International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 11 | Nov 2018 www.irjet.net p-ISSN: 2395-0072

Secure Data Deduplication for Cloud Server using HMAC Algorithm

P Jagadeesh[1], K Narayana[2], P Chandra Prakash[3]

[1]*Student, Department of Computer Science and Engineering, Sechachala Institute of Technology, Puttur, Andhra Pradesh, India.*
[2]*Associate Professor, HOD, Department of Computer Science and Engineering, Sechachala Institute of Technology, Puttur, Andhra Pradesh, India.*
[3]*Assistant Professor, Department of Computer Science and Engineering, Sechachala Institute of Technology, Puttur, Andhra Pradesh, India.*

Abstract - *The elimination of duplicate or redundant data, particularly in computer data is named deduplication. Data deduplication is a method to regulate the explosive growth of information within the cloud storage, most of the storage providers are finding more secure and efficient methods for their sensitive method. According to deduplication, we introduce a method that can eliminate redundant encrypted data owned by different users. This paper is a detail description of secure cloud auditor which is used for the maintaining integrity of shared data with efficient data deduplication on cloud. This mechanism uses concept of SecCloud system where user is able to generate data tags before storing data on cloud which helps during performing audit to check integrity of data.*

Key Words: *Cloud computing, Integrity, Auditing, Advanced Encryption Standard (AES), Data deduplication.*

1. INTRODUCTION

Cloud computing (The Fifth Generation of Computing) is a term used to describe both a platform and type of application. A cloud computing platform dynamically provisions, configures, reconfigures, and DE provisions servers as needed. Servers in the cloud can be physical machines or virtual machines. Advanced clouds typically include other computing resources such as storage area networks (SANs), network equipment, firewall and other security devices. Cloud computing also describes

Applications that are extended to be accessible through the Internet. These cloud applications use large data centers and powerful servers that host Web applications and Web services. Anyone with a suitable Internet connection and a standard browser can access a cloud application. 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 DE duplication in cloud

While achieving integrity auditing. In this work, we study the problem of integrity auditing and secure DE duplication on cloud data. Specifically, aiming at achieving both data integrity and DE duplication in cloud, we propose two secure systems, namely SecCloud and SecCloud+. Sec Cloud introduces an auditing entity with maintenance of a Map Reduce 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 DE duplication on encrypted data.

Even though cloud storage has been widely used 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 disclose both problems.

The first problem is integrity auditing. Data integrity demands maintaining and assuring the accuracy and completeness of data. A data owner always expects that his data in a cloud can be stored correctly and trustworthily. It means that the data should not be illegally tampered, improperly modified, deliberately deleted, or maliciously fabricated. If any undesirable operations corrupt or delete the data, the owner should be able to detect the corruption or loss.

The second problem is secure deduplication. Data stored at remote cloud servers are often duplicated. This fact raises a technology named deduplication, in which the cloud servers would like to de-duplicate by keeping only a single copy for each file. It is generalized to how can the cloud server efficiently confirms that the client owns the uploaded file before creating a link to this file for him/her.

Accordingly, indistinguishable data duplicates of various clients will prompt to various figure writings, making deduplication unimaginable. It scrambles/decodes a data duplicate with a private key, which is acquired by computing the cryptographic hash estimation of the substance of the data copy. Clients safeguard the keys and send the figure content to the cloud. Since, indistinguishable data duplicates will cause the same focalized key and thus the equivalent figure content.

2. RELATED WORK

Secure deduplication is interesting for both industrial and research communities; therefore, several secure deduplication schemes have been proposed. To support data integrity, two concepts, PDP (provable data possession) and POR (proof of retrievability), have been introduced. PDP for ensuring that the cloud storage providers actually possess the files without retrieving or downloading the entire data. It is basically a challenge-response protocol between the verifier (a client or TPA) and the prover (a cloud). Compared to PDP, POR not only ensures that the cloud servers possess the target files, but also guarantees their full recovery.

A simple combination of two independent techniques designed for the two above mentioned issues does not efficiently deal with the issues at once, because achieving storage efficiency contradicts with the deduplication of authentication tags. Public auditing with a deduplication scheme based on homomorphic linear authentication tags was proposed.

