ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 3, March 2013

Data Storage Security in Cloud Computing

Manoj Kokane[1], Premkumar Jain[2], Poonam Sarangdhar[3]

Government College of Engineering and Research, Awasari, Pune, India[1, 2, 3]

Abstract: Cloud computing is an emerging computing model in which resources of the computing communications are provided as services over the Internet. This paper proposed some services for data security and access control when users outsource sensitive data for sharing on cloud servers. This paper addresses this challenging open issue by, on one hand, defining and enforcing access policies based on data attributes, and, on the other hand, allowing the data owner to assign most of the computation tasks involved in fine grained data access control to un-trusted cloud servers without disclosing the underlying data contents. Extensive analysis shows that our proposed scheme is highly efficient and provably secures under existing security models. In Order to address this new problem and further achieve a secure and dependable cloud storage service, we propose in this paper a flexible distributed storage integrity auditing mechanism, utilizing the homomorphic token and distributed coded data. By third party auditing in this system, improves the availability and reliability of users data. This paper effectively supports dynamic data operations. As system is distributed, it is very essential to locate the misbehaving server so as that the user can access his sensitive information without any changes in it. This system also works against server attack and data crashes effectively.

Keywords: Homomorphic tokens, Third party auditing, SHA1

I. INTRODUCTION

Cloud computing, to put it simply, means internet computing. The internet is commonly visualized as clouds; hence the term "cloud computing" for computation done through the internet. With cloud computing users can access database resources via the internet from anywhere, for as long as they need, without worrying about any maintenance or management of actual resources. Besides, databases in cloud are very dynamic and scalable. Cloud Computing is unlike grid computing, utility computing, or autonomic computing. In fact, it is a very independent platform in terms of computing. The best example of cloud computing is Google apps where any application can be accessed using a browser and it can be deployed on thousands of computer through the internet. It also provides facilities for users to develop, deploy and manage their applications on the cloud, which entails virtualization of resources that maintains and manages itself.

Our proposed scheme enables the data owner to delegate tasks of data file re-encryption and user secret key update to cloud servers without disclosing data contents or user access privilege information. We achieve this goal by exploiting and uniquely combining techniques and algorithms (Correctness Verification and Error Localization, traditional replication-based file distribution, adding random perturbations).

In this paper, we propose an effective and flexible distributed scheme with explicit dynamic data support to ensure the correctness 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. This construction drastically reduces the communication and storage overhead as compared to the traditional replication-based file distribution techniques. By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the storage correctness insurance as well as data error localization: whenever data corruption has been detected during the storage correctness verification, our scheme can almost guarantee the simultaneous localization of data errors i.e. the identification of the misbehaving server(s).

Our work is among the first few ones in this field to consider distributed data storage in Cloud Computing. 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 state across the distributed servers, the challenge-response protocol in our work further provides the localization of data error.

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

3) Extensive security and performance analysis shows that the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

The rest of the paper is organized as follows. Section II introduces the system model, adversary model, our design goal and notations. Then we provide the detailed description of our scheme in Section III and IV. Section V which overviews the related work. Finally, Section VI gives the concluding remark of the whole paper.

II. PROBLEM STATEMENT

A. System Model

Representative network architecture for cloud data storage is illustrated in Figure 1.

Three different network entities can be identified as follows:

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 3, March 2013

• User: users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations.
• Cloud Service Provider (CSP): a CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates liv Cloud Computing systems.
• 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 Implementation of TPA is one of the main goal of this paper.

In cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, cooperated and distributed manner. Data redundancy 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. Thereafter, for application purposes, the user interacts with the cloud servers via CSP to access or retrieve his data. In some cases the user may need to perform block level operations on his data. As users no longer possess their data locally, it is of critical importance to assure users that their data are being corsrectly stored and maintained. In case that users do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the tasks to an optional trusted TPA of their respective choices. In our model, we assume that the point-to-point communication channels between each cloud server and the user is authenticated and reliable, which can be achieved in practice with little overhead. Note that we don't address the issue of data privacy in this paper, as in Cloud Computing, data privacy is orthogonal to the problem we study here.

