Research Article Impact Factor: 0.621 ISSN: 2319-507X
Ashwini Raut, IJPRET, 2014; Volume 2 (9): 573-579 IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

UNASSAILABLE AND STREAMLINED OPERATIONS TO PROVIDE DATA PROTECTION IN CLOUD COMPUTING

ASHWINI G. RAUT[1], PROF. C. M. MANKAR[2]

1. ME (Fourth Semester), Computer Engineering, SSGMCE Shegaon, Sant Gadge Baba Amravati University.
2. B. Computer Science & Engineering, SSGMCE Shegaon, Sant Gadge Baba Amravati University,

Accepted Date: 27/02/2014 ; Published Date: 01/05/2014

Abstract: Cloud storage enables users to remotely store their data and enjoy the on-demand high quality cloud applications without the burden of local hardware and software management. Though the benefits are clear, such a service is also relinquishing user's physical possession of their outsourced data, which inevitably poses new security risks toward the correctness of the data in cloud. 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 erasure-coded data. It provides very lightweight communication and computation cost to user. The auditing result not only ensures strong cloud storage correctness guarantee, but also simultaneously achieves fast data error localization. Considering the cloud data are dynamic in nature, it supports secure and efficient dynamic operations on outsourced data. Analysis shows this scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server clouding attacks.

Keywords: Data integrity, dependable distributed storage, error localization, data dynamics, cloud computing.

Corresponding Author: MR. ASHWINI G. RAUT

Access Online On:

www.ijpret.com

How to Cite This Article:

Ashwini Raut, IJPRET, 2014; Volume 2 (9): 573-579

PAPER-QR CODE

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 (CSP) for the availability and integrity of their data [2], [3]. On the one hand, although the cloud infrastructures are much more powerful and reliable than personal computing devices, broad range of both internal and external threats for data integrity still exist. Examples of outages and data loss incidents of noteworthy cloud storage services appear from time to time [4], [5], [6], [7], [8]. On the other hand, since users may not retain a local copy of outsourced data, there exist various incentives for CSP to behave unfaithfully toward the cloud users regarding the status of their outsourced data. For example, to increase the profit margin by reducing cost, it is possible for CSP to discard rarely accessed data without being detected in a timely fashion [9]. Similarly, CSP may even attempt to hide data loss incidents so as to maintain a reputation [10], [11], [12]. Therefore, although outsourcing data into the cloud is economically attractive for the cost and complexity of long-term large-scale data storage, its lacking of offering strong assurance of data integrity and availability may impede its wide adoption by both enterprise and individual cloud users.

2. MATERIAL AND METHODS

2.1 Adversary Model

From user's perspective, the adversary model has to capture all kinds of threats toward his cloud data integrity. Because cloud data do not reside at user's local site but at CSP's address domain, these threats can come from two different sources: internal and external attacks. For internal attacks, 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. For external attacks, data integrity threats may come from outsiders who are beyond the

Control domain of CSP, for example, the economically motivated attackers. They may compromise a number of cloud data storage servers in different time intervals and subsequently be able to modify or delete users' data while remaining undetected by CSP. Therefore, we consider the adversary in our model has the following capabilities, which captures both external and internal threats toward the cloud data integrity. Specifically, the adversary is interested in continuously 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. This corresponds to the threats from external attacks. In the worst case scenario, the adversary can compromise all the 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 always be the first step to fast recover the storage errors and/or identifying potential threats of external attacks. 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 [13], chosen to preserve the homomorphic properties, which can be perfectly integrated with the verification of erasurecoded data [12], [14]. Subsequently, it is shown how to derive a challenge-response protocol for verifying the storage correctness as well as identifying misbehaving servers. The procedure for file retrieval and error recovery based on erasure-correcting code is also outlined. Finally, we describe how to extend our scheme to third party auditing with only slight modification of the main design.

