IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 12, Issue 4 (Jul. - Aug. 2013), PP 69-74
www.iosrjournals.org

Secure Way of Storing Data in Cloud Using Third Party Auditor

[1]Miss. Roopa G, [2]Mr. Manjunath S
[1]*CNE, CMRIT Bangalore, India*
[2]*CSE, CMRIT Bangalore, India*

Abstract: *Cloud computing provides users to remotely store their data and high quality of cloud applications by reducing risk of local hardware and software management. To protect data from outsourced leakages, we propose a flexible distributed storage integrity auditing mechanism (FDSIAM), these mechanism utilizes the homomorphic tokens (MAC address and secret key), blocking and unblocking factors and distributed erasure-coded data. Suppose user is out of station we can generate alert information to mobile devices, if any attacker attacks the data immediate message has been given to particular user through mobiles. The auditing results not only ensure storage correctness, but simultaneously identifies misbehaving server. Our proposed design provides very lightweight communication and computation cost for users to audit the data. Cloud data are dynamic in nature, our design supports secure and efficient dynamic operations on outsourced leakages, including block insertion, modification, append and delete operations. Our experimental results are highly efficient and secure against malicious data modification attack, server colluding attacks and byzantine failures.*
Keywords- *Data integrity, dependable distributed storage, Fast data error localization, data dynamics, cloud computing, data redundancy.*

Submitted date 17 June 2013 **Accepted Date: 22 June 2013**

I. Introduction

Cloud computing is a subscription based service, which can be used to provide security for user data. Cloud is powerful processors and cheaper technique which provides Platform as a service (PaaS), Software as a service (SaaS) used to transforming data centers into pools for services. The cloud computing well known examples are Amazon Elastic Compute Cloud and Amazon Simple Storage Services[1]. As a result we can achieve data integrity and availability. Cloud infrastructure is much more powerful and reliable then personal computers for both external and internal threats still exist. Cloud service providers (CSP) discards rarely accessed data which cannot be detected in timely manner, so that we can increase profit margin by reducing the cost. Cloud relies on either an enterprise or individual cloud users.

In order to achieve the cloud data integrity and availability and quality of cloud storage services, efficient method data correctness verification for cloud users has been designed, means that it allows user to verify the data present in correct format or not. The cloud prohibits the traditional cryptography technique for the purpose of data integrity. The data stored in the cloud not only accessed but frequently updated to the users [8], including block insertion, modification, append and delete operations. Cloud computing is a data centers running in distributed, simultaneous and cooperated manner [4]. It is more advantage for individual users to store their data across multiple servers to reduce data availability and data integrity threats. Thus, distributed storage correctness is most important in achieving secure and robust cloud storage systems. In our paper, we propose an effective and flexible distributed storage integrity auditing mechanism for dynamic data support to ensure the correctness and availability of data in cloud. We use erasure correcting code in file distribution preparation to provide data redundancy and also guarantee the data dependability against failures. By utilizing homomorphic tokens and distributed erasure-coded data our scheme identifies fast data error localization i.e., whenever data has been corrupted it identifies misbehaving server(s). In order to save computation resources, time and online burden of users, we extend our scheme to third-party auditor(TPA), where users can safely delegate the integrity checking tasks and reduces the burden to use cloud storage services. Our contribution consists of three aspects:

II. Problem Statement

2.1System Model

The network architecture for cloud storage services as shown in the Figure1. There are four different entities can be identified. Namely:

- Data Owner: an entity, who has the data stored in cloud and relies on cloud for data storage and computation resources. It supports both an enterprise and individual users.
- Cloud Server or Cloud Service Provider (CSP): an entity, which can be managed by cloud service provider (CSP) to provide computation resources and has significant storage space.
- Third-Party Auditor (TPA): who has expertise and have more capabilities that users may not have, he is trusted at owner side.
- User: an entity, who can access the data if it matches with tokens otherwise user must be attacker.