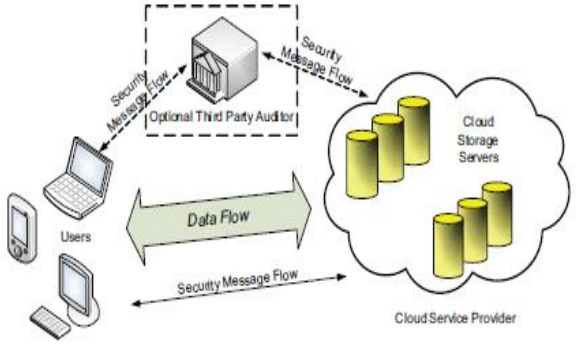

Figure 1: Cloud Data Storage Architecture

B. Adversary Model

Security threats faced by cloud data storage can come from two different sources. On the one hand, a CSP can be self-interested, untrusted and possibly malicious. Not only does it desire to move data that has not been or is rarely accessed to a lower tier of storage than agreed for monetary reasons, but it may also attempt to hide a data loss incident due to management errors, Byzantine failures and so on. On the other hand, there may also exist an economically motivated adversary, who has the capability to compromise a number of cloud data storage servers in different time intervals and subsequently is able to modify or delete user's data while remaining undetected by CSPs for a certain period. Specifically, we consider two types of adversary with different levels of capability in this paper:

Weak Adversary: The adversary is interested in corrupting the user's data files stored on individual servers. Once a server is comprised, an adversary 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.

Strong Adversary: This is the worst case scenario, in which we assume that 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 the case where all servers are colluding together to hide a data loss or corruption incident.

C. 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.

D. Notation and Preliminaries

- **F** – the data file to be stored. We assume that **F** can be denoted as a matrix of m equal-sized data vectors, each consisting of l blocks. Data blocks are all well represented as elements in Galois Field $GF(2^p)$ for $p = 8$ or 16.
- **A** – The dispersal matrix used for Reed-Solomon coding.
- **G** – The encoded file matrix, which includes a set of $n = m + k$ vectors, each consisting of l blocks.
- $f_{key}(\cdot)$ – pseudorandom function (PRF), which is defined as $f : \{0,1\}^* \times key \rightarrow GF(2^p)$.
- $\phi_{key}(\cdot)$ – pseudorandom permutation (PRP), which is defined as $\phi : \{0,1\}^{\log_2(l)} \times key \rightarrow \{0,1\}^{\log_2(l)}$.
- ver – a version number bound with the index for individual blocks, which records the times the block has been modified. Initially we assume ver is 0 for all data blocks.
- s_{ij}^{ver} – the seed for PRF, which depends on the file name, block index i, the server position j as well as the optional block version number ver.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 3, March 2013

III. ENSURING CLOUD DATA STORAGE

In cloud data storage system, users store their data in the cloud and no longer possess the data locally. Thus, the correctness 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 theory that is needed in our scheme for file distribution across cloud servers. Then, the homomorphic token is introduced. 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. 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. A (m + k, 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.

B. 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 code word determined by secret matrix P. Suppose the user wants to challenge the cloud server's 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 k_{chal} and a master permutation key K_{PRP} . After token generation, the user has the choice of either keeping the pre-computed tokens locally or storing them in encrypted form on the cloud servers. In our case here, the user stores them locally to obviate the need for encryption and lower the bandwidth overhead during dynamic data operation which will be discussed shortly. The details of token generation are shown in Algorithm 1.

Algorithm 1 Token Pre-computation

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 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_is locally.
13: **end procedure**

Once all tokens are computed, the final step before file distribution is to blind each parity block $g_i^{(j)}$ in ($G^{(m+1)}$, . . . ,$G^{(n)}$) by

$$g_i^{(j)} \leftarrow g_i^{(j)} + f_{kj}(s_{ij}), i \in \{1, \ldots, l\},$$

where k_j is the secret key for parity vector $G^{(j)}$ (j ∈ {m +1, . . . , n}). 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.

