# Increasing Performance and Efficiency of Cloud by De-Duplication Algorithms

Priyadarsini R
Assistant Professor,
Computer science and Engineering,
Muthayammal Engineering College,

Gomathi M
B.E(CSE) - Final year,
Muthayammal Engineering college

Duddukunta Priyanka
B.E(CSE) Final year,
Muthayammal Engineering college

Pasupuleti Hema Phani Sri
B.E(CSE) - Final year,
Muthayammal Engineering college,

*Abstract* - **Data duplication in the one of the major problems in the cloud which occupies a larger amount of space in the cloud server also with the larger repeated data. Also, there is some threat of security of the data because of the duplication of the same data which is not being encrypted. In this paper, we are going to implement a collaborative algorithm in combining the items classification and using the concept of the cloud model. Firstly the data are being filtered by using their property type and inner classified by the size of the data along with the data name and data similarity. Now the data is being clustered with data similarity using k-mean algorithm. Then the repeated data duplication is being filtered and deleted, keeping one original data. Now the original data is being mapped to different user having the same data and tagged with Boolean values. Now this single data is being encrypted and will be having both public and private keys, whereas the original owner of the data will be having the private key while remaining users mapped to the similar data will be having the public key which is used only to view the data. Now the data is being clustered with data similarity using k-we implement a prototype of our own proposed authorized duplicate check scheme and conduct testbed experiments using our prototype by reducing the amount of space along with avoidance of data duplications.**

***Key Words: Data duplication; De duplication; Finding duplicates***

## 1. INTRODUCTION

The unstable increase of data brings new challenges to the data storage and management in cloud settings. These data classically have to be processed in a suitable style in the cloud. Thus, any superior latency may source a huge loss to the enterprises. Duplication finding plays a very major role in data management. Data deduplication calculates a restricted fingerprint for every data chunk by using hash algorithms such as MD5 and SHA-1.

The planned fingerprint is then comparing touching other available chunks in a database that dedicates for storing the chunks.Though, there is simply one copy for every file stored in cloud immobile if such a file is owned by a massive number of users. As a conclusion, Deduplication system improves storage consumption whereas dropping reliability.

Moreover, the features of privacy for reactive data also arise while they are outsourced by users to cloud. Aiming to contract with the above safety challenges, this paper makes the first stab to honor the idea of distributed dependable Deduplication system. We propose new distributed Deduplication systems with advantaged reliability in which the data chunks are distributed diagonally various cloud servers.

The security wants of data privacy and tag consistency are also achieve by introduce a deterministic furtive sharing system in distributed storage systems, as an option of using convergent encryption as in foregoing Deduplication systems.

## 2. RELATED WORKS

### *2.1 Secure and Constant Cost Public Cloud Storage Auditing with Deduplication*

Data integrity and storage efficiency are two important requirements for cloud storage. Proof of Retrievability (POR) and Proof of Data Possession (PDP) techniques assure data integrity for cloud storage. Proof of Ownership (POW) improves storage efficiency by securely removing unnecessarily duplicated data on the storage server. However, trivial combination of the two techniques, in order to achieve both data integrity and storage efficiency, results in non-trivial duplication of metadata (i.e., authentication tags), which contradicts the objectives of POW.

In this paper we solve this open problem with a novel scheme based on techniques including polynomial-based authentication tags and homomorphic linear authenticators. Our design allows deduplication of both files and their corresponding authentication tags. Data integrity auditing and storage deduplication are achieved simultaneously.

### *2.2 Proofs of Ownership in Remote Storage Systems*

Cloud storage systems are becoming increasingly popular. A promising technology that keeps their cost down is *deduplication*, which stores only a single copy of repeating data. *Client-side deduplication* attempts to identify deduplication opportunities already at the client and save the bandwidth of uploading copies of existing files to the server.

In this work we identify attacks that exploit client-side deduplication, allowing an attacker to gain access to arbitrary size files of other users based on very small hash signatures of these files. More specifically, an attacker who knows the hash signature of a file can convince the storage service that it owns that file; hence the server lets the attacker download the entire file. (In parallel to our work, a subset of these attacks was recently introduced in the wild with respect to the Dropbox file synchronization service.) To overcome such attacks, we introduce the notion of proofs-of-ownership (PoWs), which lets a client efficiently prove to a server that that the client holds a file, rather than just some short information about it. We formalize the concept of proof-of-ownership, under rigorous security definitions, and rigorous efficiency requirements of Peta byte scale storage systems.

### *2.3 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.

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 op- posed 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.