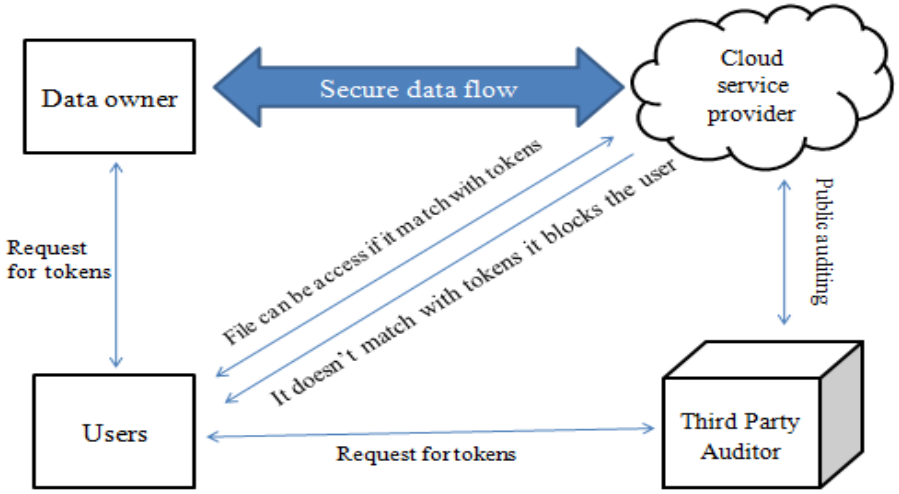

Fig 2.1 Cloud Storage Service Architecture

In cloud data storage, data owner stores his data in cloud through cloud service provider (CSP) that can be running in cooperated, distributed and simultaneous manner. Data redundancy can be achieved with the erasure-correcting coded technique for further tolerate server crashes or faults as users data grows in size. For application purposes, data owner interacts with cloud server via CSP to access the data. In some cases, owner performs block level operations including block insertion, modification, append and delete. As users no longer possess their data locally, it is most important to verify user data has been correctly stored and maintained in the CSP. Data owner should be equipped with security by making continuous correctness assurance. However for more security, we introduce TPA any possible leakages from outsourced data towards TPA through auditing protocol should be restricted. To increase the security levels in our paper, user request tokens from both TPA and data owner then only CSP provide access of data files to the user.

2.2 Adversary Model

Adversary model aim is to corrupt the users data file stored in cloud server. It captures the all kinds of threats towards cloud data integrity. Threats may come from two different sources: internal and external attacks.

Internal attack: a CSP can be untrusted, self-interested and malicious. CSP hides the data loss due to byzantine failures, management errors and so on.

External attack: threats may come from outsiders who are beyond CSP domain. In this case, they may compromise with number of cloud servers in different time intervals and intentionally modify the original data. Once server compromised, an adversary can destroy the entire original data files present in the cloud servers. In worst case situation, adversary can compromise all cloud storage services so he can intentionally modify data files as long as they are internally present.

2.3Design Goals

To ensure the security and dependability for cloud data storage against adversary model, our aim is to design efficient dynamic data verification and operations as follows:

- Storage correctness: to ensure user data stored correctly and keeps all time contact with cloud.
- Fast data error localization: whenever data has been corrupted it identifies misbehaving server(s).
- Dynamic data support: maintains same level of data even if users append, delete or modify their data files in the cloud.

- Dependability: provides data availability against malicious data modification, server crashes and byzantine failures.
- Lightweight: users perform storage correctness checks with minimum cost.

III. Ensuring Cloud Data Storage

In cloud storage system, users store their data in cloud and no longer possess the data locally. Thus the data availability and correctness of data files being stored on distributed servers must be guaranteed. One of the key factors is to detect unauthorized data modification and corruption is mainly due to server compromise or byzantine failures. In distributed systems, inconsistencies are successfully detected, to find which server is misbehaving, since it provides the fast recovery of errors or identifies the threats of external attacks. To address this problem, our main scheme for ensuring data storage in cloud is presented. Then homomorphic tokens can be introduced, using this we can perfectly integrated with erasure-coded data[10][7] for verification. The Challenge-response technique,used for identifies misbehaving servers and also for storage correctness of data. Third-Party Auditor scheme can be used for error recovery, which provides data integrity.

3.1 Challenge Token Pre-computation

In order to achieve storage correctness of data and fast data error localization, our scheme uses pre-computed verification tokens. In this case, before file distribution user pre-computes certain number of tokens on individual vectors $G^{(i)}$ ($i \in \{1,\ldots..,n\}$), each tokens covering a random subset of data blocks. Suppose when owner wants to make sure that his data present in cloud in correct format, he challenges the cloud servers with randomly generated data blocks. Upon receiving challenge, each cloud server computes short" signature" over specified blocks and returns to the data owner . The value of these signatures should match with corresponding tokens pre-computed by user. Meanwhile, all servers operates at same time, the request response values for integrity check must be valid for codeword determine by secret key matrix **P**.

