International Journal of Research in Engineering, Science and Management
Volume 4, Issue 7, July 2021
https://www.ijresm.com | ISSN (Online): 2581-5792

Deduplication of Data in the Cloud-based Server for Chat/File Transfer Application

Ankit Rai[1*], Arpit Diwan[2], Harsh Tripathi[3], K. Rajkumar[4]
[1,2,3]*Student, Department of Information Technology, Galgotias College of Engineering and Technology, Greater Noida, India*
[4]*Assistant Professor, Department of Information Technology, Galgotias College of Engineering and Technology, Greater Noida, India*

***Abstract*: Due to the heavy use and collection of data, cloud data warehouses are also becoming increasingly popular among computer users. However, the majority of the data in cloud storage is not necessary. Cloud storage providers are using data de-duplication as a mechanism of storing only one copy of a file, which, thanks to information technology cloud storage, hard disk drive, in order to reduce operating expenses, storage and handling of your data. However, the deduplication process is creating a lot of problems and issues. This article discusses the methodology, in order to ensure the de-duplication of the presentation of the data, popularity data, and the assumptions on which the data will require different levels of security, which is based on the popularity of the. We provide a mechanism for secure data de-duplication, take advantage of the dynamic of the advanced connection technology.**

***Keywords*: Cloud storage, Convergent key encryption, Dynamic hashing, Data deduplication.**

1. Introduction

With the increase in the demand for the transfer of data, the speed of the whole world cloud storage systems is becoming a necessity for the user. All of the users that have uploaded their data to the cloud storage and, if necessary, to get the distance data from. However, 70% of the data that is stored in the cloud storage, it is not necessary, which allows us to make effective use of the space and storage. Data deduplication is the process of removing the products from the cloud storage. Data de-duplication can occur in a number of ways: the client-side de-duplication, that is, before you can transfer the data to the user's cloud storage", in to order to be sure that the information is not necessary. In the server-side deduplication, and after that upload the data to the cloud storage server, no matter if it's the controls for more information, or to reject them.

Data de-duplication can be done at several levels of detail, that is to say, at the level of files and blocks. Store your data in remote locations, that is to say, in the cloud storage, it can cause a variety of problems and safety issues. A common approach to data privacy is to encrypt the data before uploading it to cloud storage. However, to download a backup copy of the information to the cloud storage, it cannot remove the data already in the cloud, as the cloud storage provider is not able to determine whether two different encrypted text files, that match the same file is present or not. It is a solution that can solve this problem is convergent encryption. Here, the encryption key is obtained by using a file into a hash function. The resulting hash code is used as the key in order to encrypt the files. Because of the properties of the hash function ensures that no two files will produce the same hash code, if the user re-encrypt your files with the help of the resulting convergent key and it will produce the same ciphertext.

This will help you to get in touch, cloud services, and disable any unnecessary files. Convergent encryption, in order to enable de-duplication of data and personal information, in accordance with (c). Convergent key encryption is vulnerable to the well-known problem, that is, brute-force attacks, and basic attacks, and. This paper proposes a mechanism to provide data deduplication considering, how popular the data item is. In general, a data item owned by several users, like a popular audio track requires less security. A data item which does not belong to many users like a research proposal requires better security. Deduplication will be applied only to the popular file. If the file is not popular, it will not be deduplicated, will be encrypted using the conventional symmetric key encryption. The popularity of a file will be considered a factor for data deduplication.

Existing System: In the present system the uploaded data will do so nailed with flexible encryption. Variable encryption defines as the data will be encrypted with one of the secure hash functions like sha, sha256, sha512 etc. from secure hash to work we get one hash value, using that hash value data will be encrypted again of ciphers i.e.) block the cipher in such a way that the buttons will do the last hash key will be stored in a cloud service database provider, whenever the client again wants to upload data and generated key will be verified with an existing database. If key matches match in the existing database, then will not store the data again displays the message as the file is being uploaded successfully. That we can save a lot space, cost, complex system time, however Flexible encryption has some problem with collision and dictionary attacks.

*Corresponding author: emailankitrai@gmail.com

Disadvantages of existing system: There are many strategies for identifying and acting delete duplicate data but these methods will do it only applies to file level i.e.) if the file existed uploaded by arun.txt name then it will be uploaded to the cloud. Whenever a client wants to upload a file with another name but the same data inside the file and the file is uploaded successfully. It was a great return to being there system.so that we need to duplicate the data within the file.

2. Data Deduplication Techniques

A. File-level data deduplication strategy

File-level classification is commonly referred to as Single Instance Storage (SIS), check backing indexes or archive files that require attributes stored in a file by comparison. If it is not the same file, it will save and update the index; Otherwise, the only deposit identifier in an existing file. Therefore, the same file stores only one instance, and then copies the rest of the "stub", while the "stub" points to the original file.

