International Journal of Scientific Engineering and Technology Research

ISSN 2319-8885
Volume.08,
Jan-Dec-2019,
Pages:122-125

www.ijsetr.com

Secure Auditing and Effective Data Storage in Cloud Environment by using Key Management

MANDAPATI ANUJA[1], P.V. SIVAKUMAR[2]

[1]PG Scholar, Department of CSE, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India.
[2]Associate Professor, Department of CSE, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India.

Abstract: As the cloud computing technology develops during the last decade, outsourcing data to cloud service for storage becomes an attractive trend, which benefits in sparing efforts on heavy data maintenance and management. Nevertheless, since the outsourced cloud storage is not fully trustworthy, it raises security concerns on how to realize data deduplication in cloud while achieving integrity auditing. In this work, we study the problem of integrity auditing and secure deduplication on cloud data. Specifically, aiming at achieving both data integrity and deduplication in cloud, we propose two secure systems, namely SecCloud and SecCloud$^+$. SecCloud introduces an auditing entity with maintenance of a MapReduce cloud, which helps clients generate data tags before uploading as well as audit the integrity of data having been stored in cloud. Compared with previous work, the computation by user in SecCloud is greatly reduced during the file uploading and auditing phases. SecCloud$^+$ is designed motivated by the fact that customers always want to encrypt their data before uploading, and enables integrity auditing and secure deduplication on encrypted data.

Keywords: SecCloud, MapReduce.

I. INTRODUCTION

Cloud storage is a model of networked enterprise storage where data is stored in virtualized pools of storage which are generally hosted by third parties. Cloud storage provides customers with benefits, ranging from cost saving and simplified convenience, to mobility opportunities and scalable service. Even though cloud storage system has been widely adopted, it fails to accommodate some important emerging needs such as the abilities of auditing integrity of cloud files by cloud clients and detecting duplicated files by cloud servers. We illustrate both problems below. The first problem is integrity auditing. The cloud server is able to relieve clients from the heavy burden of storage management and maintenance. The most difference of cloud storage from traditional in-house storage is that the data is transferred via Internet and stored in an uncertain domain, not under control of the clients at all, which inevitably raises clients great concerns on the integrity of their data. These concerns originate from the fact that the cloud storage is susceptible to security threats from both outside and inside of the cloud and the uncontrolled cloud servers may passively hide some data loss incidents from the clients to maintain their reputation. What is more serious is that for saving money and space, the cloud servers might even actively and deliberately discard rarely accessed data files belonging to an ordinary client. Considering the large size of the outsourced data files and the clients' constrained resource capabilities.

The first problem is generalized as how can the client efficiently perform periodical integrity verifications even without the local copy of data files. The second problem is secure deduplication. The rapid adoption of cloud services is accompanied by increasing volumes of data stored at remote cloud servers. Among these remote stored files, most of them are duplicated: according to a recent survey by EMC, 75% of recent digital data is duplicated copies. This fact raises a technology namely dedu- plication, in which the cloud servers would like to deduplicate by keeping only a single copy for each file (or block) and make a link to the file (or block) for every client who owns or asks to store the same file (or block). Unfortunately, this action of deduplication would lead to a number of threats potentially affecting the storage system, for example, a server telling a client that it (i.e., the client) does not need to send the file reveals that some other client has the exact same file, which could be sensitive sometimes. These attacks originate from the reason that the proof that the client owns a given file (or block of data) is solely based on a static, short value (in most cases the hash of the file). Thus, the second problem is generalized as how can the cloud servers efficiently confirm that the client (with a certain degree assurance) owns the uploaded file (or block) before creating a link to this file (or block) for him/her. SecCloud introduces an auditing entity with maintenance of a MapReduce cloud, which helps clients generate data tags before uploading as well as audit the integrity of data having been stored in cloud.

This design fixes the issue of previous work that the computational load at user or auditor is too huge for tag generation. For completeness of fine-grained, the functionality of auditing designed in SecCoud is supported on both block level and sector level. In addition, SecCoud also enables secure deduplication. Notice that the "security" considered in SecCoud is the prevention of leakage of side channel information. In order to prevent the leakage of such side channel information, we follow the tradition of and design a proof of ownership protocol between clients and cloud servers, which allows clients to prove to cloud servers that they exactly own the target data.