C. Correctness Verification and Error Localization

Error localization is a key prerequisite for eliminating errors in storage systems. However, many previous schemes do not explicitly consider the problem of data error localization, thus only provide binary results for the

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 3, March 2013

storage verification. Our scheme outperforms those by integrating the correctness verification and error localization in our challenge-response protocol: the response values from servers for each challenge not only determine the correctness of the distributed storage, but also contain information to locate potential data error(s).

D. 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. However, by choosing system parameters (e.g., r, l, t) appropriately and conducting enough times of verification, we can guarantee the successful file retrieval with high probability.

Algorithm 2 Correctness Verification and Error Localization

procedure CHALLENGE(i)
  Recompute $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP};
  Send $\{\alpha_i, k_{prp}^{(i)}\}$ to all the cloud servers;
  Receive from servers:
  $\{R_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)] | 1 \le j \le n\}$
  for $(j \leftarrow m+1, n)$ **do**
    $R^{(j)} \leftarrow R^{(j)} - \sum_{q=1}^{r} f_{k_j}(s_{I_q,j}) \cdot \alpha_i^q, I_q = \phi_{k_{prp}^{(i)}}(q)$
  end for
  if $((R_i^{(1)}, \ldots, R_i^{(m)}) \cdot \mathrm{P} == (R_i^{(m+1)}, \ldots, R_i^{(n)}))$ **then**
    Accept and ready for the next challenge.
  else
    for $(j \leftarrow 1, n)$ **do**
      if $(R_i^{(j)} != v_i^{(j)})$ **then**
        return server j is misbehaving.
      end if
    end for
  end if
end procedure

Algorithm 3 Error Recovery

procedure
  % Assume the block corruptions have been detected among
  % the specified r rows;
  % Assume $s \le k$ servers have been identified misbehaving
  Download r rows of blocks from servers;
  Treat s servers as erasures and recover the blocks.
  Resend the recovered blocks to corresponding servers.
end procedure

On the other hand, whenever the data corruption is detected, the comparison of pre-computed tokens and received response values can guarantee the identification of misbehaving server(s), again with high probability, which will be discussed shortly. Therefore, 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, shown in Algorithm 3, as long as there are at most k misbehaving servers are identified. The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.

IV. TOWARDS THIRD PARTY AUDITING

As discussed in our architecture, in case 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. However, as pointed out by the recent work [27], [28], to 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. Now 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. Recall that the reason of blinding process is for protection of the secret matrix P against cloud servers. However, this can be achieved either by blinding the parity vector or by blinding the data vector (we assume k < m). Thus, if we blind data vector before file distribution encoding, then the storage verification task can be successful delegated to third party auditing in a privacy-preserving manner.

