IJCOT -Special Issue– The Malla Reddy National Conference on Information System and Knowledge Engineering (MRNC-ISKE 2013) - July 2013

Distributed Challenge Response Protocol for Error Localization

L.Vandana*1, Sukesh Manyam*2, P. Preeti Payal*3

Assistant Professor, Department of CSE, NNRGI, Ghatkesar, D.t: Ranga Reddy, A.P, India

Assistant Professor, Department of C.S.E, VCE, Bollikunta, D.t: Warangal, A.P, India

Associate Professor, Department of C.S.E, PPGCM, Ramanthpur, D.t: Hyderabad A.P, India

ABSTRACT

Cloud Computing has been envisioned as the next generation architecture of IT Enterprise. In contrast to traditional solutions, where the IT services are under proper physical, logical and personnel controls, Cloud Computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. This unique attribute, however, poses many new security challenges which have not been well understood.

In this article, we focus on cloud data storage security, which has always been an important aspect of quality of service. To ensure the correctness of users' data in the cloud, we propose an effective and flexible distributed scheme with two salient features, opposing to its predecessors. By utilizing the homomorphic token with distributed verification of erasure-coded data. This work studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, we consider the task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of client through the auditing of whether his data stored in the cloud is indeed intact, which can be important in achieving economies of scale for Cloud Computing.

I. INTRODUCTION

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. 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. Moving data into the cloud offers great convenience to users since they don't have to care about the complexities of direct hardware management. The pioneer of Cloud Computing vendors, Amazon Simple Storage Service (S3) and Amazon Elastic Compute Cloud (EC2) [1] are both well known examples. While these internet-based online services do provide huge amounts of storage space and customizable computing resources, this computing platform Shift, however, is eliminating the responsibility of local machines for data maintenance at the same time. As a result, users are at the mercy of their cloud service providers for the availability and integrity of their data. Recent downtime of Amazon's S3 is such an example [2]. From the perspective of data security, which has always been an important aspect of quality of service, Cloud Computing inevitably poses new challenging security threats for number of reasons. Firstly, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted due to the users' loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. Considering various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of verifying correctness of data storage in the cloud becomes even more challenging. Secondly,

Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion, deletion, modification, appending, reordering, etc. To ensure storage correctness under dynamic data update is hence of paramount importance. However, this dynamic feature also makes traditional integrity insurance techniques futile and entails new solutions. Last but not the least, the deployment of Cloud Computing is powered by data centers running in a simultaneous, cooperated and distributed manner. Individual user's data is redundantly stored in multiple physical locations to further reduce the data integrity threats. Therefore, distributed protocols for storage correctness assurance will be of most importance in achieving a robust and secure cloud data storage system in the real world. However, such important area remains to be fully explored in the literature. Recently, the importance of ensuring the remote data integrity has been highlighted by the following research works [3]–[7]. These techniques, while can be useful to ensure the storage correctness without having users possessing data, cannot address all the security threats in cloud data storage, since they are all focusing on single server scenario and most of them do not consider dynamic data operations. As an complementary approach, researchers have also proposed distributed protocols [8]–[10] for ensuring storage correctness across multiple servers or peers. Again, none of these distributed schemes is aware of dynamic data operations. As a result, their applicability in cloud data storage can be drastically limited.

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).

II.ARCHITECTURE

2.1 CLOUD MODEL

Treat a cloud for simplicity as a highly resourced, monolithic entity, and denote each entity relying on resources as a client. Denote the set of n clients of the entity in the cloud. In the model of cloud computing, clients are thin. They have limited local computation and storage, delegating as much as possible to a cloud provider. And they are not consistently on-line. They may deposit data in the cloud and go offline indefinitely. Consequently, a cloud provider assumes responsibility for processing data in the absence of its owners. Applications that operate over the data of multiple clients respect access-control policies. The client stores her data in the server without keeping a local copy. Hence, it is of critical importance that the client should be able to verify the integrity of the data stored in the remote untrusted server. If the server modifies any part of the client's data, the client should be able to detect it; furthermore, any third party verifier should also be able to detect it. In case a third party verifier verifies the integrity of the client's data, the data should be kept private against the third party verifier.

2.2 TECHNICAL PRELIMINARIES

Consider a cloud storage system in which there area client and an untrusted server. The client stores data in the server without keeping a local copy. Hence, it is of critical importance that the client should be able to verify the integrity of the data stored in the remote untrusted server. If the server modifies any part of the client's data, the client should be able to detect it; furthermore, any third party verifier should also be able to detect it. In case a third party verifier.

