IJDCST @June-July-2015, Issue- V-3, I-5, SW-05
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Scalable and secured TPA auditing for dynamic cloud storage systems with preserved privacy

S. Sandeep Kumar, Dr.K. Vijaya Lakshmi, Dr. Rishi Sayal

Abstract: Cloud computing offers the wide range of storage technologies under IaaS policy. This service alleviates the burden of local data storage and maintenance with a shared pool of configurable computing resources. Although the cloud relieves the user from storage management burden, still there are some unresolved confidentiality issues due to no physical command to the cloud user on stored data. Foremost in IaaS, Data Integrity became a prominent issue in this cloud storage due to the external threats and unauthorized access (attacks) which cause to damage the data to loss of integration. To ensure the data integrity of cloud storage files, TPA based auditing become popular recently. Due to the Honest-but-Curious nature of TPA, cloud user may lose the privacy on his private data. In this paper we proposed a new policy with proper guidelines to implement privacy preserving and public auditing for secured cloud storage systems. By Implementing the PKA, Challenge-Responses and metadata verification proofs in a streamlined manner with a secured policy, our proposed system achieves the security and confidentiality while auditing the cloud user data by TPA. Experimental results are showing that our approach is having the scalability and security over other technologies while implementing the public TPA auditing on cloud storage systems.

Keywords: Cloud computing, cloud storage, TPA auditing, Data Integrity

I. INTRODUCTION

Verifying the authenticity of data has emerged as a critical issue in storing data on untrusted servers. It arises in peer-to-peer storage systems [1, 2], network file systems, long-term archives, web-service object stores [2, 4], and database systems. Such systems prevent storage servers from misrepresenting or modifying data by providing authenticity checks when accessing data.

However, archival storage requires guarantees about the authenticity of data on storage, namely that storage servers possess data. It is insufficient to detect that data have been modified or deleted when accessing the data, because it may be too late to recover lost or damaged data. Archival storage servers retain tremendous amounts of data, little of which are accessed. They also hold data for long periods of time during which there may be exposure to data loss from administration errors as the physical implementation of storage evolves, e.g., backup and restore, data migration to new systems, and changing memberships in peer-to-peer systems.

Recently, the notion of public auditability has been proposed in the context of ensuring remotely stored data integrity under different system and security models [8], [10]. Public auditability allows an external party, in addition to the user himself, to verify the correctness of remotely stored data.

However, most of these schemes [5], [6] do not consider the privacy protection of users' data against external auditors. Indeed, they may potentially reveal user data information to the auditors. This severe drawback greatly affects the security of these protocols in Cloud Computing. From the perspective of protecting data privacy, the users, who own the data and rely on TPA just for the storage security of their data, do not want this auditing process introducing new vulnerabilities of unauthorized information leakage towards their data security. The basic public TPA (TTP) based data integrity checkup model is described in the given below figure1.

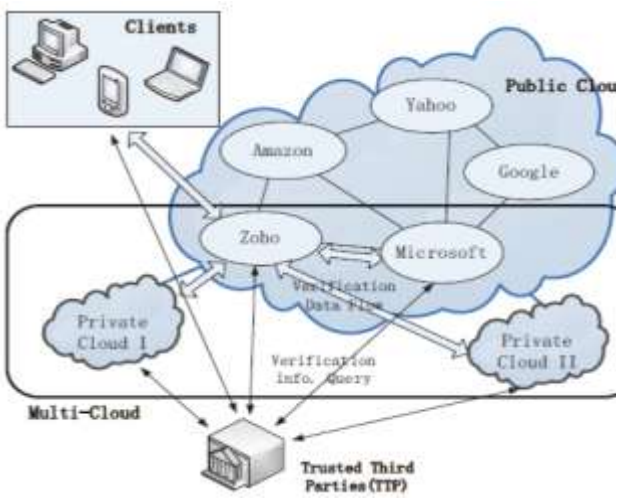


Figure1. public TPA (TTP) based data integrity checkup mod

The first provable data possession (PDP) mechanism [3] to perform public auditing is designed to check the correctness of data stored in an untrusted server, without retrieving the entire data. Moving a step forward, Wang et al. [3] is designed to construct a public auditing mechanism for cloud data, so that during public auditing, the content of private data belonging to a personal user is not disclosed to the third party auditor.

My Research only considers how to audit the integrity of shared data without data leakage in the cloud with static and dynamic groups. It means the group is pre-defined or on-demand before shared data is created in the cloud and the membership of users in the group is not changed during data sharing. The original user is responsible for deciding who is able to share her data before outsourcing data to the cloud.

In this paper we proposed a new policy with proper guidelines to implement privacy preserving and public auditing for secured cloud storage systems. By Implementing the PKA, Challenge-Responses and metadata verification proofs in a streamlined manner with a secured policy, our proposed system achieves the security and confidentiality while auditing the cloud user data by TPA. Experimental results are showing that our approach is having the scalability and security over other technologies while implementing the public TPA auditing on cloud storage systems.

