View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by International Journal of Innovative Technology and Research (IJITR)

Machani Prasanna Kumar* et al.
(IJITR) INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH
Volume No.4, Issue No.4, June – July 2016, 3367 – 3372.

Detection of Duplication in Cloud

MACHANI PRASANNA KUMAR
PG Student (Software Engineering)
St. Johns College of Engineering and Technology
Kurnool, Andhra Pradesh, India

G K VENKATA NARASIMHA REDDY
Associate professor, Department of CSE
St. Johns College of Engineering and Technology
Kurnool, Andhra Pradesh, 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 a 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.

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. These great features attract more and more customers to utilize and storage their personal data to the cloud storage: according to the analysis report, the volume of data in cloud is expected to achieve 40 trillion gigabytes in 2020.

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 [1], 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 [2], 75% of recent digital data is duplicated copies. This fact raises a technology namely deduplication, 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 [3][2], 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) [3]. 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.*

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.

 @ 2013-2016 http://www.ijitr.com All rights Reserved.

A. Integrity Auditing

The definition of provable data possession (PDP) was introduced by Ateniese et al. [5][6] 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. After Ateniese et al.'s proposal [5], several works concerned on how to realize PDP on dynamic scenario: Ateniese et al. [7] proposed a dynamic PDP schema but without insertion operation; Erway et al. [8] improved Ateniese et al.'s work [7] and supported insertion by introducing authenticated flip table; A similar work has also been contributed in [9]. Nevertheless, these proposals [5][7][8][9] suffer from the computational overhead for tag generation at the client. To fix this issue, Wang et al. [10] proposed proxy PDP in public clouds. Zhu et al. [11] proposed the cooperative PDP in multi-cloud storage.

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, Halevi et al. [3] 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 [3] to enable secure client-side deduplication. Pietro and Sorniotti [19] 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

We now discuss some preliminary notions that will form the foundations of our approach.

A. Bilinear Map and Computational Assumption

B. Convergent Encryption

IV. SECCLOUD

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

In what follows, we illustrate the proposed SecCloud in detail.

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 outsourced 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 Fig. 1.

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.

 @ 2013-2016 http://www.ijitr.com All rights Reserved.

• ***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 ntegrity 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.

B. SecCloud Details

In this subsection, we respectively describe the three protocols including file uploading protocol, integrity auditing protocol and proof of ownership protocol in SecCloud. Before our detailed elaboration, we firstly introduce the system setup phase of SecCloud, which initializes the public and private parameters of the system. As declared in Section IV-A, the file uploading protocol involves three phases. In the first phase shown in Fig. 2, the client runs the deduplication test by sending hash value of the file Hash(*F*) to the cloud server. If there is a duplicate, the cloud client performs Proof of Ownership protocol with the cloud server which will be described later. If it is passed, the user is authorized to access this stored file without uploading the file.

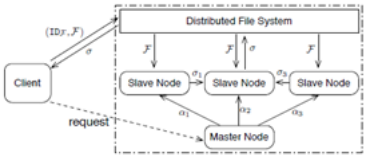

Fig. 2. Phase 1 in File Uploading Protocol

2) Integrity Auditing Protocol: In the integrity auditing protocol, either the MapReduce auditing cloud or the client works as the verifier. Thus, without loss of generality, in the rest of the description of this protocol, we use verifier to identify the client or MapReduce auditing cloud. The auditing protocol is designed in a challenge-response model. Specifically, the verifier randomly picks a set of block identifiers (say *IF*) of *F* and asks the cloud server (working as prover) to response the blocks corresponding to the identifiers in *IF*. In order to keep randomness in each time of challenge, even for the same *IF*, we introduce a random coefficient for each block in challenge. Upon receiving the challenge *C*, as shown in Fig. 3,

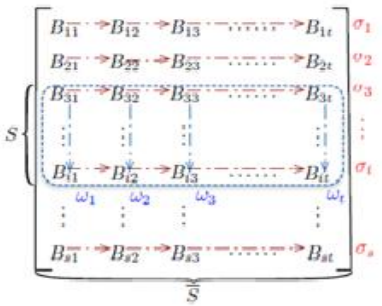

Fig. 3. Matrix for Proof of Retrievability

 @ 2013-2016 http://www.ijitr.com All rights Reserved.

