Enabling Identity-Based Integrity Auditing and Data Sharing with Sensitive Information Hiding for Secure Cloud Storage

C.Karpagavalli[1], [1]Assistant Professor, Department of Computer Science and Engineering, St.Mother Theresa Engineering College, Tuticorin, India

S. Sakthi selvi[2], V. Latha[3], M.Kalaiarasi[4], J.Epsiba[5]

[2,3,4,5]UG scholars, Department of Computer Science and Engineering, St.Mother Theresa Engineering College, Tuticorin, India.

Abstract*—with cloud storage services, users can remotely store their data to the cloud and realize the data sharing with others. Remote data integrity auditing is proposed to guarantee the integrity of the data stored in the cloud. In some common cloud storage systems such as the Electronic Health Records (EHRs) system, the cloud file might contain some sensitive information. The sensitive information should not be exposed to others when the cloud file is shared. Encrypting the whole shared file can realize the sensitive information hiding, but will make this shared file unable to be used by others. How to realize data sharing with sensitive information hiding in remote data integrity auditing still has not been explored up to now. In order to address this problem, we propose a remote data integrity auditing scheme that realizes data sharing with sensitive information hiding in this paper. In this scheme, a sanitizer is used to sanitize the data blocks corresponding to the sensitive information of the file and transforms these data blocks' signatures into valid ones for the sanitized file. These signatures are used to verify the integrity of the sanitized file in the phase of integrity auditing. As a result, our scheme makes the file stored in the cloud able to be shared and used by others on the condition that the sensitive information is hidden, while the remote data integrity auditing is still able to be efficiently executed. Meanwhile, the proposed scheme is based on identity- based cryptography, which simplifies the complicated certificate management. The security analysis and the performance evaluation show that the proposed scheme is secure and efficient.

1. INTRODUCTION

With the explosive growth of data, it is a heavy burden for users to store the sheer amount of data locally. Therefore, more and more organizations and individuals would like to store their data in the cloud. However, the data stored in the cloud might be corrupted or lost due to the inevitable software bugs, hardware faults and human errors in the cloud [1]. In order to verify whether the data is stored correctly in the cloud, many remote data integrity auditing schemes have been proposed [2–8]. In remote data integrity auditing schemes, the data owner firstly needs to generate signatures for data blocks before uploading them to the cloud. These signatures are used to prove the cloud truly possesses these data blocks in the phase of integrity auditing. And then the data owner uploads these data blocks along with their corresponding signatures to the cloud. The data stored in the cloud is often shared across multiple users in many cloud storage applications, such as Google Drive, Dropbox and iCloud. Data sharing as one of the most common features in cloud storage, allows a number of users to share their data with others. However, these shared data stored in the cloud might contain some

sensitive information. For instance, the Electronic Health Records (EHRs) [9] stored and shared in the cloud usually contain patients' sensitive information (patient's name, telephone number and ID number, etc.) and the hospital's sensitive information (hospital's name, etc.). If these EHRs are directly uploaded to the cloud to be shared for research purposes, the sensitive information of patient and hospital will be inevitably exposed to the cloud and the researchers. Besides, the integrity of the EHRs needs to be guaranteed due to the existence of human errors and software/hardware failures in the cloud. Therefore, it is important to accomplish remote data integrity auditing on the condition that the sensitive information of shared data is protected.

2. RELATED WORK

[2] Provable Data Possession at Untrusted Stores

We introduce a model for provable data possession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. Thus, the PDP model for remote data checking supports large data sets in widely-distributed storage systems. We present two provably-secure PDP schemes that are more efficient than previous solutions, even when compared with schemes that achieve weaker guarantees. In particular, the overhead at the server is low (or even constant), as opposed to linear in the size of the data. Experiments using our implementation verify the practicality of PDP and reveal that the performance of PDP is bounded by disk I/O and not by cryptographic computation.