III. Problem Definition

At a high-level, a verifiable computation scheme is a two-party protocol in which a *client* chooses a function and then provides an encoding of the function and inputs to the function to a *worker*. The worker is expected to evaluate the function on the input and respond with the output. The client then verifies that the output provided by the worker is indeed the output of the function computed on the input provided

Mostly cloud data storage service involving three different entities, as illustrated in

Fig. 1: the cloud user, who has large amount of data files to be stored in the cloud; the cloud server (CS), which is managed by the cloud service provider (CSP) to provide data storage service and has significant storage space and computation resources; the third party auditor (TPA),who has expertise and capabilities that cloud users do not have and is trusted to assess the cloud storage service reliability on behalf of the user upon request. Users rely on the CS for cloud data storage and maintenance. Asusers no longer possess their data locally, it is to critical importance for users to ensure that their data are being correctly stored and maintained. To save the computation resource as well as the online burden potentially brought by the periodic storage correctness verification, cloud users may resort to TPA for ensuring the storage integrity of their outsourced data, while hoping to keep their data private from TPA.

IV. DESIGN GOALS

To enable privacy-preserving technique for cloud data storage under the aforementioned model, proposed protocol design should achieve the following guarantees.

1) Public Verifiability: To allow TPA to verify the correctness of the cloud data on demand without retrieving a copy of the whole data or introducing additional online burden to the cloud users. 2) Storage correctness: To ensure that there exists no cheating cloud server that can pass the TPA's audit without indeed storing users' data intact.

3) Privacy-preserving: To ensure that the TPA cannot derive users' data content from the information collected during the verifying process.

4) Dynamic data operation support: To allow the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. The design should be as efficient as possible so as to ensure the seamless integration of public verifiability and dynamic data operation support

5) Block less verification: No challenged file blocks should be retrieved by the verifier (e.g., TPA) during verification process for both efficiency and security concerns.

6) Stateless verification: To eliminate the need for state information maintenance at the verifier side between audits throughout the long term of data storage in the cloud storage device.

V. 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.

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.

Algorithm 1 Token Pre-computation

procedure
  Choose parameters l, n and function f, ϕ;
  Choose the number t of tokens;
  Choose the number r of indices per verification;
  Generate master key K_{prp} and challenge k_{chal};
  for vector $G^{(j)}$, $j \leftarrow 1, n$ **do**
    for round $i \leftarrow 1, t$ **do**
      Derive $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP}.
      Compute $v_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)]$
    end for
  end for
  Store all the v_is locally.
end procedure

B. Challenge Token Precomputation

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 $\in$ {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.

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

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 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

VI Backgrounds

We summarize Yao's protocol for two-party private computation. For more details, we refer the interested reader to Lindell and Pinkas' excellent description. We assume two parties, Alice and Bob, wish to compute a function *F* over their private inputs *a* and *b*. For simplicity, we focus on polynomial-time deterministic functions, but the generalization to stochastic functions is straightforward.

The order of the ciphertexts is randomly permuted to hide the structure of the circuit (i.e., we shuffle the ciphertexts, so that the first ciphertext does not necessarily encode the output for. In Yao's protocol, Alice transfers all of the ciphertexts to Bob, along with the wire values corresponding to the bit-level representation of her input. In other words, she transfers either $k0\ a$ if her input bit is 0 or $k1\ a$ if her input bit is 1. Since these are randomly chosen values, Bob learns nothing about Alice's input. Alice and Bob then engage in an oblivious transfer so that Bob can obtain the wire values corresponding to his inputs (e.g., $k0\ b$ or $k1\ b$). Bob learns exactly one value for each wire, and Alice learns nothing about his input. Bob can then use the wire values to recursively decrypt the gate ciphertexts, until he arrives at the final output wire values. When he transmits these to Alice, she can map them back to 0 or 1 values and hence obtain the result of the function computation.

VII. CONCLUSION

In this paper, we investigated the problem of data security in cloud data storage, which is essentially a distributed storage system. To ensure the correctness of users' data in cloud data storage, we proposed 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). Through detailed security and performance analysis, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks.

VIII REFERENCES

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

[2] N. Gohring, "Amazon's S3 down for several hours," Online at http://www.pcworld.com/businesscenter/article/142549/amazons s3 down for several hours.html, 2008.

[3] A. Juels and J. Burton S. Kaliski, "PORs: Proofs of Retrievability for Large Files," *Proc. of CCS '07*, pp. 584–597, 2007.

[4] H. Shacham and B. Waters, "Compact Proofs of Retrievability," *Proc. of Asiacrypt*.

[5] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Cryptology ePrint Archive, Report 2008/175, 2008, http://eprint.iacr.org/.

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," *Proc. Of CCS '07*, pp. 598–609, 2007.

[7] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and

Efficient Provable Data Possession," *Proc. of SecureComm '08*, pp. 1– 10, 2008.

[8] T. S. J. Schwarz and E. L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," *Proc.of ICDCS '06*, pp. 12–12, [9] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard,

"A Cooperative Internet Backup Scheme," *Proc. of the 2003 USENIX Annual Technical Conference (General Track)*, pp. 29–41,

[10] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Cryptology ePrint Archive, Report 2008/489, 2008, http://eprint.iacr.org/.