ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Volume 1, Issue 10, December 2013

To Provide Security & Integrity for Storage Services in Cloud Computing

[1]vinothlakshmi.S

Assistant Professor, Dept of IT, Bharath Unversity, Chennai, TamilNadu, India

ABSTRACT: we propose in this paper a flexible distributed storage integrity auditing mechanism, utilizing the homomorphism token-pre computation and distributed erasure-coded data. The proposed configure users to audit the cloud storage with very lightweight communication and computation cost. The result auditing not only provides secure cloud storage correctness assurance, but also at the same time achieves fast data error localization. Considering the cloud data are motion in nature, the proposed design further supports secure and efficient dynamic operations on outside sourced data, including block modifying, deleting, and append. Analysis shows the proposed scheme is highly efficient and flexible against complex failure, malicious data modification attack, and even server colluding attacks. It includes file splitting process, which means storing of data into multiple servers .Extensive security and performance analysis shows the proposed schemes are provably secure and highly efficient. Here the data or applications are processed in the cloud are taken in the form of tokens or small blocks of data.

I. INTRODUCTION

Cloud Computing has been envisioned as the next-generation architecture of IT Enterprise. Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. 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. Meanwhile, the increasing network bandwidth and reliable yet flexible network connections make it even possible that clients can now subscribe high quality services from data and software that reside solely on remote data centers. One of the biggest concerns with cloud data storage is that of data integrity verification at untrusted servers. For example, the storage service provider, which experiences complex failures occasionally, may decide to hide the data errors from the clients for the benefit of their own. Consider the large size of the outsourced electronic data and the client's constrained resource capability, the core of the problem can be generalized as how can the client find an efficient way to perform periodical integrity verifications without the local copy of data files. In order to achieve the assurances of cloud data integrity and availability and enforce the quality of cloud storage service, efficient methods that enable on-demand data correctness verification on behalf of cloud users have to be designed. In this paper, we propose an effective and flexible distributed storage verification scheme with explicit dynamic data support to ensure the correctness and availability 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 against Byzantine servers, where a storage server may fail in arbitrary ways. we further explore the algebraic property of our token computation and erasure-coded data, and demonstrate how to efficiently support dynamic operation on data blocks, while maintaining the same level of storage correctness assurance. we also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third-party auditors and be worry-free to use the cloud storage services.
Our contribution can be summarized as the following three aspects:
1) Compared to many of its predecessors, which only provide binary results about the storage status across the distributed servers, the proposed scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server(s).

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

3) The experiment results demonstrate the proposed scheme is highly efficient. Extensive security analysis shows our scheme is resilient against Byzantine failure, malicious data

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Volume 1, Issue 10, December 2013

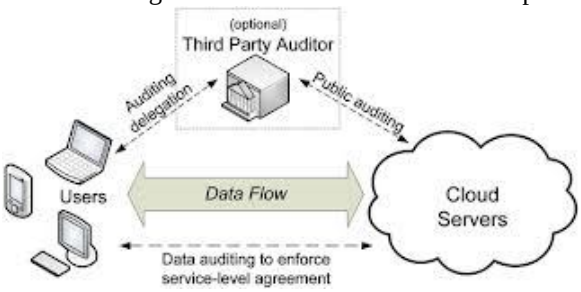
II. PROBLEM STATEMENT

2.1 System Model

User: an entity, who has data to be stored in the cloud and relies on the cloud for data storage and computation, can be either enterprise or individual customers.

Cloud Server (CS): an entity, which is managed by *cloud service provider* (CSP) to provide data storage service and has significant storage space and computation resources (we will not differentiate CS and CSP hereafter.).

Third Party Auditor (TPA): an optional 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.



2.3 Design Goals:

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.

III. MODULES

1) Token pre-computation:

Client files are stored in the cloud servers For Data integrity problem, we should split the file into many. In order to store the files into multiple servers, we should split the given files. Based on the splitted file size, files are stored into respective servers

2)Encrypt Splitted Files: Since client files are stored in the cloud server, they have lesser security options. To overcome these security we have implemented the crypto process. For crypto process we use DES algorithm for the encryption and decryption process using DES algorithm files are converted as crypto files.