II. RELATED WORK

Since our work is related to both integrity auditing and secure deduplication, we review the works in both areas in the following subsections, respectively.

A. Integrity Auditing

The definition of provable data possession (PDP) was introduced by Ateniese for assuring that the cloud servers possess the target files without retrieving or downloading the whole data. Essentially, PDP is a probabilistic proof protocol by sampling a random set of blocks and asking the servers to prove that they exactly possess these blocks, and the verifier only maintaining a small amount of metadata is able to perform the integrity checking. In order to achieve efficient data dynamics, combined the privacy-preserving word search algorithm with the insertion in data segments of randomly generated short bit sequences, and developed a new POR protocol. Li et al. considered new cloud storage architecture with two independent cloud servers for integrity auditing to reduce the computation load at client side. Recently, Li et al. utilized the key-disperse paradigm to fix the issue of a significant number of convergent keys in convergent encryption.

B. Secure Deduplication

Deduplication is a technique where the server stores only a single copy of each file, regardless of how many clients asked to store that file, such that the disk space of cloud servers as well as network bandwidth are saved. However, trivial client side deduplication leads to the leakage of side channel information. For example, a server telling a client that it need not send the file reveals that some other client has the exact same file, which could be sensitive information in some case. In order to restrict the leakage of side channel information, introduced the proof of ownership protocol which lets a client efficiently prove to a server that that the client exactly holds this file. Several proof of ownership protocols based on the Merkle hash tree are proposed to enable secure client-side deduplication. Pietro and Sorniotti proposed an efficient proof of ownership scheme by choosing the projection of a file onto some randomly selected bit-positions as the file proof.

III. PRELIMINARY

A. Bilinear Map and Computational Assumption

Definition 1 (Bilinear Map): Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of large prime order p. A bilinear pairing is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

- *Bilinear:* $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $g_1, g_2 \in_R \mathbb{G}$ and $a, b \in_R \mathbb{Z}_p$;
- *Non-degenerate:* There exists $g_1, g_2 \in \mathbb{G}$ such that $e(g_1, g_2) \neq 1$;
- *Computable:* There exists efficient algorithm to compute $e(g_1, g_2)$ for all $g_1, g_2 \in_R \mathbb{G}$.

We now discuss some preliminary notions that will form the foundations of our approach. The examples of such groups can be found in supersingular elliptic curves or hyperelliptic curves over finite fields, and the bilinear pairings can be derived from the Weil or Tate pairings. We then describe the Computational Diffie-Hellman problem, the hardness of which will be the basis of the security of our proposed schemes.

Definition 2 (CDH Problem): The Computational Diffie-Hellman problem is that, given $g, g^x, g^y \in \mathbb{G}_1$ for unknown $x, y \in \mathbb{Z}_p^*$, to compute g^{xy}.

B. Convergent Encryption

Convergent encryption provides data confidentiality in deduplication. A user (or data owner) derives a convergent key from the data content and encrypts the data copy with the convergent key. In addition, the user derives a tag for the data copy, such that the tag will be used to detect duplicates. Here, we assume that the tag correctness property holds, i.e., if two data copies are the same, then their tags are the same. Formally, a convergent encryption scheme can be defined with four primitive functions:

- $\mathsf{KeyGen}(\mathcal{F})$: The key generation algorithm takes a file content $\mathcal{F}$ as input and outputs the convergent key $ck_{\mathcal{F}}$ of $\mathcal{F}$;
- $\mathsf{Encrypt}(ck_{\mathcal{F}}, \mathcal{F})$: The encryption algorithm takes the convergent key $ck_{\mathcal{F}}$ and file content $\mathcal{F}$ as input and outputs the ciphertext $ct_{\mathcal{F}}$;
- $\mathsf{Decrypt}(ck_{\mathcal{F}}, ct_{\mathcal{F}})$: The decryption algorithm takes the convergent key $ck_{\mathcal{F}}$ and ciphertext $ct_{\mathcal{F}}$ as input and outputs the plain file $\mathcal{F}$;
- $\mathsf{TagGen}(\mathcal{F})$: The tag generation algorithm takes a file content $\mathcal{F}$ as input and outputs the tag $tag_{\mathcal{F}}$ of $\mathcal{F}$. Notice that in this paper, we also allow $\mathsf{TagGen}(\cdot)$ to generate the (same) tag from the corresponding ciphertext

IV. SECCLOUD

In this section, we describe our proposed SecCloud system. Specifically, we begin with giving the system model of SecCloud as well as introducing the design goals for SecCloud.