1. **Start**
2. Choose function ϕ, f and parameters n, l;
3. Choose the number t of tokens;
4. Choose the number r if indices per verification;
5. Generate master key M_{PRP} and challenge key C_{key};
6. **for** vector $G^{(i)}$, i<- 1, n do
7. **for** round j<-1, t do
8. Derive $\beta_j = f_{ckey}(j)$ and $k_{prp}^{(j)}$ from M_{PRP}.
9. Compute $u_j^{(i)} = \sum^{r}_{q=1} \alpha^{q}_{j} * G^{(i)} [\phi_{kprp}^{(j)} (q)]$
10. **end for**
11. **end for**
12. Store all u_j's locally
13. **End**

Whenever data owner wants to challenge the cloud service provider t times to verify correctness of data storage.Then, he must pre-compute t verification tokens for each individual vector $G^{(i)}$ ($i \in \{1,\ldots,n\}$), using a PRP $\phi(.)$, PRF f(.), a challenge key C_{key} and permutation master key M_{PRP}. Particularly, to generate the j^{th} token for server i, the owner acts as follows:

- Derive a random challenge value β_j of $GF(2^p)$ by $\beta_j = f_{ckey}^{(j)}$ and permutation key $k_{prp}^{(j)}$ based on master key M_{PRP}.
- Compute set of r random indices: $Iq = \phi_{kprp}^{(j)} (q)$.
- Calculate the tokens:

$u_i^{(j)} = \sum^{r}_{q=1} \beta^{q}_{i} * G^{(i)} [Iq] = g_{Iq}^{(i)}$

Where $u_i^{(j)}$ = is an element of $GF(2^p)$, which receives the responses from server i where owner challenges it on specified blocks of data.

- Once all tokens are computed, final step is to blind each parity block $g_j^{(i)}$ before file distribution in $(G^{(m+1)},\ldots G^{(n)})$ by

$g_j^{(i)} <- g_j^{(i)} + f_{ki}(s_{ji})$, $j \in (1,\ldots., l)$,

Where k_i is secret key for parity vector $G^{(i)}$ ($i \in \{m+1,\ldots,n\}$).This is for protecting of secret key matrix **P**. After blinding parity information, the owner disperses all n encoded vectors $G^{(j)}$ ($i \in \{1,\ldots,n\}$) across cloud servers s_1, s_2, $s_3,\ldots..,s_n$.

3.2 Correctness Verification and Error Localization

Error localization is the key factor used for eliminating errors in cloud storage systems. It is also important to identify potential threats from external attacks [6][12].Our scheme integrates the storage correctness verification and data error localization(i.e., identifies misbehaving servers). In challenge response

protocol the response values from servers not only determine the correctness but also provide information to locate data errors. Once the inconsistency has been detected, we rely on pre-computed verification tokens to further determine where the data error occurs.

1. **Start** CHALLENGE(j)
2. Recompute $\beta_j = f_{Ckey}(j)$ and $k_{prp}^{(j)}$ from M_{PRP};
3. Send { β_j, $k_{prp}^{(j)}$} to all cloud servers;
4. Receive from servers:
   $\{R_j^{(i)} = \sum^r{}_{q=1} \beta^q{}_j * G^{(i)}[\phi_{kprp}{}^{(j)}(q)]|1<=i<=n\}$
5. **For**(i<- m+1, n) do
6. $R^{(i)}$<- $R^{(i)} - \sum^r{}_{q=1} f_{ki}(s_{Iq}, i) . \beta^q{}_j$, $I_q = \phi_{kprp}{}^{(j)}(q)$
7. **End for**
8. **If**(($R_i^{(1)}$,…..,$R_i^{(m)}$). P = =($R_i^{(m+1)}$,….,$R_i^{(n)}$)) **then**
9. Accept and ready for next challenge.
10. **Else**
11. **For** (i<-1, n) **do**
12. **If** ($R^{(i)}{}_j$! = $V^{(i)}{}_j$) then
13. Return server i is misbehaving.
14. **End if**
15. **End for**
16. **End if**
17. **End**

The procedure of j-th challenge-response for cross check over n servers is as follows:

- The data owner reveals β_j or j-th permutation key $k_{prp}^{(j)}$ to each cloud servers.
- The server storing vector $G^{(i)}$ (i €{1,…,n}) aggregates r rows specified by $k_{prp}^{(j)}$ into linear combination $R^{(i)}{}_j = \sum^r{}_{q=1} * G^{(i)}[\phi_{kprp}{}^{(j)}(q)]$.

And send back $R^{(i)}{}_j$ (i €{1,…,n}).

