomorphic token and stores the token persistently and securely for further verification. In our scheme we assume that TPA is secure and he is responsible to protect from threats, users will pay some incentives to TPA for maintenance.

2.2. Adversary Model

Adversary model was introduced to explore some of threats associated in this model. As we know that the data is not present at users place because data is stored at cloud servers. It may lead to some security threats mainly two, internal attacks and external attacks. Internal attacks comes from the cloud servers itself, these servers may be malicious and lead to byzantine failures and hide some data loss issues. Secondly external attacks are from outsiders who are compromised the data from cloud service providers without its permission. Outsider attacks may lead to modification of data or deleting the users and so on which are completely masked from cloud service providers. All though TPA can also possibly hack the data for itself interested and it is also a case for inside attacks, but we ensure that TPA's are trusted party servers. Therefore, we consider the adversary in our model to capture all types of attacks both internal and external threats. Once the server is compromised, the data is polluted with fraudulent data and users cannot get the original data from the clouds.

2.3 Design Goals

To ensure the security and dependability for cloud data storage under the aforementioned adversary model, we aim to design efficient mechanisms for dynamic data verification and operation and achieve the following goals:

(1) Storage correctness: to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud.

(2) Fast localization of data error: to effectively locate the malfunctioning server when data corruption has been detected

(3) Dynamic data support: to maintain the same level of storage correctness assurance even if users modify, delete or append their data files in the cloud.

(4) Dependability: to enhance data availability against Byzantine failures, malicious data modification and server colluding attacks, i.e. minimizing the effect brought by data errors or server failures.

(5) Lightweight: to enable users to perform storage correctness checks with minimum overhead.

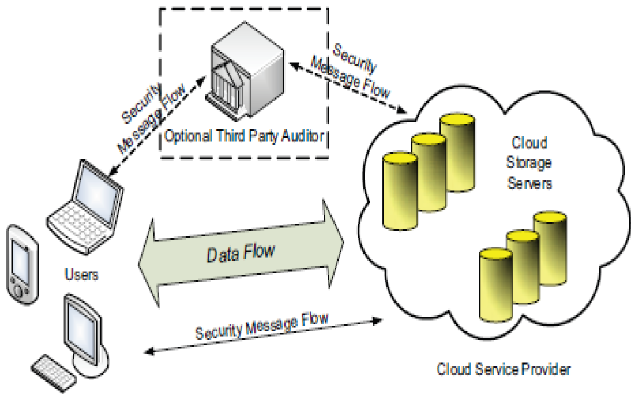

Fig 3: Cloud Storage System Architecture

III. ENSURING DATA STORAGE OVER CLOUD

In cloud data storage system, users store their data remotely i,e., on clouds, so that the correctness and availability of data files must be guaranteed to be identical. Our aim is to detect the servers which behaves differently and may leads to internal and external threats. In this paper, we explore the technique used to detect the modified blocks easily with very less overhead using homomorphic token pre computation technique ,later we can use erasure coded technique to acquire the desired blocks from different servers

3.1. Challenge Token Pre-Computation

To achieve data storage correctness and data integrity, we use an algorithm which takes a few parameters and compute the token

Algorithm 1: Token pre computation

```
1:  procedure
2:      Choose parameters $l, n$ and function $f, \phi$;
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 is a key prerequisite for eliminating errors in storage systems. It is also of critical importance to identify potential threats from external attacks. However, many previous schemes do not explicitly consider the problem of data error localization, thus only providing binary results for the storage verification.

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:
        $\{R_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)] | 1 \leq j \leq n\}$
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 (($R_i^{(1)}, \ldots, R_i^{(m)}$)·P==($R_i^{(m+1)}, \ldots, R_i^{(n)}$)) ther
9:          Accept and ready for the next challenge.
10:     else
11:         for ($j \leftarrow 1, n$) do
12:             if ($R_i^{(j)}! = v_i^{(j)}$) then
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 can reconstruct the original file by downloading the data vectors from the first m servers, assuming that they return the correct response values. Notice that our verification scheme is based on random spot-checking, so 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 $s \leq k$ servers have been identified misbe-
    having
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. Towards Third Party Auditing

The user does not have the time, feasibility or resources to perform the storage correctness verification; he can optionally delegate this task to an independent third party auditor, making the cloud storage publicly verifiable and securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities towards user data privacy. Namely, TPA should not learn user's data content through the delegated data auditing. We show that with only slight modification, our protocol can support privacy-preserving third party auditing. The new design is based on the observation of linear property of the parity vector blinding process.

IV. CLOUD OPERATIONS

4.1. Update Operation

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

4.2. Delete Operation

Sometimes, after being stored in the cloud, certain data blocks may need to be deleted. The delete 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 delete operation is actually a special case of the data update operation, where the original data blocks can be replaced with zeros or some predetermined special blocks.

4.3. 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.

4.4. 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.

V. PERFORMANCE EVALUATION

5.1 File Distribution Preparation

The file distribution preparation includes the generation of parity vectors (the encoding part) as well as the corresponding parity blinding part. We consider two sets of different parameters for the (m, k) Reed- Solomon encoding, both of which work over GF(216). Figure 3 shows the total cost for preparing a 1 GB file before outsourcing. In the figure on the left, we set the number of data vectors m constant at 10, while decreasing the number of parity vectors k from 10 to 2. In the one on the right, we keep the total number of data and parity vectors m + k fixed at 22, and change the number of data vectors m from 18 to 10. From the figure, we can see the number k is the dominant factor for the cost of both parity generation and parity blinding

5.2 Challenge Token Computation

Although in our scheme the number of verification token t is a fixed priori determined before file distribution, we can overcome this issue by choosing sufficient large t in practice. For example, when t is selected to be 7300 and 14600, the data file can be verified every day for the next 20 years and 40 years, respectively, which should be of enough use in practice. Note that instead of directly computing each token, our implementation uses the Horner algorithm suggested in to calculate token vi(j) from the back, and achieves a slightly faster performance. Specifically

$$v_i^{(j)} = \sum_{q=1}^{r} \alpha_i^{r+1-q} * G^{(j)}[I_q] = (((G^{(j)}[I_1] * \alpha_i + G^{(j)}[I_2]) * \alpha_i + G^{(j)}[I_3] \ldots) * \alpha_i + G^{(j)}[I_r]) * \alpha_i,$$

Which only requires r multiplication and (r − 1) XOR operations. With Jerasure library, the multiplication over GF (216) in our experiment is based on discrete logarithms.

VI. CONCLUSION

In this paper, we investigate the problem of data security in cloud data storage, which is essentially a distributed storage system. To achieve the assurances of cloud data integrity and availability and enforce the quality of dependable cloud storage service for users, we propose an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. Our future 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.

VII. REFERENCES

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

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

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

[4] J. Kincaid, "MediaMax/TheLinkup Closes Its Doors," Online at http://www.techcrunch.com/ 2008/07/10/ mediamaxthelinkup-closes-its- doors/, July 2008.

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