Data security in Cloud Computing

S D Choudhari
Department of information Technology
J L Chaturvedi,College of engineering
Nagpur,India

Dr S K Shrivastava
Director
SBITM
Betul,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 centres, 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 paper, we focus on cloud data storage security i.e. Data Verification, Tampering, Loss and Theft, 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, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server(s). Unlike most prior works, the new scheme further supports secure and efficient dynamic operations on data blocks. Extensive security and performance analysis shows that the proposed scheme is highly efficient and resilient against data tampering, loss and theft in cloud computing.

1. 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 centres.

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].

2. Data verification, Tampering, Loss and Theft in cloud computing

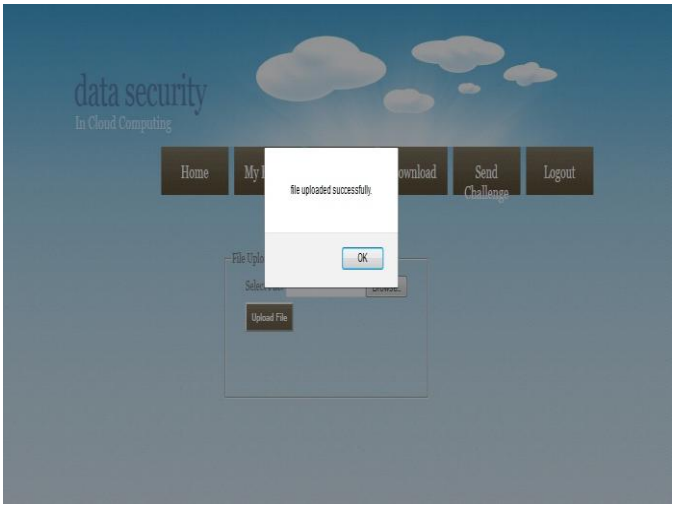
In cloud computing, data security has always been an important aspect, Cloud Computing inevitably poses new challenging security threats for number of reasons. Firstly, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted due to the users' loss control 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 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.

So when user uploads a file to cloud server, a success message is shown to user as follows:



After uploading, if user wants to download a file he can download as follows:

3. Solution

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.

To address these Data theft in cloud computing, our main scheme for ensuring cloud data storage is presented in this section. Our solution is divided into two parts as file distribution across cloud servers and then the homomorphic token is introduced.

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: 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 Kprp and challenge kchal;

6:$for\ vector\ G(j), j \leftarrow 1, n\ do$

7: for round i $\leftarrow$ 1, t do

8:$Derive\ \alpha i\ =\ fkchal(i)\ and\ k(i)prp\ from\ KPRP$.

9: $V_i(j) = \sum_{q=1}^{r} \alpha_i^q * G(j)\ [\varphi\ k(i)prp\ (q)]$

10: end for

11: end for

12: Store all the V_i locally.

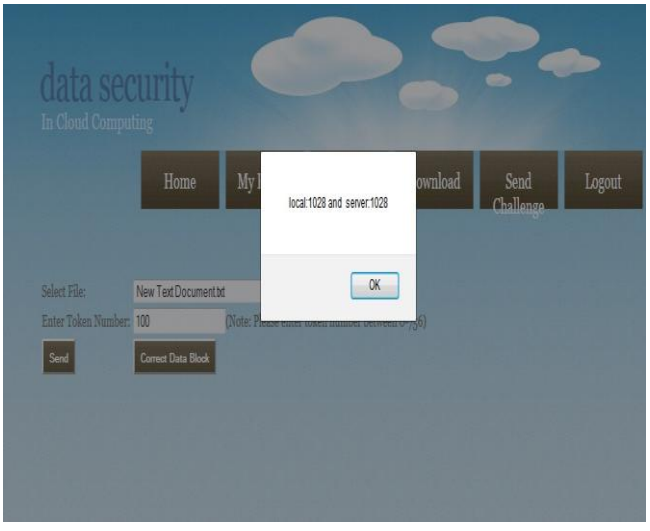
13: end procedure

This is explained as follows:

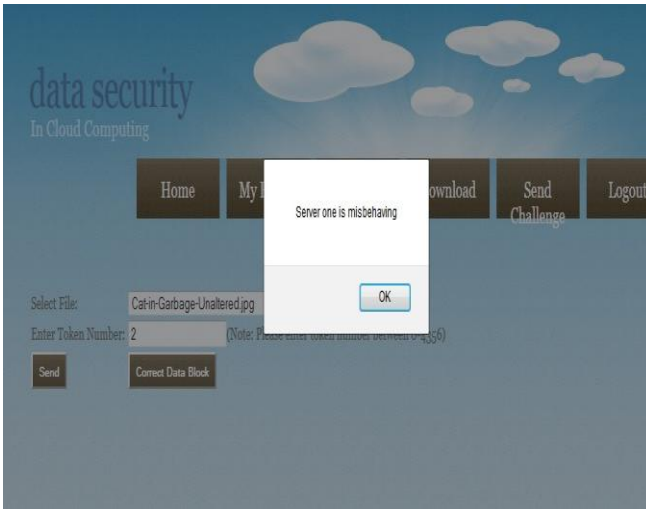
In order to achieve assurance of data storage correctness and data error localization simultaneously, our scheme entirely relies on the pre-computed verification tokens. The main idea is as follows: before file distribution the user pre-computes a certain number of short verification tokens on individual vector

Following screen shows, how user can avoid data tampering, loss and theft. As mentioned in algorithm, user can pre compute token prior to file upload. And when he has to check file version, he will just send token to server. If both the tokens as client and servers matches as follows then the file version on server is correct. Hence using token generation, we can avoid data theft.

Following screen shows token match if file version is not changed or data is not lost.



Following screen shows token mismatch if file version is changed or data is lost.



$G(j)$ (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.

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 φ(.), a challenge key $K_{chal}$and a master permutation key K_{PRP}. To generate the token for server j, the user acts as follows:

1) Derive a random challenge value α_i of GF (2^p)

By $\alpha_i = fk_{chal}^{(i)}$ and a permutation key k_{prp}^{i} based onK_{PRP}.

2) Compute the set of r randomly-chosen indices:

$\{I_{q \in [1,\ldots,l]} \mid 1 \le q \le r\}$, Where $I_q = \varphi k_{prp}^{i}$ (q).

3) Calculate the token as:

$$\underset{i}{V}(j) = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[I_q], \text{where } G^{(j)}[I_q] = g_{I_q}^{(j)}$$

4. Conclusion

In this way we have proposed and implemented one technique for security in cloud computing.

References

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

[2] [2] N. Gohring, "Amazon's S3 down for several hours," Online at http://www.pcworld.com/businesscenter/article/142549/ amazons s3down for several hours.html, 2008.

[3] [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 '08, Dec. 2008.

[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, 2006.

[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, 2003.

[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/.

[11] L. Carter and M. Wegman, "Universal Hash Functions," Journal of Computer and System Sciences, vol. 18, no. 2, pp. 143–154, 1979.

[12] J. Hendricks, G. Ganger, and M. Reiter, "Verifying Distributed Erasure-coded Data," Proc. 26th ACM Symposium on Principles of Distributed Computing, pp. 139–146, 2007.

[13] J. S. Plank and Y. Ding, "Note: Correction to the 1997 Tutorial on Reed-Solomon Coding," University of Tennessee, Tech. Rep. CS-03-504, 2003.

[14] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance," Proc. of IEEE INFOCOM, 2009.

[15] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession," Proc. of ICDCS '08, pp. 411–420,

2008.

[16] D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating Data Possession and Uncheatable Data Transfer," Cryptology ePrint Archive, Report 2006/150, 2006, http://eprint.iacr.org/.

[17] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest," Proc. 11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS '07), pp. 1–6, 2007.