- After receiving $R^{(i)}{}_j$'s from all servers, owner takes away blind values in $R^{(i)}{}_j$ (i €{m+1,…,n}) by,

$R^{(i)}{}_j$ <- $R^{(i)}{}_j - \sum^r{}_{q=1} f_{ki}(s_{Iq}, i) . \beta^q{}_j$

- Then data owner verifies whether the received value is a valid codeword determined by secret key matrix **p**: ($R_j^{(1)}$,….$R_j^{(m)}$) . P = ($R_j^{(m+1)}$,…,$R_j^{(n)}$).
- Once inconsistency has been successfully detected among the data storage, we relies on token pre-compute verification to further determine data errors that occures. The computed tokens $u_i{}^{(j)}$ is exactly equal to response values of the server $R^{(i)}{}_j$, thus owner can easily find misbehaving servers by $R^{(i)}{}_j = u_j{}^{(i)}$, i € {1,…,n}.

3.3 File Retrieval and Error Recovery

The file matrix is symmetric, the user can reconstruct original data files by downloading data vectors from first 'm' servers, and they return correct response values. The verification of token scheme is mainly based on random spot checking. Whenever data has been corrupted, the comparison of pre-computed tokens and received response values can guarantee the identification of misbehaving servers. Therefore, the user can always ask servers to send back blocks of 'r' rows and regenerate the correct blocks by erasure correction. Error recovery can be done by Third-Party Auditor, using TPA we can achieve data integrity and availability[6].

1. **Start**

% Assume the data file corruptions has been detected among the specified r rows;
% Assume s<= k has identified has misbehaving servers

2. Download r rows of data blocks from servers;
3. Assume s servers as erasure and recover data blocks.
4. Resend recover blocks to corresponding servers.
5. **end**

3.4Remote User

To access the data file, user has to request tokens from data owner and TPA for security purpose. After receiving request he has to send tokens(MAC and Secrete key) to cloud server if it match with owner file it permits to access the file for remote user if it does not match with particular tokens the user may be an attacker and blocked in the cloud server using TPA.

```
Private void btnReqToOwnerActionPerformed (java.awt.event.ActionEvent evt)
{
String str=txtOwner.getText();
if(str.equals(""))
```

```
JOptionPane.showMessageDialog(null,"Enter the       Owner Name & Port");
else
{
String [] tmp=str.split("/");
String onr=tmp[0];
String port=tmp[1];
userController.sendREQ(onr,port,userName,this.port    );
}
}
```

3.5 Towards Third-Party Auditor

The user does not have sufficient time, resources to perform storage correctness verification, our scheme can be extended to third party auditor where users can safely delegate integrity checking tasks[4]. To give more security for user data introduce an effective TPA, auditing process should bring new vulnerabilities towards user data. TPA does not learn user's data content through the delegate data auditing. The new design is based on observation of linear property of the parity vector blinding process. Blinding process is to protect the secret key P against the cloud servers. If we blind data vector before file distribution encoding, then we can achieve storage verification tasks can be successfully delegated to TPA.

IV. Performance Evaluation

The performances for the proposed scheme storage auditing technique mainly focus on cost of file distribution preparation and token generation scheme. Our security analysis focuses on adversary model and efficiency can be increased based on token pre-computation and file distribution. Our experiment system consists of Intel Core 2 Processor at 1.86 GHz, 7200 RPM Western Digital 250GB and 2048 MB of RAM. Algorithms can be implemented using open-source erasure code because open-source is most suitable for real-time applications.

4.1 File Distribution Preparation

File distribution preparation scheme includes the generation of parity blinding process to protect secret key against the adversary model. Consider two different sets of parameters (m, k) data vector and parity vectors Reed-Solomon encoding, which works over GF(2^{16}).

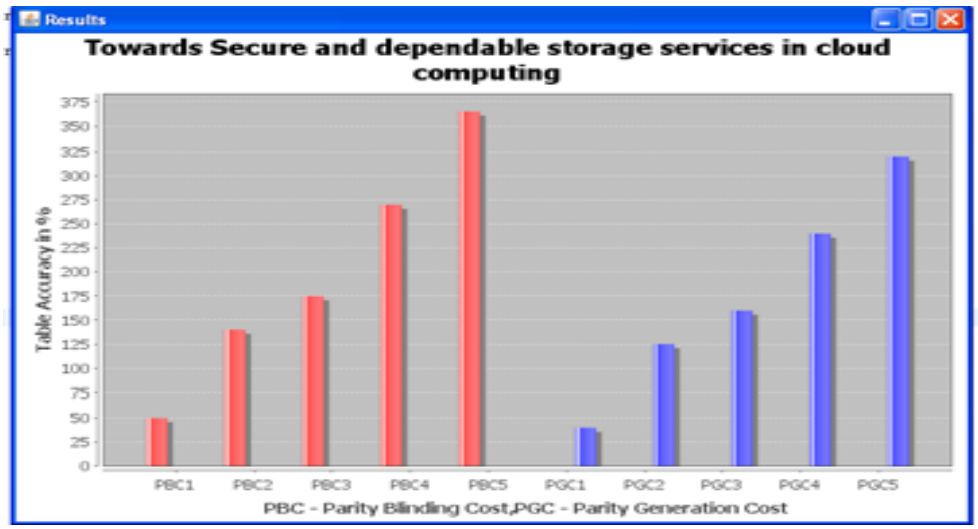

Fig. 4.1 : Performance comparison between two parameters (m, k) Setting for 1GB file distribution preparation