A. System Model

Aiming at allowing for auditable and deduplicated storage, we propose the SecCloud system. In the SecCloud system, we have three entities:

- Cloud Clients have large data files to be stored and rely on the cloud for data maintenance and computation. They can be either individual consumers or commercial organizations;
- Cloud Servers virtualize the resources according to the requirements of clients and expose them as storage

pools. Typically, the cloud clients may buy or lease storage capacity from cloud servers, and store their individual data in these bought or rented spaces for future utilization;

- Auditor which helps clients upload and audit their out-sourced data maintains a MapReduce cloud and acts like a certificate authority. This assumption presumes that the auditor is associated with a pair of public and private keys. Its public key is made available to the other entities in the system.

The SecCloud system supporting file-level deduplication includes the following three protocols respectively highlighted by red, blue and green in Fig1.

Fig 1: SecCloud Architecture.

1. File Uploading Protocol

This protocol aims at allowing clients to upload files via the auditor. Specifically, the file uploading protocol includes three phases:

Phase 1 (cloud client → cloud server): client performs the duplicate check with the cloud server to confirm if such a file is stored in cloud storage or not before uploading a file. If there is a duplicate, another protocol called Proof of Ownership will be run between the client and the cloud storage server. Otherwise, the following protocols (including phase 2 and phase 3) are run between these two entities.

Phase 2 (cloud client → auditor): client uploads files to the auditor, and receives a receipt from auditor.

Phase 3 (auditor → cloud server): auditor helps generate a set of tags for the uploading file, and send them along with this file to cloud server.

2. Integrity Auditing Protocol

It is an interactive protocol for integrity verification and allowed to be initialized by any entity except the cloud server. In this protocol, the cloud server plays the role of prover, while the auditor or client works as the verifier. This protocol includes two phases:
Phase 1 (cloud client/auditor → cloud server): verifier (i.e., client or auditor) generates a set of challenges and sends them to the prover (i.e., cloud server).

Phase 2 (cloud server → cloud client/auditor): based on the stored files and file tags, prover (i.e., cloud server) tries to prove that it exactly owns the target file by sending the proof back to verifier (i.e., cloud client or auditor). At the end of this protocol, verifier outputs true if the integrity verification is passed.

3. Proof of Ownership Protocol

It is an interactive protocol initialized at the cloud server for verifying that the client exactly owns a claimed file. This protocol is typically triggered along with file uploading protocol to prevent the leakage of side channel information. On the contrast to integrity auditing protocol, in PoW the cloud server works as verifier, while the client plays the role of prover. This protocol also includes two phases
Phase 1 (cloud server → client): cloud server generates a set of challenges and sends them to the client.

Phase 2 (client → cloud server): the client responds with the proof for file ownership, and cloud server finally verifies the validity of proof.

Our main objectives are outlined as follows.
Integrity Auditing: The first design goal of this work is to provide the capability of verifying correctness of the remotely stored data. The integrity verification further requires two features: 1) public verification, which allows anyone, not just the clients originally stored the file, to perform verification; 2) stateless verification, which is able to eliminate the need for state information maintenance at the verifier side between the actions of auditing and data storage.

Secure Deduplication: The second design goal of this work is secure deduplication. In other words, it requires that the cloud server is able to reduce the storage space by keeping only one copy of the same file. Notice that, regarding to secure deduplication, our objective is distinguished from previous work [3] in that we propose a method for allowing both deduplication over files and tags.

Cost-Effective: The computational overhead for providing integrity auditing and secure deduplication should not represent a major additional cost to traditional cloud storage, nor should they alter the way either uploading or downloading operation.