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. Our solutions for PDP fit this model: They incur a low (or even constant) overhead at the server and require a small, constant amount of communication per challenge. Key components of our schemes are the homomorphic verifiable tags. They allow to verify data possession without having access to the actual data file. Experiments show that our schemes, which offer a probabilistic possession guarantee by sampling the server's storage, make it practical to verify possession of large data sets. Previous schemes that do not allow sampling are not practical when PDP is used to prove possession of large amounts of data. Our experiments show that such schemes also impose a significant I/O and computational burden on the server.

[4] Compact Proofs of Retrievability

In a proof-of-retrievability system, a data storage center must prove to a verifier that he is actually storing all of a client's data. The central challenge is to build systems that are both efficient and provably secure — that is, it should be possible to extract the client's data from any prover that passes a verification check. In

this paper, we give the first proof-of-retrievability schemes with full proofs of security against arbitrary adversaries in the strongest model, that of Juels and Kaliski. Our first scheme, built from BLS signatures and secure in the random oracle model, features a proof-of- retrievability protocol in which the client's query and server's response are both extremely short. This scheme allows public verifiability: anyone can act as a verifier, not just the file owner. Our second scheme, which builds on pseudorandom functions (PRFs) and is secure in the standard model, allows only private verification. It features a proof-of-retrievability protocol with an even shorter server's response than our first scheme, but the client's query is long. Both schemes rely on homomorphic properties to aggregate a proof into one small authenticator value.

3. FRAMEWORK

The system model involves five kinds of different entities: the cloud, the user, the sanitizer, the Private Key Generator (PKG) and the Third Party Auditor (TPA), as shown in Fig.1.

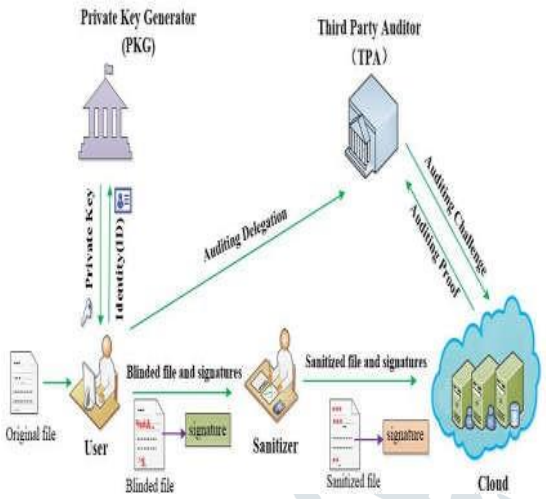

Fig.1. The system model

(1) Cloud: The cloud provides enormous data storage space to the user. Through the cloud storage service, users can upload their data to the cloud and share their data with others.

User: The user is a member of an organization, which has a large number of files to be stored in the cloud.

(2) Sanitizer: The sanitizer is in charge of sanitizing the data blocks corresponding to the sensitive information (personal sensitive information and the organization's sensitive information) in the file, transforming these data blocks' signatures into valid ones for the sanitized file, and uploading the sanitized file and its corresponding signatures to the cloud.

(3) PKG: The PKG is trusted by other entities. It is responsible for generating system public parameters and the private key for the user according to his identity ID.

(4) TPA: The TPA is a public verifier. It is in charge of verifying the integrity of the data stored in the cloud

on behalf of users.

The user firstly blinds the data blocks corresponding to the personal sensitive information of the file, and generates the corresponding signatures. These signatures are used to guarantee the authenticity of the file and verify the integrity of the file. Then the user sends this blinded file and its corresponding signatures to the sanitizer. After receiving the message from the user, the sanitizer sanitizes these blinded data blocks and the data blocks corresponding to the organization's sensitive information, and then transforms the signatures of sanitized data blocks into valid ones for the sanitized file. Finally, the sanitizer sends this sanitized file and its corresponding signatures to the cloud. These signatures are used to verify the integrity of the sanitized file in the phase of integrity auditing. When the TPA wants to verify the integrity of the sanitized file stored in the cloud, he sends an auditing challenge to the cloud. And then, the cloud responds to the TPA with an auditing proof of data possession. Finally, the TPA verifies the integrity of the sanitized file by checking whether this auditing proof is correct or not.