4.2 Challenge Token Pre-Computation

In our scheme, number of t verification tokens is fixed which can be determined before file distribution. For example, when t is selected as 7300 and 14600 data files can be verified every day for next 40 years and 60 years, respectively. Instead of directly computing each tokens, our scheme uses Horner algorithm to calculate tokens from back and achieves faster performance. For security analysis, Where, PBC-Parity blinding cost and PGC- parity generation cost. We consider parameter r=460 for token computation. Our implementation scheme

shows that average cost for token pre-computation is 0.4ms. This scheme is faster than the hash function[13]. To verify encoded data distribution over 14 servers, total cost for token pre-computation is no more than 1 and 1.5 minutes, for next 40 and 60 years.

Verify daily next 40years	(m, k)= (10, 4)	(m, k) = (10, 6)	(m, k) = (10, 8)	(m, k) = (14, 8)
Storage overhead (KB)	399.22	456.25	513.28	627.34
Computation overhead	82.79	94.62	106.45	130.10
Verify daily next 60 years	(m, k)= (10, 4)	(m, k) = (10, 6)	(m, k) = (10, 8)	(m, k) = (14, 8)
Storage overhead (KB)	598.83	684.37	769.92	941.01
Computation overhead	124.19	141.93	159.67	195.15

TABLE 4.2: The storage and computation cost for token pre-computation.

V. Conclusion

In our paper, we find the problem for data security in cloud data storage in distributed system. To achieve the correctness of cloud data availability and integrity and dependable cloud services for users, we propose a flexible distributed storage integrity auditing mechanism with explicit dynamic data operations, including block insert, update, append and delete. We use erasure-correcting code to achieve data redundancy and data dependability. By using homomorphic tokens with distributed erasure- coded data, our scheme achieves storage correctness of data and fast data error localization, i.e., whenever data has been corrupted, we can easily identify misbehaving server(s). Data owner does not have sufficient time, computation resources to audit the data file and to reduce the burden of user TPA (Third Party Auditor) can be introduced. Using TPA users can safely audit the integrity checking tasks and reduces the risk to use cloud storage services. Our experimental results are highly efficient and resilient against byzantine failure, server colluding attacks and malicious data modification attacks.

Acknowledgement

This work has supported in part by US National Science Foundation under grant CNS-0626601, CNS-0831963, CNS-0831628 and CNS-0716306.

References

[1]. Amazon.com, "Amazon web services (aws)," at http://aws.amazon.com/, 2009.
[2]. Q. Wang, C. Wang, W. Lou, and K.. Ren, "Ensuring data storage security in cloud computing," in Proc. Of IWQoS'09, July 2009, pp.1–9.
[3]. Q. Wang, C. Wang, W. Lou, and K. Ren "Privacy-preserving public auditing for storage security in cloud computing," in Proc.of IEEE INFOCOM'10, San Diego, CA, USA, March 2010.
[4]. Sun Microsystems, Inc., "Building customer trust in cloud computing with security," at https://www.sun. com/offers/details/sun transparency.xml, November 2009.
[5]. C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for storage security in cloud computing," in Proc.of IEEE INFOCOM'10, San Diego, CA, USA, March 2010.
[6]. A. Juels, K. D. Bowers, and A. Opera, " A high integrity and availability layer for cloud data storage. In lroc. Of CCS'09, 2009.
[7]. J. Hen, M. Rei, and G. Gan, " Verifying distributed correctness of coded data", in proc. Of 26th ACM of distributed computing, 2007, pp.139-146.
[8]. G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. of SecureComm'08, 2008, pp. 1–10.
[9]. Q. Wang, C. Wang, K. Ren and W. Lou, " Privacy-preserving public auditing for secure cloud storage". 2011
[10]. K. Ren, Q. Wang, Y. Zhang and W. Lou "Dependable and secure data storage with dynamic integrity assurance", IEEE INFOCOM 2011
[11]. 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, 2010.
[12]. E. L. Miller and T. Schwarz, " Check, store, and forget: using algebraic signatures to check remotely stored data", in proc of ICDCS'06, 2006.