ABHIYANTRIKI
An International Journal of Engineering & Technology
(A Peer Reviewed & Indexed Journal)
Vol. 4, No. 5 (May, 2017) http://www.aijet.in/ eISSN: 2394-627X

Outsourced Proof of Data Retrievability and Recovery in Hybrid Cloud

Prof. Chandrashekar B S
Deptt. of Computer Science Engineering,
RNS Institute of Technology, Bengaluru, India

Pavithra B*
Deptt. of Computer Science Engineering,
RNS Institute of Technology, Bengaluru, India

Abstract

Cloud computing moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. In this work, we study the problem of ensuring the integrity of data storage in cloud computing. To reduce the computational cost at user side during the integrity verification of their data, the notion of public verifiability has been proposed. However, the challenge is that the computational burden is too huge for the users with resource-constrained devices to compute the public authentication tags of file blocks. To tackle the challenge, we propose OPDR, a new cloud storage scheme involving a cloud storage server and a cloud audit server, where the latter is assumed to be semi-honest. In particular, we consider the task of allowing the cloud audit server, on behalf of the cloud users, to pre-process the data before Uploading to the cloud storage server and later verifying the data integrity. OPDR outsources and offloads the heavy computation of the tag generation to the cloud audit server and eliminates the involvement of user in the auditing and in the pre-processing phases. Furthermore, we strengthen the proof of retrievability (PoR) model to support dynamic data operations, as well as ensure security against reset attacks launched by the cloud storage server in the upload phase.

Keywords*: Data Retrievability, Hybrid Cloud, Amazon EC2.*

****Author for correspondence*** pavithra.manjunatha@gmail.com

1. Introduction

CLOUD computing has been envisioned as the next generation architecture of the IT enterprise due to its long list of unprecedented advantages: on-demand self-service, ubiquitous network access, location-independent resource pooling, rapid resource elasticity, and usage-based pricing. In particular, 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. Although having appealing advantages as a promising service platform for the Internet, this new data storage paradigm in - cloud brings many challenging issues which have profound influence on the usability, reliability, scalability, security, and performance of the overall system. One of the biggest concerns with remote data storage is that of data integrity verification at untrusted servers. For instance, the storage service provider may decide to hide such data loss incidents as the Byzantine failure from the clients to maintain a reputation. What

is more serious is that for saving money and storage space the service provider might deliberately discard rarely accessed data files which belong to an ordinary client. Considering 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 verification without the local copy of data files.

In order to overcome this problem, many schemes have been proposed under different system and security models [1, 2, 3, 4, 5, 6, 7, 8, 9]. In all these works, great efforts have been made to design solutions that meet various requirements: high scheme efficiency, stateless verification, unbounded use of queries and retrievability of data, etc. According to the role of the verifier in the model, all the schemes available fall into two categories: private verifiability and public verifiability. Although achieving higher efficiency, schemes with private verifiability impose computational burden on clients. On the other hand, public verifiability alleviates clients from performing a lot of computation for ensuring the integrity of data storage. To be specific, clients are able to delegate a third party to perform the verification without devotion of their computation resources. In the cloud, the clients may crash unexpectedly or cannot afford the overload of frequent integrity checks. Thus, it seems more rational and practical to equip the verification protocol with public verifiability, which is expected to play a more important role in achieving better efficiency for cloud computing. What's more, there is another major concern among previous designs that is the support of dynamic data operation for cloud data storage applications. In cloud computing, the remotely stored electronic data might not only be accessed but also be updated by the clients, e.g., through block modification, deletion, insertion etc. Unfortunately, the-state-of-the art in the context of remote data storage mainly focus on static data files and this dynamic data updates has received limited attention in the data possession applications so far [1, 2, 3, 9]. In view of the key role of public verifiability and dynamic data operation support for cloud data storage, in this paper we present a frame work and an efficient construction for seamless integration of these two components in our protocol design. In addition, most of existing works adopt weaker security models which do not take into account the reset attack. Specifically, the cloud storage server can trigger reset attacks in the upload phase to violate the soundness of the scheme. To the best of our knowledge, it seems that no existing scheme can simultaneously provide provable security in the enhanced security model and enjoy desirable efficiency, that is, no scheme can resist reset attacks while supporting efficient public verifiability and dynamic data operations simultaneously.

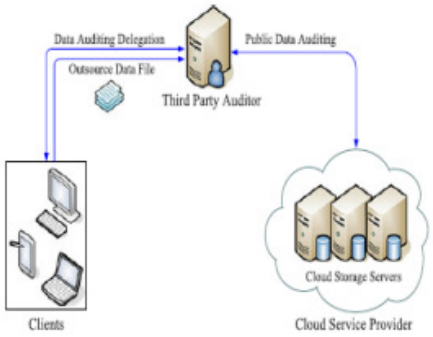

Fig: Cloud data storage architecture

2. Contribution

Our contribution can be summarized as follows: We propose OPDR, a new PoR scheme with two independent cloud servers. Particularly one server is for auditing and the other server for storage of data. The cloud audit server (CAS) is not required to have high storage capacity. Different from the previous work with auditing server and storage server, the user is relieved from the computation of the tags for files, which is moved and outsourced to the cloud audit server. Furthermore, the cloud audit server also plays the role of auditing for the files remotely

stored in the cloud storage server. We develop a strengthened security model by considering the reset attack against the storage server in the upload phase of an integrity verification scheme. It is the first PoR model that takes reset attack into account for cloud storage system. We present an efficient verification scheme for ensuring remote data integrity in cloud storage. The proposed scheme is proved secure against reset attacks in the strengthened security model while supporting efficient public verifiability and dynamic data operations simultaneously.