suppose the Merkle hash tree has been constructed as in Fig. 4, and the challenge blocks *IF = {2; 5}* (i.e., challenge *B*2*;B*5). The hashes of *B*2 and *B*5 (highlighted by black in Fig. 4), Ω2 (highlighted by blue in Fig. 4) and Ω5 (highlighted by orange in Fig. 4) are as the proof for retrievability on block-level. It is worth noting that, although the node labeled by *x* in Fig. 4 is a sibling of node in Path(*B*2), it should not be included in Ω2. This is because the node *x* also belongs to Path(*B*5) and can be re-constructed using Hash(*B*5) and Ω5. The benefit of excluding the nodes in other challenge blocks paths is that, it allows us to reconstruct only a *single* version of root node of the Merkle hash tree for auditing *all* the challenge blocks.

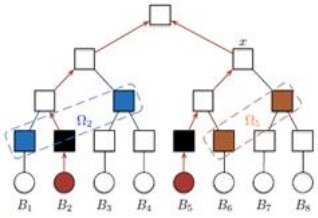

Fig. 4. Auxiliary Information in Merkle Hash Tree

3) Proof of Ownership Protocol: The PoW protocol aims at allowing secure deduplication at cloud server. Specifically, in deduplication, a client claims that he/she has a file *F* and wants to store it at the cloud server, where *F* is an existing file having been stored on the server. The cloud server asks for the proof of the ownership of *F* to prevent client unauthorized or malicious access to an unowned file through making cheating claim. In SecCloud, the PoW protocol is similar to [3] and the details are described as follows.

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 [4]. But our work is distinguished with the previous work [4] 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. That is, the client needs to communicate with the key server to get the convergent key for encrypting the uploading file before the phase 2 in SecCloud. Unlike SecCloud, another design goals of file confidentiality is desired in SecCloud+ as follows.

• File Confidentiality. The design goal of file confidentiality requires to prevent the cloud servers from accessing the content of files. Specially, we require that the goal of file confidentiality needs to be resistant to "dictionary attack". That is, even the adversaries have pre-knowledge of the "dictionary" which includes all the possible files, they still cannot recover the target file [4].

VI. SECURITY ANALYSIS

In this section, we attempt to analyze the security of our proposed both schemes. Before this, we firstly formalize the security definitions our schemes aim at capturing.

A. Security Definitions

Based on the paradigm of SecCloud and SecCloud+, we define the security definitions, adapting to the integrity auditing and secure deduplication goals. Our both definitions capture the philosophy of game-based definition. Specifically, we define two games respectively for integrity auditing and secure deduplication, and both of the games are played by two players, namely adversary and challenger. The adversary (the role of which is worked by semi-honest cloud server and cloud client respectively in integrity auditing and secure deduplication definition) is trying to achieve the goal condition explicitly specified in the game. Having this intuition, we give our security definitions as follows.

1) Integrity Auditing: An integrity auditing protocol is sound if any cheating cloud server that convinces the verifier that it is storing a file *F* is actually storing this file. To capture this spirit, we define its game based on Proof of Retrievability (PoR).

2) Secure Deduplication: Similarly, we can also define a game between challenger and adversary for secure deduplication below. Notice that the game for secure deduplication captures the intuition of allowing the malicious client to claim it has a challenge file *F* through colluding with all the other clients not owning this file.

The security in terms of secure deduplication is achieved, if for all probabilistic polynomial-time adversaries *A*, the probability that *A* succeeds in the above experiment is negligible.

 @ 2013-2016 http://www.ijitr.com All rights Reserved.

B. Security Proof

Theorem 1: Assume that the CDH problem is a hard problem. Then, the proposed public-verifiable PoR scheme satisfies the soundness. That is, no adversary could generate an integrity proof for any file such that the verifier accepts it with nonnegligible probability.