II. LITERATURE REVIEW

In this section we review the former researcher contributions with inspirations and limitations in a brief manner. Deswarte et al. [7] provide techniques to verify that a remote server stores a file using RSA-based hash functions. Unlike other hash-based approaches, it allows a client to perform multiple challenges using the same metadata. Schwarz and Miller [9] propose a scheme that allows a client to verify the storage of m/n erasure-coded data across

multiple sites even if sites collude. The data possession guarantee is achieved using a special construct, called an "algebraic signature": A function that fingerprints a block and has the property that the signature of the parity block equals the parity of the signatures of the data blocks.

Sebe et al. [10] give a protocol for remote file integrity checking, based on the Diffie-Hellman problem in ZN . The client has to store N bits per block, where N is the size of an RSA modulus, so the total storage on the client is O(n) (which does not conform to our notion of an outsourced storage relationship). Indeed, the authors state that this solution only makes sense if the size of a block is much larger than N. Moreover, the protocol requires the server to access the entire file.

Erway et al. [11] developed a skip lists based scheme to enable provable data possession with full dynamics support. However, the verification in these two protocols requires the linear combination of sampled blocks and thus does not support privacy preserving auditing. While all the above schemes provide methods for efficient auditing and provable assurance on the correctness of remotely stored data, none of them meet all the requirements for privacy preserving public auditing in cloud computing.

III. SCALABLE AND SECURED TPA AUDITING

To enable privacy-preserving public auditing for cloud data storage under the aforementioned model, our protocol design should achieve the following security and performance guarantees.

- Public auditability: 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.
- Storage correctness: to ensure that there exists no cheating cloud server that can pass the TPA's audit without indeed storing users' data intact.
- Privacy-preserving: to ensure that the TPA cannot derive users' data content from the information collected during the auditing process.

To achieve privacy-preserving public auditing, we propose to uniquely integrate the homomorphism linear authenticator with random masking technique. In our protocol, the linear combination of sampled blocks in the server's response is masked with randomness generated the server. With random masking, the TPA no longer has all the necessary information to build up a correct group of linear equations and therefore cannot derive the user's data content, no matter how many linear combinations of the same set of file blocks can be collected. On the other hand, the correctness validation of the block authenticator pairs can still be carried out in a new way which will be shown shortly, even with the presence of the randomness. Our design makes use of a public key based HLA, to equip the auditing protocol with public auditability. The whole implementation process is explained is the below figure2 is a sequence diagram as explained above.

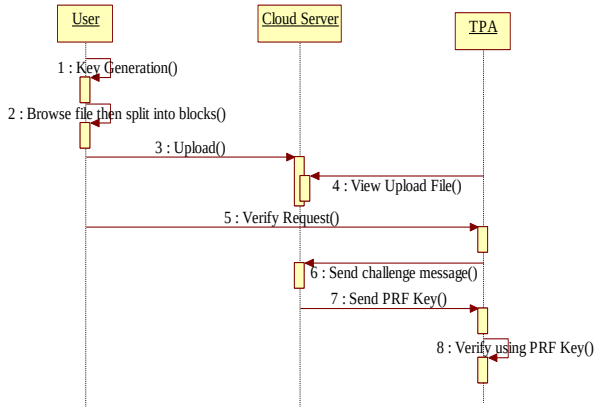


Figure2 workflow sequence of secured TPA auditing approach

Our Proposed public auditing scheme consists of four algorithms (KeyGen, SigGen, GenProof, VerifyProof). KeyGen is a key generation algorithm that is run by the user to setup the scheme. SigGen is used by the user to generate verification metadata, which may consist of MAC, signatures, or other related information that will be used for auditing. GenProof is run by the cloud server to generate a proof of data storage correctness, while VerifyProof is run by the TPA to audit the proof from the cloud server.

Running a public auditing system consists of two phases, Setup and Audit:

• Setup: The user initializes the public and secret parameters of the system by executing KeyGen, and pre-processes the data file F by using SigGen to generate the verification metadata. The user then stores the data file F and the verification metadata at the cloud server, and delete its local copy. As part of pre-processing, the user may alter the data file F by expanding it or including additional metadata to be stored at server.

• Audit: The TPA issues an audit message or challenge to the cloud server to make sure that the cloud server has retained the data file F properly at the time of the audit. The cloud server will derive a response message from a function of the stored data file F and its verification metadata by executing GenProof. The TPA then verifies the response via VerifyProof.

Our framework assumes the TPA is stateless, which is a desirable property achieved by our proposed solution. It is easy to extend the framework above to capture a stateful auditing system, essentially by spliting the verification metadata into two parts which are stored by the TPA and the cloud server respectively

IV. EXPERIMENTS

We now assess the performance of the proposed privacy-preserving public auditing schemes to show that they are indeed lightweight. We will focus on the cost of the efficiency of the privacy-preserving protocol and our proposed batch auditing technique.

We measure the performance of our approach and the benefits of sampling based on our implementation of policies in Linux. As a basis for comparison, we have also implemented the scheme of Deswarte et al. [11] and Filho et al. [12] (B-PDP), and the more efficient scheme in [7] (MHT-SC) suggested by David Wagner.