3. Security Analysis

We analyze the security of our scheme under a variant of Shacham and Waters' PoR model [3] which supports public verifiability and dynamic update operations. Besides, our model offers strengthened security by allowing a malicious storage server to perform reset attack against the client (the cloud audit server) in upload phase. The basic goal of PoR model is to achieve proof of retrievability. Informally, this property ensures that if an adversary can generate valid integrity proofs of any file F for a non negligible fraction of challenges, we can construct a PPT machine to extract F with overwhelming probability. It is formally defined by the following game between a challenger C and an adversary A, where C plays the role of the audit server (the client) and A plays the role of the storage server.

Setup phase: The challenger C runs the setup algorithm to generate its key pair pk; sk, and forward spk to the adversary A.

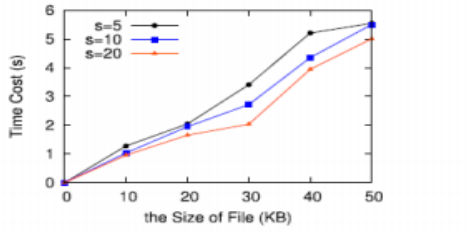

Fig:Tag generation time

Upload phase: C initiates an empty table called R-list. A can adaptively query an upload oracle with reset capability as follows: Upload when a query on a file F and a state index i comes, C checks if there is an entry i; ri in the R-list. If the answer is yes, C overwrites ri on to its random tape; otherwise, C inserts i; ri into R-list where ri is the content on its random tape. Then C runs F_; t Upload sk; F; ri, and returns the stored file F_ and the file tag t. Here Upload p; ri denotes an execution of the upload algorithm using randomness ri.

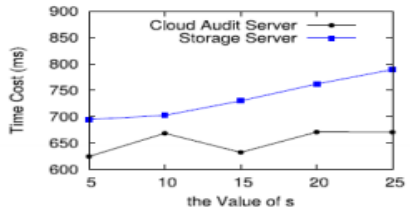

Fig:Verifacation time

Challenge phase: A can adaptively make the following two kinds of oracle queries: Integrity verify, when a query on a file tag t comes, C runs the integrity verification protocol. Integrity update, when a query on a file tag t and a data operation request - update comes, C runs the update protocol Update {A = C(sk, t, update) with A.

Output phase: A outputs a file tag t and the description of a pr over Pt.

4. Performance Analysis

We build our test bed by using 64-bit M2 high-memory quadruple extra large Linux servers in Amazon EC2 platform as the auditing server and storage server, and a Linux machine with Intel (R) Core (TM)2 Duo CPU clocked at 2.40 GHz and 2GB of system memory as the user. In order to achieve _ ¼ 80 bit securities, the prime order p of the GDH group G of the bilinear mapping

should be 160 bits in length. Note that in all the evaluations, the groups G and GT are selected in 160 bit and 512 bit length respectively. Suppose there is a 4GB file with block size 4KB, then it has n¼ 1, 000, 000 blocks and s¼ 25 sectors each block. When it is uploaded onto the storage server, the set of signatures on the file blocks only requires for an additional storage of 20MB for data integrity verification.

5. Conclusion

Paper proposes OPDR, a new proof of retrievability for cloud storage, in which a trustworthy audit server is introduced to preprocess and upload the data on behalf of the clients. In OPDR, the computation overhead for tag generation on the client side is reduced significantly. The cloud audit server also performs the data integrity verification or updating the outsourced data upon the clients' request. Be sides, we construct another new PoR scheme proven secure under PDR model with enhanced security against reset attack in the upload phase. The scheme also supports public verifiability and dynamic data operation simultaneously. There are several interesting topics to do along this research line. For instance, we can (1) reduce the trust on the cloud audit server for more generic applications, (2) strengthen the security model against reset attacks in the data integrity verification protocol, (3) find more efficient constructions requiring for less storage and communication cost, and (4) extend the proposed approach to other data persistent schemes such as NoSql databases. We leave the study of these problems as our future work.

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. (2007). Provable data possession at untrusted stores. *In* Proc. of 14th ACM Conf. Comput. Commun. Security, pp. 598–609.

[2] A. Juels and B. S. Kaliski. (2007). PoRs: Proofs of retrievability for large files. *In* Proc. 14th ACM Conf. Comput. Commun. Security, pp. 584–597.

[3] H. Shacham and B. Waters. (2008). Compact proofs of retrievability. *In* Proc. 14th Int. Conf. Theory Appl. Cryptol. Inf. Security, pp. 90–107.

[4] K. D. Bowers, A. Juels, and A. Oprea. (2009). Proofs of retrievability: Theory and implementation. *In* Proc. ACM Workshop Cloud Comput. Security, pp. 43–54.

[5] M. Naor and G. N. Rothblum. (2009). The complexity of online memory checking. *J. ACM*, 56(1), pp. 2:1–2:46.

[6] E-C Chang and J. Xu. (2008). Remote integrity check with dishonest storage server. *In* Proc. 13th Eur. Symp. Res. Comput. Security, pp. 223–237.

[7] M. A. Shah, R. Swaminathan, and M. Baker. (2008). Privacy-preserving audit and extraction of digital contents, Cryptology ePrint Archive, Report 2008/186 [Online]. Available at http://eprint.iacr. org/

[8] A. Opera, M. K. Reiter, and K. Yang. (2005). Space-efficient block storage integrity. *In* Proc. 12th Annual Netw. Distrib. Syst. Security Symp., pp. 17–28.

[9] T. S. J. Schwarz and E. L. Miller. (2006). Store, forget, and check: using algebraic signatures to check remotely administered storage. *In* Proc. 26th IEEE Int. Conf. Distrib. Comput. Syst., p. 12.