Proof: We prove the soundness of the construction by reduction. Firstly, assume there is an adversary who can break the soundness with non-negligible probability. We show that how to construct a simulator to break the computational Diffie-Hellman problem through interacting with the adversary. During this phase, the simulator is required to answer all the queries as the real application. In more details, the simulator has to answer the tag generation and integrity proof queries from the adversary. After the simulation, if the adversary outputs a valid tag that is not from client, the simulator can use this algorithm to solve the CDH problem. Notice that the simulation for the n slave nodes can be reduced to just one node because of the assumption that all the slave nodes are honest-but-curious and they will not collude. More clearly, the master key α can be split to n subkeys by choosing $n - 1$ random values and assigned to slave nodes as the corresponding private keys, while the n-th node is assigned the key of α minus the sum of these random values.

Furthermore, all the data has been encrypted before they are outsourced. The data is encrypted with the traditional symmetric encryption scheme and the key is generated by the key server. The convergent key is encrypted by another master key and stored in the cloud server. The convergent key has been computed from both the file and private key of the key server, which means that the convergent key is not deterministic only in terms of the file. Even if the file is predictable, the adversary cannot guess the file with brute-force attack if the adversary is not allowed to collude with the key server.

Because we used the PoW technique, based on the assumption of secure PoW scheme, any adversary without the file cannot convince the cloud storage server to get the corresponding access privilege. Thus, our deduplication system is secure in terms of the security model.

VII. PERFORMANCE ANALYSIS

In this section, we will provide a thorough experimentale valuation of our proposed schemes. We build our testbed by using 64-bit t2.Micro Linux servers in Amazon EC2 platform as the auditing server and storage server. In order to achieve _ = 80 bit security, the prime order p of the bilinear group G and GT are respectively chosen as 160 and 512 bits in length. We also set the block size as 4 KB and each block includes 25 sectors.

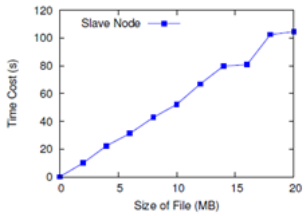

Fig. 5. Tag Generation

Fig. 5 shows 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 (the phase 2 and 3 in file uploading protocol) could be handled in preprocessing, and it is not necessary for client to wait until uploading file.

To capture the spirit of probabilistic auditing, we set the probability confidence _ = 70%*;* 85% and 99%, and draw the relationships between _ and m in Fig. 6. It demonstrates that if we want to achieve low (i.e., 70%), medium (i.e., 85%) and high (i.e., 99%) confidence of detecting any small fraction of corruption, we have to respectively ask for 130*;* 190 and 460 blocks for challenge.

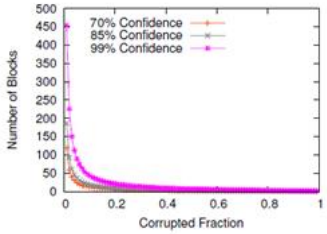

Fig. 6. Number of Challenging Blocks with Fixed Confidence

Now, we come back to evaluate the time cost of file auditing in Fig. 7, 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

 @ 2013-2016 http://www.ijitr.com All rights Reserved.

aggregate the homomorphic signature of the challenged blocks.

Fig. 7. File Auditing

VIII. 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.

IX. 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, pp. 598– 609.

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote data checking using provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 12:1–12:34, 2011.

[7] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, ser. SecureComm '08. New York, NY, USA: ACM, 2008, pp. 9:1–9:10.

[8] C. Erway, A. K¨upc¸ ¨u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 213–222.

[9] F. Seb´e, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Trans. on Knowl. and Data Eng.*, vol. 20, no. 8, pp. 1034–1038, 2008.

[10] H. Wang, "Proxy provable data possession in public clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 551–559, 2013.

AUTHOR's PROFILE

Machani Prasanna kumar received B.Tech Degree from Sri Sai Jyothi Engineering College in Hyderabad. He is currently pursuing M.tech Degree in Software Engineering specialization in Yerrakota, kurnool, India.

Mr.GKV Narasimha Reddy received Ph.D from Annamalai University and received M.tech degree from Sathyabama University. He is currently working as Associate professor, Department of CSE, in St.Johns College of engineering and technology, Kurnool, Andhra Pradesh, India. His interests include Computer Networks, Operating system, Data Base Management Systems.

 @ 2013-2016 http://www.ijitr.com All rights Reserved.