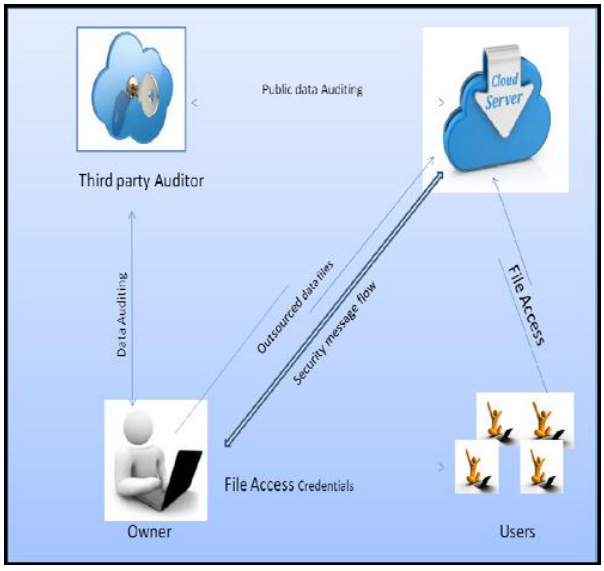

Figure 2: Towards Public Auditing

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 3, March 2013

V. RELATED WORK

Jules [2] described a formal "proof of retrievability" (POR) model for ensuring the remote data integrity. Their scheme combines spot-checking and error correcting code to ensure both possession and retrievability of files on archive service systems. Shacham [3] built on this model and constructed a random linear function based homomorphic authenticator which enables unlimited number of challenges and requires less communication overhead due to its usage of relatively small size of BLS signature. In their subsequent work, Ateniese [4] described a PDP scheme that uses only symmetric key based cryptography. This method has lower-overhead than their previous scheme and allows for block updates, deletions and appends to the stored file, which has also been supported in our work. However, their scheme focuses on single server scenario and does not provide data availability guarantee against server failures, leaving both the distributed scenario and data error recovery issue unexplored. The explicit support of data dynamics has further been studied in the two recent works [5] and [6]. The incremental cryptography work done by Bellare [10] also provides a set of cryptographic building blocks such as hash, MAC, and signature functions that may be employed for storage integrity verification while supporting dynamic operations on data. However, this branch of work falls into the traditional data integrity protection mechanism, where local copy of data has to be maintained for the verification. It is not yet clear how the work can be adapted to cloud storage scenario where users no longer have the data at local sites but still need to ensure the storage correctness efficiently in the cloud.

In other related work, Curtmola [9] aimed to ensure data possession of multiple replicas across the distributed storage system. They extended the PDP scheme to cover multiple replicas without encoding each replica separately, providing guarantee that multiple copies of data are actually maintained. Very recently, C. Wang [8] gave a study on many existing solutions on remote data integrity checking, and discussed their pros and cons under different design scenarios of secure cloud storage services.

Portions of the work presented in this paper have previously appeared as an extended abstract in [7]. We have revised the article a lot and add more technical details as compared to [7]. The primary improvements are as follows: Firstly, we provide the protocol extension for privacy-preserving third-party auditing, and discuss the application scenarios for cloud storage service. Secondly, we add correctness analysis of proposed storage verification design. Thirdly, we completely redo all the experiments in our performance evaluation part, which achieves significantly improved result as compared to [7]. We also add detailed discussion on the strength of our bounded usage for protocol verifications and its comparison with state-of-the-art.

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. 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 Byzantine failure, malicious data modification attack, and even server colluding attacks.

REFERENCES

[1] S.Sajithabanu and Dr.E.George Prakash Raj, "Data Storage Security in Cloud" *IJCST* Vol. 2, Issue 4, Oct. - Dec. 2011

[2] A. Jules and J. Burton S. Kaliski, "Pors: Proofs of retrievability for large files," in *Proc. of CCS'07*, Alexandria, VA, October 2007, pp. 584–597.

[3] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. of Asiacrypt'08, volume 5350 of LNCS*, 2008, pp. 90–107.

[4] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. of Secure Comm'08*, 2008, pp. 1–10.

[5] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. of ESORICS'09, volume 5789 of LNCS*. Springer-Verlag, Sep. 2009, pp. 355–370.

[6] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. of CCS'09*, 2009, pp. 213–222.

[7] 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.

[8] 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.

[9] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "Mr-pdp: Multiple-replica provable data possession," in *Proc. of ICDCS'08*. IEEE Computer Society, 2008, pp. 411–420.

[10] M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental cryptography: The case of hashing and signing," in *Proc. Of CRYPTO'94, volume 839 of LNCS*. Springer-Verlag, 1994, pp. 216–233.

[11] Amazon.com, "Amazon Web Services (AWS)," Online at http://aws.amazon.com, 2008.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 3, March 2013

BIOGRAPHY

MANOJ B. KOKANE
Government College of Engineering and Reasearch, Awasari, Pune, India.
B.E.(Computer Science and Engineering)

PREMKUMAR S. JAIN
Government College of Engineering and Reasearch, Awasari, Pune, India.
B.E.(Computer Science and Engineering)

POONAM S. SARANGDHAR
Government College of Engineering and Reasearch, Awasari, Pune, India.
B.E.(Computer Science and Engineering