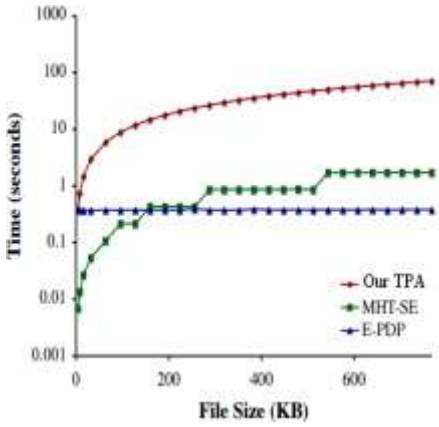
All experiments were conducted on an Intel 2.8 GHz Pentium IV system with a 512 KB cache, an 800 MHz EPCI bus, and 1024 MB of RAM. The system runs Red Hat Linux 9, kernel version 2.4.22. Algorithms use the crypto library of OpenSSL

version 0.9.8b with a modulus N of size 1024 bits and files have 4KB blocks. Experiments that measure disk I/O performance do so by storing files on an ext3 file system on a Seagate Barracuda 7200.7 (ST380011A) 80GB Ultra ATA/100 drive. All experimental results represent the mean of 20 trials. Because results varied little across trials, we do not present confidence intervals.

$Hash^t_{\mathbb{G}_1}$	hash t values into the group $\mathbb{G}_1$.
$Mult^t_{\mathbb{G}}$	t multiplications in group $\mathbb{G}$.
$Exp^t_{\mathbb{G}}(\ell)$	t exponentiations g^{a_i}, for $g \in \mathbb{G}$, $\lvert a_i \rvert = \ell$.
$m\text{-}MultExp^t_{\mathbb{G}}(\ell)$	t m-term exponentiations $\prod_{i=1}^{m} g^{a_i}$.
$Pair^t_{\mathbb{G}_1,\mathbb{G}_2}$	t pairings $e(u_i, g_i)$, where $u_i \in \mathbb{G}_1$, $g_i \in \mathbb{G}_2$
$m\text{-}MultPair^t_{\mathbb{G}_1,\mathbb{G}_2}$	t m-term pairings $\prod_{i=1}^{m} e(u_i, g_i)$.

Table1. Cost in terms of basic cryptographic operations

We begin by estimating the cost in terms of basic cryptographic operations, as notated in above Table 1. Suppose there are c random blocks specified in the challenge message chal during the Audit phase. Under this setting, we quantify the cost introduced of the privacy preserving auditing in terms of server computation, auditor computation as well as communication overhead. The bellow graph1 shows the time complexity analysis and comparison of our approach with the other methods as specified. Experimental results are proving that our TPA approach consumes less time to do auditing than other approaches.



Graph1. Auditing process time complexity comparison

To get a complete view of batching efficiency, we conduct a timed batch auditing test, where the number of auditing tasks is increased from 1 to approximately 200 with intervals of 8. The performance of the corresponding non-batched (individual) auditing is provided as a baseline for the measurement. Following the same experimental settings c = 300 and c = 460, the average per task auditing time, which is computed by dividing total auditing time by the number of tasks.

V. CONCLUSION

We focused on the problem of verifying if an untrusted server stores a client's data. We introduced a model for provable data possession, in which it is desirable to minimize the file block accesses, the computation on the server, and the client-server communication. Due to the Honest-but-Curious nature of TPA, cloud user may lose the privacy on his private data. In this paper we proposed a new policy with proper guidelines to implement privacy preserving and public auditing for secured cloud storage systems. By Implementing the PKA,

Challenge-Responses and metadata verification proofs in a streamlined manner with a secured policy, our proposed system achieves the security and confidentiality while auditing the cloud user data by TPA. Experimental results are showing that our approach is having the scalability and security over other technologies while implementing the public TPA auditing on cloud storage systems.

VI. REFERENCES

1) J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In Proceedings of ACM ASPLOS '00. ACM, November 2000.
2) A. Y. Yumerefendi and J. Chase. Strong accountability for network storage. In Proc. of FAST, 2007.
3) Amazon.com, "Amazon s3 availability event: July 20, 2008," Online at http://status.aws.amazon.com/s3-20080720.html, 2008.
4) 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
5) H. Shacham and B. Waters, "Compact proofs of retrievability," in Proc. of Asiacrypt 2008, vol. 5350, Dec 2008, pp. 90–107.
6) D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," J. Cryptology, vol. 17, no. 4, pp. 297–319,2004.
7) G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. Of SecureComm'08, 2008, pp. 1–10.
8) E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. In Proceedings of NDSS. The Internet Society, 2004.
9) M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable secure file sharing on untrusted storage. In Proc. of FAST, 2003.
10) H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Proc. of CRYPTO '05, volume 3621 of LNCS, pages 546–566. Springer, 2005.
11) I. Damg˚ard. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, editor, CRYPTO91, volume 576, pages 445–456. Springer, 1992.
12) R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic signature schemes. In Proc. of CT-RSA, volume 2271 of LNCS, pages 244–262. Springer, 2002.