B. Block-level data deduplication technology

Block-level data deduplication technology to data stream is broken into blocks, check the data block, and decide whether to meet the same data before the block (usually using a hash algorithm for each data block to create a digital signature or a different identifier. Index Other than that, the only deposit indicator to keep the same data booth location. This path is less powerful than duplicate data blocks, rather than duplicating data blocks, thus saving disk space. The Hash algorithm used to judge duplicate data, can lead to conflict between hash errors. MD5, SHA-I hash algorithm, etc. has been tested against data blocks to create a unique code. Although there may be a potential conflict with hash data corruption, they were less likely

Post process: Post-processing printing means recurrence after data is imported. In short, linearization means recurrence before data is written to disk while processing after processing means recurrence after that.

3. Proposed Method

For this project a client must create an account to store their data. Registered client has to login using their username and password to upload the files in the cloud server using the client application. The uploaded file will be encrypted using the SHA-1 hashing algorithm. And the checksum will be saved in the user account metadata. The uploaded file will be stored in the temporary location for the analysis of the data deduplication. The server or cloud will start the data deduplication process and eliminate the redundant data, if the data is found then the data will be deleted from the temporary location. And if the data is unique or no other data copies found in the cloud then the files will be moved from the temporary location to the drive location.

4. Hash based Deduplication

Hash-based data usage methods use a hashing algorithm to identify "masses" of data. The most widely used algorithms are Secure Hash Algorithm 1 (SHA-1) and Message-Digest Algorithm 5 (MD5). When data is processed with a hashing algorithm, a hash is generated that represents the data. The hash is a thin wire (128 pieces of MD5 and 160 pieces of SHA-1) representing the data used. If you process the same data with the hashing algorithm multiple times, the same hash is generated each time.

Here are some examples of hash codes:

MD5 – 16-byte long hash
– # echo “The Quick Brown Fox Jumps Over the Lazy Dog” | md5sum
9d56076597de1aeb532727f7f681bcb0
– # echo “The Quick Brown Fox Dumps Over the Lazy Dog” | md5sum
5800fccb352352308b02d442170b039d
SHA-1 – 20-byte long hash
– # echo “The Quick Brown Fox Jumps Over the Lazy Dog” | sha1sum
F68f38ee07e310fd263c9c491273d81963fbff35
– # echo “The Quick Brown Fox Dumps Over the Lazy Dog” | sha1sum
d4e6aa9ab83076e8b8a21930cc1fb8b5e5ba2335

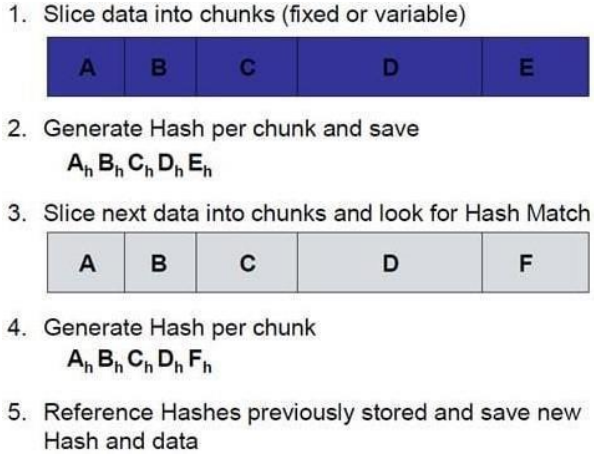


Fig. 1. Hash based deduplication

Hash based de-duplication breaks data into “chunks”, either fixed or variable lengths, and processes a “chunk” with a hashing algorithm for creating hash. If a hash already exists, the data is considered duplicate and is not saved. If the hash is missing, data is saved and the hash index is updated with the new hash.

In figure 1, data “chunks” A, B, C, D, and E are processed by the hash algorithm and creates hashes Ah, Bh, Ch, Dh, and Eh; For the purposes of this example, we assume that this is all new data.

Later, “chunks” A, B, C, D, and F are processed. UF produces new hash Fh. Since A, B, C, and D have produced the same hash, the data is treated as the same hash, so it is not stored again. As F generates a new hash, a new hash and new data are stored.

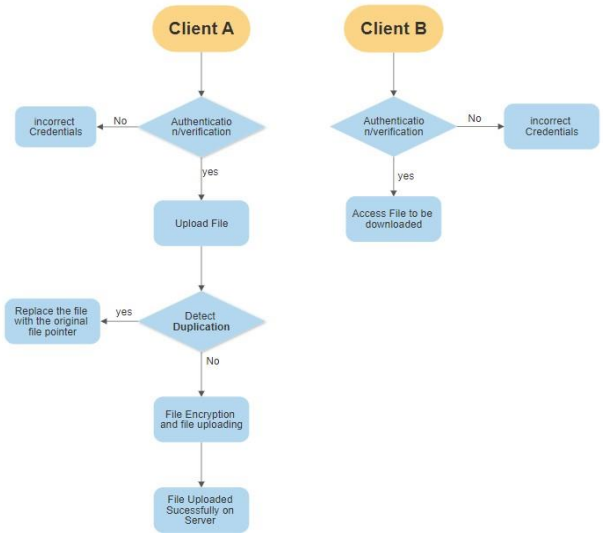
5. System Architecture

Whenever a user wants to upload files, they must provide the appropriate login credentials to their account to upload that file and upload it successfully. Whenever a user wants to download that file they must provide the appropriate login credentials to their account and one password which they send to their mobile phone to download that file and that file will be downloaded successfully.