3) File Replication in Cloud Server: It is nothing but we are organizing the files that is being stored as splitted files into the multiple servers. Here,We are keeping the Single file content into multiple servers.

4) File Retrieval: In order to decrypt and process the cloud files, given files should be retrieved. This process consist of file retrieval that is being stored from multiple servers.

5) Client Decryption:In order to get view the original content of the files, the encrypted files should be decrypted. Each and every encrypted files should be decrypted. Decryption process is done by DES Algorithm.

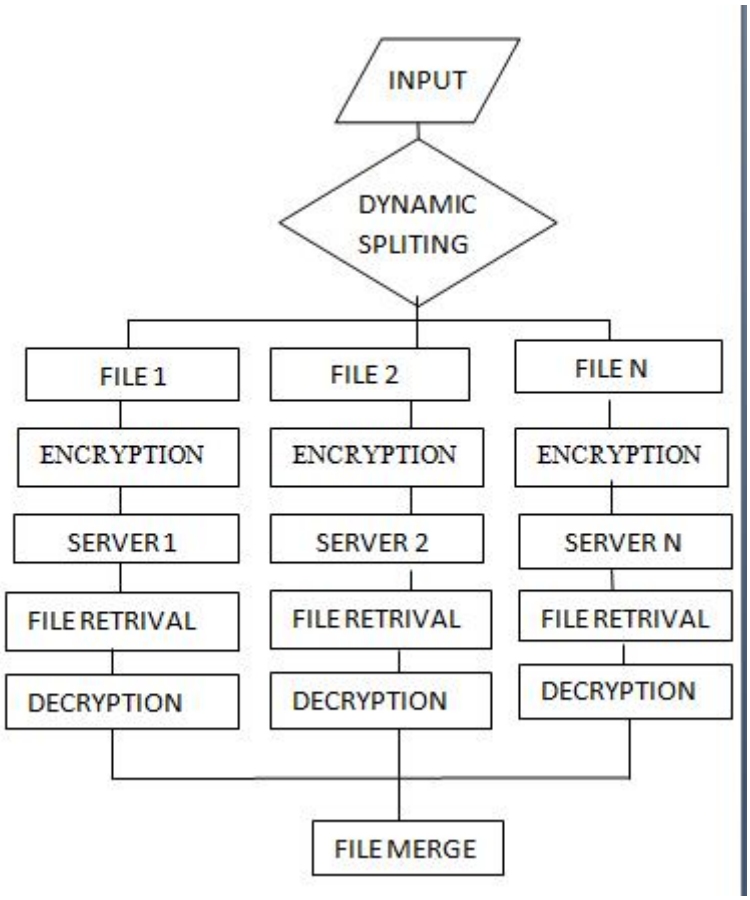
6)Raw File Merge: In order to get full content of the given data and the splitted files should be merged.By means of merging the files ,Data integrity is achieved.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Volume 1, Issue 10, December 2013



IV ENSURING CLOUD DATA STORAGE

3.1 File Distribution Preparation: It is well known that erasure-correcting code may be used to tolerate multiple failures in distributed storage systems. In cloud data storage, we rely on this technique to disperse the data file F redundantly across a set of n = m+ k distributed servers. An (m, k) Reed-Solomon erasure-correcting code is used to create k redundancy parity vectors from m data vectors in such a way that the original m data vectors can be reconstructed from any m out of the m+k data and parity vectors. By placing each of the m+k vectors on a different server, the original data file can survive the failure of any k of the m+k servers without any data loss, with a space overhead of k/m. For support of efficient sequential I/O to the original file, our file layout is systematic, i.e., the unmodified m data file vectors together with k parity vectors is distributed across m + k different servers.