## 3. METHODOLOGY

### *Deduplication*

Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data. Related and somewhat synonymous terms are intelligent (data) compression and single-instance (data) storage. In this part, we appear how to derive the fine grained block level distributed deduplication. In this system, the client also demands to perform the file level deduplication before uploading file. The user partition this files into blocks, if no duplication is found and performs block-level deduplication system. The system set up is similar to file-level deduplication and also block size parameter will be defined.

### *Data Sharing*

In this module two algorithms are used which are Share and Recover. Share algorithm is used for partitioned and shared secret. With sufficient shares, Extracted and retrieved the secret with the help of Recover algorithm. Share divides secret S into (k-r) fragments of same size, which produces r for random fragments of the equal size, and translates into simple language the k fragments using a non-systematic k-of–n erasion code into n shares of the similar size.

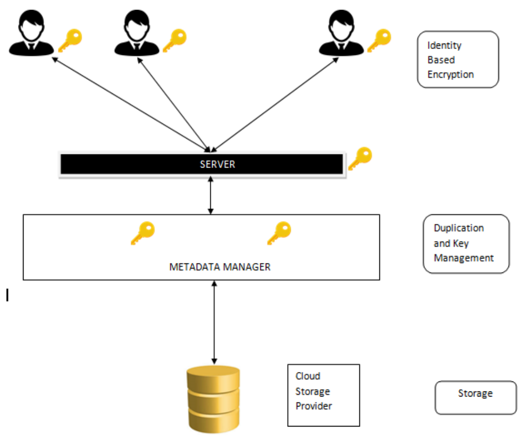

Fig. 1. Architecture Diagram

### *Distributed Storage*

This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. In the deduplication process, unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk. Given that the same byte pattern may occur dozens, hundreds, or even thousands of times (the match frequency is dependent on the chunk size), the amount of data that must be stored or transferred can be greatly reduced.

### **User Revocation**

User revocation is performed by the group manager via a public available revocation list, based on which group members can encrypt their data files and ensure the confidentiality against the revoked users. Let the group manger update the revocation list each day even no user has being revoked in the day. In other words, the others can verify the freshness of the revocation list from the contained current date tRL. In addition, the revocation list is bounded by a signature sig declare its validity. The signature is generated by the group manager with the BLS signature algorithm, i.e., finally, the group manager mi-grates the revocation list into the cloud for public usage.

### *File Restored*

File stored in the cloud can be deleted by the data owner (i.e., the member who uploaded the file into the server). To delete a file IDdata, the group manager computes a signature IDdata and sends the signature along with IDdata to the cloud. The cloud will delete the file.
File Access:To learn the content of a shared file, a member does the following actions:

- Getting the data file and the revocation list from the cloud server. After a successful verification, the cloud server responds the corresponding data file and the revocation list to the user.

- Checking the validity of the revocation list. This operation is similar to the step 2 of file generation phase.
- Verifying the validity of the file and decrypting it. The format of the downloaded file coincides with that given in Table.

## ALGORITHM

```
(1) Monitoring period($t_m$)
1: for every link j do
2:measure link-quality($l_q$) using passive monitoring;
3: end for
4: send monitoring results to a gateway  g;
(2) Failure detection and group formation period($t_f$)
5: if link l violates link requirements  r then
6: request a group formation on channel c  of link l ;
7: end if
8: participate in a leader election if a request is received;
(3) Planning period (M, $t_p$)
9: if node  i  is elected as a leader then
10: send a planning request message (c,M ) to gateway;
11: else if node  i is a gateway then
12:synchronize requests from reconfiguration groups $M_n$
13:generate a reconfiguration plan  (p) for  $M_i$;
14:send a reconfiguration plan p  to a leader of  $M_i$ ;
15:end if
(4) Reconfiguration period ( p,$t_r$ )
16: if p  includes changes of node  i  then
17: apply the changes to links at t ;
18: end if
19: relay p to neighboring members, if any
```

## 4. CONCLUSION

Managing encrypted data with deduplication is significant in practice for running a secure, dependable, and green cloud storage service, especially for big data processes. Future work includes efficient data ownership verification, scheme optimization with hardware acceleration at IoT devices for practical deployment, and development of a flexible solution to support deduplication and data access controlled by either the data owner or its representative agent.

## REFERENCES

[1] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, “Message-locked encryption for lock-dependent messages,”
[2] M. Azraoui, K. Elkhiyaoui, R. Molva, and M.O¨nen, “Stealthguard:Proofs of retrievability with hidden watchdogs,”
[3] M.Bellare, S.Keelveedhi, and T. Ristenpart, “Message-locked encryption and secure deduplication,”
[4] J. Li, X. Tan, X. Chen, and D. Wong, “An efficient proof of retrievability with public auditing in cloud computing,”