Our application is deployed over the cloud architecture as shown below, Information about the original files and the new segments/chunks is maintained in a metadata structure. Deduplicated segments and metadata structure are sent over the network on client's request.

We're using the TCP and sockets protocol for establishing the connection between the client application and the server.

6. Implementation



We have implemented a strategy for the data deduplication. In order to implement data deduplication, we must make several decisions that are based on the outcome that we want to achieve. We have implemented the server-side deduplication. Server-side data deduplication is a two-step process in which duplicate data is identified and then storage space is reclaimed to remove the duplicate data. Determine database log size requirements. It is essential that you properly size the storage capacity for the database active log and archive log.

7. Analysis

This section describes the security features of our scheme, focusing on how to implement privacy protection, brute force attacks resistance, and data availability. A brute force attack is a trial-and-error method used to decode sensitive data. The most common applications for brute force attacks are cracking passwords and cracking encryption keys. In our system application, we manage authorization systems can take measures such as locking out IP addresses that have generated too many failed logins, and incorporating a delay in their password-checking software. If the server detects that a security attack is in progress, the current session is cancelled. The server also issues a message that a potential security attack was detected and that server-side data deduplication was disabled for the node.

8. Conclusion and Future Work

Deduplication in last few years has become an active area of research mainly for storage reduction but there are many associated issues such as maximizing storage reduction, bandwidth utilization and authentication, privacy, security and availability of the shared files. In this paper, our focus is on saving maximum possible storage and providing a secure deduplication mechanism in order to keep the client's trust on the system. For Block-level deduplication when the size of a specific file is greater than a predefined threshold, we made chunks of that specific size and applied crypto-hash function which significantly improved our deduplication percentage. After the comparison of our experimental results, we found our system showed 5% improvement to the previously discussed framework, in addition to improving results we also had made comparison of the File-level data deduplication and Blocklevel data deduplication, the security model of our approach provides the same privacy and security measures to the user data.

This simulation work was specifically focused on the performance comparison of File-level deduplication and Block-level deduplication, but our future work includes further optimizing these already taken results and making its practical deployment and usable prototype. We will further evaluate these solutions in terms of security, latency, throughput, bandwidth and reconstruction of the blocks at during file retrieval. Furthermore, we also have plans to implement the same deduplication system on Server-side as well. Server-side deduplication solutions will always perform a full upload from the client to the server and are therefore not susceptible to the client side-channel attacks. The downside of server-side deduplication is that; it provides minimum bandwidth savings

References

[1] Harnik, Danny, Benny Pinkas, and Alexandra Shulman-Peleg, "Side channels in cloud services, the case of deduplication in cloud storage," IEEE Security & Privacy, vol. 8, no. 6 (2010): 40-47.

[2] Xu, Jia, and Jianying Zhou. "Leakage-resilient proofs of ownership in cloud storage, revisited." International Conference on Applied Cryptography and Network Security. Springer International Publishing, 2014.

[3] Armknecht, Frederik, et al. "Transparent data deduplication in the cloud," Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015.

[4] Bellare, Mihir, Sriram Keelveedhi, and Thomas Ristenpart. "Message-locked encryption and secure deduplication." In Advances in Cryptology EUROCRYPT 2013, pp. 296-312. Springer Berlin Heidelberg, 2013.

[5] B. Tirapathi Reddy, U.Ramya, M. V. P. Chandra Sekhara Rao, "A comparative study on data deduplication techniques in cloud storage", IJPT, vol. 8, no. 3, pp. 18521-18530, Sept. 2016.

[6] Di Pietro, Roberto, and Alessandro Sorniotti, "Proof of ownership for deduplication systems: A secure, scalable, and efficient solution." Computer Communications, vol. 82 (2016): 71-82.

[7] Xu, Jia, Ee-Chien Chang, and Jianying Zhou, "Weak leakage-resilient client-side deduplication of encrypted data in cloud storage." In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 195-206. ACM, 2013.

[8] Stanek, Jan, Alessandro Sorniotti, Elli Androulaki, and Lukas Kencl, "A secure data deduplication scheme for cloud storage." In Financial Cryptography and Data Security, pp. 99-118. Springer Berlin Heidelberg, 2014.

[9] Leesakul, Waraporn, Paul Townend, and Jie Xu. "Dynamic data deduplication in cloud storage." Service-Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on. IEEE, 2014.

[10] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: Server aided encryption for deduplicated storage," In USENIX Security Symposium, 2013.

[11] Halevi, Shai, et al. "Proofs of ownership in remote storage systems," Proceedings of the 18th ACM conference on computer and communications security. ACM, 2011.

[12] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," in IEEE Transactions on Parallel and Distributed Systems, 2013.