4. EXPERIMENTAL RESULTS

We evaluate the performance of the proposed scheme by several experiments. We run these experiments on a Linux machine with an Intel Pentium 2.30GHz processor and 8GB memory. All these experiments use C programming language with the free Pairing-Based Cryptography (PBC) Library [10] and the GNU Multiple Precision Arithmetic (GMP) [11]. In our experiments, we set the base field size to be 512 bits, the size of an element in Z^*_p to be

|p| =160 bits, the size of data file to be 20MB composed by 1,000,000 blocks, and the length of user identify to be 160 bits.

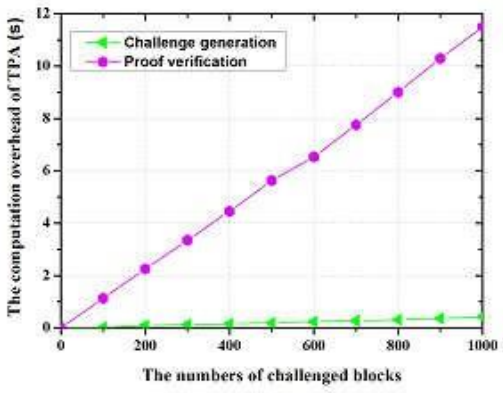

Fig.2. The computation overhead of the TPA in the phase of integrity auditing.

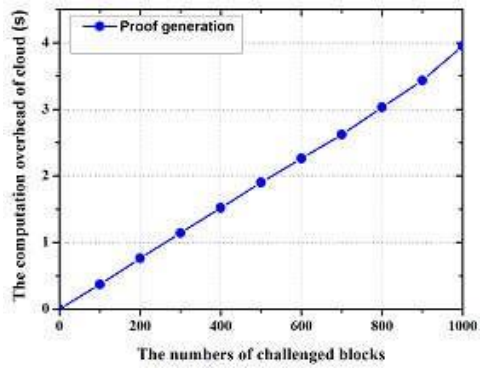

Fig.3. The computation overhead of the cloud in the phase of integrity auditing

5. CONCLUSION

In this paper, we proposed an identity-based data integrity auditing scheme for secure cloud storage, which supports data sharing with sensitive information hiding. In our scheme, the file stored in the cloud can be shared and used by others on the condition that the sensitive information of the file is protected. Besides, the remote data integrity auditing is still able to be efficiently executed. The security proof and the experimental analysis demonstrate that the proposed scheme achieves desirable security and efficiency.

REFERENCES

[1] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," IEEE Internet Computing, vol. 16, no. 1, pp. 69–73, Jan 2012.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proceedings of the 14th ACM Conference on Computer and Communications Security, ser. CCS '07, 2007, pp. 598–609.

[3] A. Juels and B. S. Kaliski, "Pors: Proofs of retrievability for large files," in Proceedings of the 14th ACM Conference on Computer and Communications Security, ser. CCS '07, 2007, pp. 584–597.

[4] H. Shacham and B. Waters, "Compact proofs of retrievability," J. Cryptology, vol. 26, no. 3, pp. 442–483, Jul. 2013.

[5] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," IEEE Transactions on Computers, vol. 62, no. 2, pp. 362–375, 2013.

[6] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," Comput. Electr. Eng., vol. 40, no. 5, pp. 1703–1713, Jul. 2014.

[7] C. Guan, K. Ren, F. Zhang, F. Kerschbaum, and J. Yu, "Symmetric-key based proofs of retrievability supporting public verification," in Computer Security – ESORICS 2015. Cham: Springer International Publishing, 2015, pp. 203–223.

[8] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, and R. Hao, "Light-weight and privacy- preserving secure cloud auditing scheme for group users via the third party medium," Journal of Network and Computer Applications, vol. 82, pp. 56–64, 2017.

[9] J. Sun and Y. Fang, "Cross-domain data sharing in distributed electronic health record systems," IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 6, pp. 754–764, June 2010.

[10] B. Lynn, "The pairing-based cryptographic library," https://crypto.stanford.edu/pbc/, 2015.

[11] "The gnu multiple precision arithmetic library (gmp)," http://gmplib.org/.