V. SECCLOUD^{+}

We specify that our proposed SecCloud system has achieved both integrity auditing and file deduplication. However, it cannot prevent the cloud servers from knowing the content of files having been stored. In other words, the functionalities of integrity auditing and secure deduplication are only imposed on plain files. In this section, we propose SecCloud+, which allows for integrity auditing and deduplication on encrypted files.

A. System Model

Compared with SecCloud, our proposed SecCloud+ involves an additional trusted entity, namely key server,

which is responsible for assigning clients with secret key (according to the file content) for encrypting files. This architecture is in line with the recent work. But our work is distinguished with the previous work by allowing for integrity auditing on encrypted data. SecCloud+ follows the same three protocols (i.e., the file uploading protocol, the integrity auditing protocol and the proof of ownership protocol) as with SecCloud. The only difference is the file uploading protocol in SecCloud+ involves an additional phase for communication between cloud client and key server.

VI. PERFORMANCE ANALYSIS

In this section, we will provide a thorough experimental evaluation of our proposed schemes. Micro Linux servers in Amazon EC2 platform as the auditing server and storage server. The time cost of slave node in MapReduce for generating file tags. It is clear the time cost of slave node is growing with the size of file. This is because the more blocks in file, the more homomorphic signatures are needed to be computed by slave node for file uploading. We also need to notice that there does not exist much computational load difference between common slave nodes and the reducer. Compared with the common slave nodes, reducer only additionally involves in a number of multiplications, which is lightweight operation. It is worthwhile noting that, the procedure of tag generation could be handled in preprocessing, and it is not necessary for client to wait until uploading file. Now, we come back to evaluate the time cost of file auditing, which shows the time cost of auditing for detecting the misbehavior of cloud storage respectively with 70%, 85% and 99% confidence. Obviously, as the growth of the number of blocks for challenge (to guarantee higher confidence), the time cost for response from cloud storage server is increasing. This is because it needs to compute all the exponentiations for each challenge block as well as the coefficient for each column of S. Correspondingly, the time cost at auditor grows with the number of challenge blocks as well. But compared with cloud storage, the rate is slightly lower, because auditor only needs to aggregate the homomorphic signature of the challenged blocks.

VII. CONCLUSION

Aiming at achieving both data integrity and deduplication in cloud, we propose SecCloud and SecCloud+. SecCloud introduces an auditing entity with maintenance of a MapReduce cloud, which helps clients generate data tags before uploading as well as audit the integrity of data having been stored in cloud. In addition, SecCoud enables secure deduplication through introducing a Proof of Ownership protocol and preventing the leakage of side channel information in data deduplication. Compared with previous work, the computation by user in SecCloud is greatly reduced during the file uploading and auditing phases. SecCloud+ is an advanced construction motivated by the fact that customers always want to encrypt their data before uploading, and allows for integrity auditing and secure deduplication directly on encrypted data.

VIII. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M.Zaharia, "A view of cloud computing," Communication of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[2] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in IEEE Conference on Communications and Network Security (CNS), 2013, pp. 145–153.

[3] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in Proceedings of the 18th ACM Conference on Computer and Communications Security. ACM, 2011, pp. 491–500.

[4] S. Keelveedhi, M. Bellare, and T. Ristenpart, "Dupless: Serveraided encryption for deduplicated storage," in Proceedings of the 22Nd USENIX Conference on Security, ser. SEC'13. Washington, D.C.: USENIX Association, 2013, pp. 179–194. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technicalsessions/presentation/bellare.

[5] 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. New York, NY, USA: ACM, 2007.

Author's Profile:

Mandapati Anuja, have been graduated in B.Tech. Computer Science and Engineering from CVSR college of Engineering and Technology, Hyderabad in the period of 2011-2015. Currently pursuing post graduation in M.Tech. Computer Science and Engineering from VNR Vignana Jyothi Institute of Engineering and Technology Hyderbad, that is affiliated to Jawaharlal Nehru Technological University, Hyderabad in the period of 2018-2019.

Dr. P.V.Sivakumar is an Associate Professor at the Department of Computer Science & Engineering, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad. His research interest is in the area of Image Processing and Cloud computing. He has done his post graduation from Jawaharlal Nehru Technological University, Hyderabad. He has been awarded with PhD in Computer Science and Engineering from Acharya Nagarjuna University in the year 2016.