Integrity Auditing

From the perspective of the client, integrity auditing of the outsourced data is one of the important issues for secure outsourcing, as the outsourced data can be corrupted by unintentional errors. Ateniese et al. [7] proposed a notion of provable data possession (PDP) for ensuring integrity of remote data, in which the client can audit the integrity of the target file without maintaining the entire file. Ateniese et al. [21] proposed a highly efficient PDP scheme based on symmetric key cryptography, with the support of a dynamic scenario except insertion. To support dynamic scenario with insertion, Erway et al. [22] proposed dynamic-PDP based on a rank-based skip list. Wang et al. [23] proposed a proxy-PDP, in order to relax the computational overhead for tag generation. Zhu et al. [24] proposed a cooperative-PDP scheme in a multicloud environment. Based on convergence encryption, Liu et al. [25] proposed integrity auditing scheme and considered integrity tag deduplication over encrypted data.

Secure Client-Side Deduplication with Integrity Auditing

As a method that provides both secure deduplication and integrity auditing, Zheng and Xu [9] firstly proposed proof of storage with deduplication (POSD), based on public key cryptography. However, an error in security occurs if the first uploader maliciously generates a pair of public and private keys [31], and POSD does not ensure confidentiality of the outsourced data as it is run over plaintext. Yuan and Yu [2] proposed a scheme called PCAD that supports both deduplication and integrity auditing with batch auditing, in which the server can simultaneously prove the possession of multiple files. Li et al. [1] proposed two schemes, namely SecCloud and SecCloud+. In both schemes, the author introduced an auditing entity that maintains a MapReduce cloud, which helps the client to generate block tags for integrity auditing. Additionally, SecCloud+ ensures confidentiality, where the client encrypts files using a message-derived encryption key distributed from the key server. In terms of efficiency improvement, Youn el al. [32] proposed a new scheme based on the homomorphic linear authenticator [26].

3. PROPOSED METHODOLOGY

After analyzing different research paper on spam detection in Twitter, I have considered Hash Message Authentication Code (HMAC) for survey.

HMAC Method

1) HMAC

In recent years, there has been increased interest in developing a MAC derived from a cryptographic hash code, such as MD5, SHA-1, or RIPEMD-160. The motivations for this interest are:

Cryptographic hash works by and large execute quicker in programming than symmetric square figures, for example, DES. Library code for cryptographic hash capacities is broadly accessible. There are no fare confinements for cryptographic hash capacities, though symmetric square figures, notwithstanding when utilized for MACs, are limited.

A hash capacity, for example, MD5 was not intended for use as a MAC and can't be utilized specifically for that reason since it doesn't depend on a mystery key. There have been various recommendations to fuse a mystery enter into a current hash calculation. HMAC got the most help. HMAC has been picked as the required to-actualize MAC for IP Security, and is utilized in other Internet conventions, such as Transport Layer Security (TLS, soon to replace Secure Sockets Layer) and Secure Electronic Transaction (SET).

2) HMAC-Design Objectives

Design objectives that RFC 2104 lists for HMAC include:

- To use, without alterations, accessible hash capacities. Specifically, hash works that perform well in programming, and for which code is uninhibitedly and generally accessible. To consider simple replaceability of the inserted hash work in the event that quicker or more secure hash capacities are found or required. To save the first execution of the hash work without acquiring a noteworthy debasement. To utilize and handle enters essentially.

- To have a surely knew cryptographic examination of the quality of the validation system dependent on sensible suppositions on the implanted hash work.

The initial two targets are imperative to the adequacy of HMAC. HMAC regards the hash work as a black box. This has two advantages. Initial, a current usage of a hash capacity

can be utilized as a module in executing HMAC. The main part of the HMAC code is prepackaged and prepared to use without change. Second, to supplant a given hash work in a HMAC execution, you should basically evacuate the current hash work module and drop in the new module. This should be possible if a quicker hash work were wanted. More imperative, if the security of the implanted hash work were imperiled, the security of HMAC could be held basically by supplanting the installed hash work with a more secure one (supplanting MD5 with SHA-1, for instance).

The last structure objective in the former rundown is, truth be told, the primary preferred standpoint of HMAC over other proposed hash-based plans. HMAC can be proven secure provided that the embedded hash function has some reasonable cryptographic strengths.

3) The HMAC Algorithm

Illustrates the overall operation of HMAC :

Affix zeros to one side end of K to make a b-bit string K+ (for instance, if K is of length 160 bits and b = 512, at that point K will be attached with 44 zero bytes 0x00).

Note the XOR with ipad brings about flipping one-portion of the bits of K. Thus, the XOR with opadresults in flipping one-portion of the bits of K, yet an alternate arrangement of bits. As a result, by passing Si thus through the pressure capacity of the hash calculation, you have pseudorandomly produced two keys from K. HMAC ought to execute in around indistinguishable time from the inserted hash work for long messages. HMAC includes three executions of the hash pressure work (for Si, So, and the square created from the inward hash).