3.2 Challenge Token Pre-computation: In order to achieve assurance of data storage correctness and data error localization simultaneously, our scheme entirely relies on the pre-computed verification tokens. The main idea is as follows: before file distribution the user pre-computes a certain number of short verification tokens on individual vector G(j) (j ∈{1, . . . , n}), each token covering 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. Upon receiving challenge, 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 secret matrix P

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Volume 1, Issue 10, December 2013

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 KPRP and challenge key kchal;
6: **for** vector G(j), j ← 1, n **do**
7: **for** round i← 1, t **do**
8: Derive _i = fkchal (i) and k(i) prp from KPRP .
9: Compute v(j) i = Pr q=1 _q i G(j)[_k(i) prp (q)]
10: **end for**
11: Store all the vi's locally.
12: **end procedure**
Suppose the user wants to challenge the cloud servers t times to ensure the correctness of data storage. Then, he must pre-compute t verification tokens for each G(j) (j ∈{1, . . . , n}), using a PRF f(·), a PRP _(·), a challenge key kchal and a master permutation key KPRP .
Specifically, to generate the ith token for server j, the user acts as follows:
1) Derive a random challenge value _i of GF(2p) by _i = fkchal (i) and a permutation key k(i) prp based on KPRP .
2) Compute the set of r randomly-chosen indices:
{Iq ∈1, ..., l]|1 ≤ q ≤ r},where Iq = _k (i) prp (q).
3) Calculate the token as:
v(j) i =Xr q=1_q i *G(j)[Iq],where G(j)[Iq] = g(j)Iq
. The details of token generation are shown in Algorithm:
1) Once all tokens are computed, the final step before
file distribution is to blind each parity block g(j)i in (G(m+1), . . . ,G(n)) by g(j) i ← g(j) i + fkj (sij), i ∈1, . . . , l},
where kj is the secret key for parity vector G(j) (j ∈m + 1, . . . , n}). This is for protection of the secret matrix P. We will discuss the necessity of using blinded parities in detail in Section 5.2. After blinding the parity information, the user disperses all the n encoded vectors G(j) (j ∈1, . . . , n}) across the cloud servers S1, S2, . . . , Sn.

3.3 Correctness Verification and Error Localization:
Algorithm 2 Correctness Verification and Error Localization
1: **procedure** CHALLENGE(i)
2: Recompute _i = fkchal (i) and k(i)prp from KPRP ;
3: Send {_i, k(i)prp} to all the cloud servers;
4: Receive from servers:
{R(j)i =Pr q=1 _q i *G(j)[_k(i)prp (q)]|1 ≤ j ≤ n}
5: **for** (j ← m + 1, n) **do**
6: R(j) ← R(j)− Prq=1 fkj (sIq,j)·_q i , 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(j) i ! =v(j) i) **then**
13: **return** server j is misbehaving.
14: **end if**
15: **end for**
16: **end if**
17: **end procedure**

Specifically, the procedure of the i-th challenge response for a cross-check over the n servers is described as follows:

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Volume 1, Issue 10, December 2013

1) The user reveals the _i as well as the i-th permutation key k(i) prp to each servers.
2) The server storing vector G(j) (j ∈ {1, . . . , n}) aggregates those r rows specified by index k(i) prp into a linear combination

R(j)i =Xrq=1_qi G(j)[_k(i)prp(q)]. and send back R(j)i (j ∈{1, . . . , n}).

3.4 File Retrieval and Error Recovery
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**

IV. CONCLUSION

In this paper, we investigate the problem of data securitynin 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. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s). Considering the time, computation resources, and even the related online burden of users, we also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third-party auditors and be worry-free to use the cloud storage services. Through detailed security and extensive experiment results, we show that our scheme is highly efficient and resilient to complex failure, malicious data modification attack, and even server colluding attacks.

REFERENCES

1. K. Ren, C. Wang, and Q. Wang, “Security Challenges for the Public Cloud,” IEEE Internet Computing, vol. 16, no. 1, pp. 69-73,2012.
2. Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, “Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing,” IEEE Trans. Parallel and Distributed Systems, vol. 22,no. 5, pp. 847-859, 2011.
3. C. Wang, S.S.M. Chow, Q. Wang, K. Ren, and W. Lou, “Privacy-Preserving Public Auditing for Secure Cloud Storage,” IEEE Trans. Computers, preprint, 2012, doi:10.1109/TC.2011.245.