2.2 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 ¼ m þ k distributed servers. An ðm; kÞ 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 þ k data and parity vectors. By placing each of the m þ k vectors on a different server, the original data file can survive the failure of any k of the m þ 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 þ k different servers. Challenge Token Precomputation In order to achieve assurance of data storage correctness and data error localization simultaneously, our scheme entirely relies on the precomputed verification tokens. The main idea is as follows: before file distribution the user precomputes a certain number of short verification tokens on individual vector Gðjþ ðj 2 f1; . . . ; ngþ, 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 precomputed 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 the secret matrix P.Correctness Verification and Error Localization Error localization is a key prerequisite for eliminating errors in storage systems. It is also of critical importance to identify potential threats from external attacks. However, many previous schemes [13], [14] do not explicitly consider the problem of data error localization, thus only providing binary results for the storage verification. Our scheme outperforms those by integrating the correctness verification and error localization (misbehaving server identification) 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).

2.3 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 ðe:g:; r; l; tÞ appropriately and conducting enough times of verification, we can guarantee the successful file retrieval with high probability. On the other hand, whenever the data corruption is detected, the comparison of precomputed 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 as long as the number of identified misbehaving servers is less than k. (otherwise, there is no way to recover the corrupted blocks due to lack of redundancy, even if we know the position of misbehaving servers.) The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.

4. RESULTS AND DISCUSSION

To ensure the security and dependability for cloud data storage under the aforementioned adversary model, I 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.

3. CONCLUSION AND FUTURE WORK

As private data moves online, the need to secure it properly becomes increasingly urgent. The good news is that the same forces concentrating data in enormous datacenters will also aid in using collective security expertise more effectively. Adding protections to a single cloud platform can immediately benefit hundreds of million users.

In future, this work can be extended to develop a more formal model for data protection:

1. Attribute Clustering:

Having computed the correlations for each pair of attributes, we use clustering to partition attributes into columns.

2. Attribute Partitioning:

Partitions attributes so that highly correlated attributes are in the same column. This is good for both utility and privacy.

REFERENCES

1. Amazon.com, "Amazon Web Services (AWS)," http://aws. amazon.com, 2009.
2. Sun Microsystems, Inc., "Building Customer Trust in Cloud Computing with Transparent security", https://www.sun.com/offers/details/sun_transparency.xml,Nov.2009.
3. K. Ren, C. Wang, and Q. Wang, "Security Challenges for the
4. Public Cloud," IEEE Internet Computing, vol. 16, no. 1, pp. 69-73,2012.
5. M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions,"
6. http://www.techcrunch.com/2006/12/28/gmail-disasterreportsof-mass-email-deletions, Dec. 2006.
7. J. Kincaid, "MediaMax/TheLinkup Closes Its Doors,"
8. http://www.techcrunch.com/2008/07/10/mediamaxthelinkup-closesits-doors, July 2008.
9. Amazon.com, "Amazon S3 Availability Event: July 20, 2008,"
10. http://status.aws.amazon.com/s3-20080720.html, July 2008.
11. S. Wilson, "Appengine Outage," http://www.cio-weblog.com/
12. 50226711/appengine_outage.php,June 2008.
13. B. Krebs, "Payment Processor Breach May Be Largest Ever,"
14. http://voices.washingtonpost.com/securityfix/2009/01/payment_processor_breach_may_b .html, Jan. 2009.
15. A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.
16. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS'07), pp. 598-609, Oct. 2007.
17. M.A. Shah, M. Baker, J.C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest," Proc. 11th USENIX Workshop Hot Topics in Operating Systems (HotOS '07), pp. 1-6, 2007.

18. M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents," Cryptology ePrint Archive, Report 2008/186, http://eprint.iacr.org, 2008.

19. T. Schwarz and E.L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage,"Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '06),pp. 12-12, 2006.Recovery," ACM Trans. Computer Systems, vol. 20, no. 4, pp. 398-461, 2002.

20. L. Carter and M. Wegman, "Universal Hash Functions," J.Computer and System Sciences, vol. 18, no. 2, pp. 143-154, 1979.

21. J. Hendricks, G. Ganger, and M. Reiter, "Verifying Distributed Erasure-Coded Data," Proc. 26th ACM Symp. Principles of Distributed Computing, pp. 139-146, 2007.