4) HMAC Security

The security of any MAC function based on an embedded hash function depends in some way on the cryptographic strength of the underlying hash function. The interest of HMAC is that its creators have possessed the capacity to demonstrate a correct connection between the quality of the implanted hash work and the quality of HMAC. The security of a MAC work is for the most part communicated as far as the likelihood of effective fraud with a given measure of time spent by the counterfeiter and a given number of message-MAC sets made with a similar key. Generally, it very well may be demonstrated that, for a given level of exertion (time, message-MAC sets), on messages created by real clients and seen by aggressors, the likelihood of an effective assault on HMAC is comparable to one of the accompanying assaults on the installed hash work:

1. Assailants can process a yield of the pressure work even with an Initial Value (IV) that is irregular, mystery, and obscure to aggressors.

2. Aggressors discover crashes in the hash work notwithstanding when the IV is irregular and mystery.

In the principal assault, you can see the pressure work as proportionate to the hash work connected to a message comprising of a solitary b-bit square. For this assault, the IV of the hash work is supplanted by a mystery, arbitrary estimation of n bits. An assault on this hash work requires either a beast compel assault on the key, which is a level of exertion on the request of 2n, or a birthday assault, which is a unique instance of the second assault.

In the second assault, aggressors are searching for two messages, M and M', that deliver a similar hash: H(M)=H(M'). This requires a level of exertion of 2n/2 for a hash length of n. On this premise, the security of MD5 is raised doubt about, on the grounds that a level of exertion of 264 looks doable with the present innovation. Does this imply a 128-piece hash capacity, for example, MD5 is inadmissible for HMAC? The appropriate response is no. To assault MD5, aggressors can pick any arrangement of messages and work on these disconnected on a devoted figuring office to discover a crash. Since aggressors know the hash calculation and the default IV, assailants can produce the hash code for every one of the messages that aggressors create. Be that as it may, while assaulting HMAC, assailants can't produce message/code sets disconnected in light of the fact that aggressors don't know K. In this manner, aggressors must watch a grouping of messages created by HMAC under a similar key and play out the assault on these known messages. For a hash code length of 128 bits, this requires 264 watched squares (273 bits) produced utilizing a similar key. On a 1-Gbps interface, you would need to watch a ceaseless stream of messages with no adjustment in the key for around 250,000 years to succeed. Thus, if speed is a concern, it is fully acceptable to use MD5 rather than SHA-1 or RIPEMD-160 as the embedded hash function for HMAC.

4. CONCLUSION

With the help of Hybrid Cloud Approach and Convergent Encryption we achieved secured data deduplication. To discover copy information Proof of Ownership (PoW) convention is utilized. Which gave reference of file which is already present on Public Cloud.

5. FUTURE SCOPE

Currently By using HMAC Algorithm, Hash Value computation for large File Size will be performance hit. So further in future work we can use Attribute-Based Encryption for encrypting the File Data. To compute the File Tag, instead of using whole file data we will use File Attributes. In Future Scope, It can be finding Video Spam content & Image Spam Content.

REFERENCES

[1] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," in Proc. IEEE Trans.ParallelDistrib.Syst.,http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.284,2013.

[2] A. Rahumed, H. C. H. Chen, Y. Tang, P. P. C. Lee, and J. C. S. Lui, "A secure cloud backup system with assured deletion and version control," in Proc. 3rd Int. Workshop Secutiry Cloud Comput., 2011,pp. 160–167.

[3] "Proofs of Ownership in Remote Storage Systems" Shai Halevi1, Danny Harnik2, Benny Pinkas3, and Alexandra Shulman-Peleg2 1IBM T. J. Watson Research Center, 2IBM Haifa Research Lab, 3Bar Ilan University.

[4] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: Serveraided encryption for deduplicated storage," in Proc. 22nd USENIX Conf. Sec. Symp., 2013, pp. 179– 194.

[5] S. Quinlan and S. Dorward, "Venti: A new approach to archival storage," in Proc. 1st USENIX Conf. File Storage Technol., Jan. 2002.

[6] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system, " in Proc. Int. Conf. Distrib. Comput. Syst., 2002.

[7] Jan Stanek, Alessandro Sorniotti, Elli Androulaki, and Lukas Kencl, "A Secure Data Deduplication Scheme for Cloud Storage," Tech Rep. IBM Reasearch, Zurich.