A Survey on Secure Deduplication of Data in Cloud Storage

Babaso D. Aldar
ME Student
Computer Engineering Department
Shah and Anchor Kutchhi Engineering College, Chembur, Mumbai, Maharashtra, India

Vidyullata Devmane
Asst. Professor
Computer Engineering Department
Shah and Anchor Kutchhi Engineering College, Chembur, Mumbai, Maharashtra, India

Abstract- Cloud Computing provides many resources to users as services such as highly available storage space. To manage the ever-increasing volume of data in cloud is a critical task. To make data management scalable in cloud computing, data deduplication is a technique. It is the technique of data compression for eliminating duplicate copies of repeating data in cloud storage to reduce the amount of storage space. To better protect the security of data during deduplication, the technique can be used called convergent encryption. This technique is used to encrypt the data before outsource to the data storage server and it should be authorized. The differential privileges of users can be considered while supporting the data deduplication. The data is encrypted by using convergent key, which is derived from the hash value of the same data. A critical issue of making convergent encryption practical is to efficiently and reliably manage a huge number of convergent keys derived for different data files or blocks, first baseline approach can be used, in which each user holds an independent master key for encrypting the convergent keys. Dekey is the derived key, which is the second approach, in which users do not need to manage any keys on their own but instead securely distribute the convergent key shares across multiple cloud servers. To implement Dekey, Ramp secret sharing scheme can be used.

Keywords- Deduplication, Convergent encryption, Authorized duplicate check, Dekey.

I. INTRODUCTION

Cloud service providers offer highly available storage space and massively parallel computing resources at relatively low costs. The advent of Cloud Storage motivates enterprises and organizations to outsource data storage to third party cloud providers. An increasing amount of data is being stored in the cloud and shared by users with specified privileges, which define the access rights of the stored data. Gmail is an example of cloud storage which is used by most of us regularly. One of the major issues of cloud storage services is the management of the ever-increasing volume of data. To make data management scalable in cloud computing, deduplication is a technique and has attracted more and more attention recently. Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeated data in storage. Data deduplication is also known as single instancing or intelligent compression technique [1]. This technique is used to improve storage utilization. Instead of keeping multiple data copies with the same content on cloud, deduplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy.

Deduplication can take place at either the file level or the block level [2]. For file level deduplication, it eliminates duplicate copies of the same file. Microsoft's Single Instance Server (SIS) and EMC's Centera use a file level deduplication [3]. For block level deduplication, it eliminates duplicate blocks of data that occur in non-identical files. Dropbox cloud storage uses a very large fixed-size (4MB) block-level deduplication [3]. Deduplication can occurs at Inline, Post-process, Client-side and Target-based [4]. In Inline deduplication, it occurs before data stored on cloud i.e. it is performed at the time of storing data on storage system. It reduces the disk space needed in the system [4]. In Post-process deduplication, it occurs after storing data on cloud i.e. it refers to the type of system where software processes, filters the redundant data from a data set only after it has already been transferred to a data stored location. In Client-side deduplication, it occurs at Owner/User side, in that duplicate data is first only identified before it has to be sent over the network. This will definitely create burden on the CPU but at the same time reduces the load on the network. It is proposed to minimize bandwidth and space needed to upload and store duplicated data. Kim et al.[3] given that many leading cloud based storage services including Dropbox,

wuala, Memopal, JustCloud, and Mozy use data deduplication techniques at a source i.e at a client side to save network bandwidth from a user to cloud servers, which in turn increases the speed of data upload as well as storage space. In Target-based de-duplication, it occurs at storage service provider side. The Target deduplication will remove the redundancies from a backup transmission as and when it passes through an appliance that is present between the source and the target. Unlike source deduplication, the Target deduplication does not reduce the total amount of data that need to be transferred across a WAN or LAN during the backup, but it reduces the amount of storage space required [4].

Data deduplication brings a lot of benefits, security and privacy concerns arise as users' sensitive data are susceptible to both insider and outsider attacks. Traditional encryption, while providing data confidentiality, is incompatible with data de-duplication. Traditional encryption requires different users to encrypt their data with their own keys by which identical data copies of different users will lead to different cipher texts, making de-duplication impossible [5]. The solution for balancing confidentiality and efficiency in deduplication was described by M. Bellare et al [6] called convergent encryption. It has been proposed to enforce data confidentiality while making deduplication. It encrypts/decrypts a data copy with a convergent key, which is derived by computing the cryptographic hash value of the content of the data copy itself [7]. To prevent unauthorized access, a secure proof of ownership protocol [8] is also needed to provide the proof that the user indeed owns the same file when a duplicate is found. After the proof, subsequent users with the same file will be provided a pointer from the server without needing to upload the same file.

However, previous deduplication systems cannot support differential authorization duplicate check [8]. In an authorized deduplication system, each user is issued a set of privileges during system initialization. Each file uploaded to the cloud is also bounded by a set of privileges to specify which kind of users is allowed to perform the duplicate check and access the files. Before submitting his duplicate check request for some file, the user needs to take this file and his/her privileges as inputs. The user is able to find a duplicate for this file if and only if there is a copy of this file and a matched privilege stored in cloud.

II. REVIEW OF LITERATURE

Deduplication is actively used by a number of cloud backup providers (e.g. Bitcasa) as well as various cloud services provider (e.g. Dropbox) [9]. Jin et al. [7], today's commercial cloud storage services, such as Mozy, Memopal, have been applying deduplication to user data to save maintenance cost. If it was possible, IT organizations would only protect the unique data from their backups. Instead of saving everything repeatedly, the ideal scenario is one where only the new or unique content is saved. Data deduplication provides this basic capability.

A. Data De-Duplication Types [4]:

a. File-level de-duplication :

It is commonly known as single-instance storage. File-level data deduplication compares a file that has to be archived or backup that has already been stored by checking all its attributes against the index. The index is updated and stored only if the file is unique, if not then only a pointer to the existing file that is stored references. Only the single instance of file is saved in the result and relevant copies are replaced by "stub" which points to the original file.

b. Block-level de-duplication :

Block-level data deduplication operates on the basis of sub-file level. As the name implies, that the file is being broken into segments blocks or chunks that will be examined for previously stored information vs. redundancy. The popular approach to determine redundant data is by assigning identifier to chunk of data, by using hash algorithm. Files can be broken into fixed length blocks or variable length blocks. Fixed-size chunking [10] is conceptually simple and fast. However, this method has an important drawback. When a small amount of data is inserted into a file or deleted from a file, an entirely different set of chunks is generated from the updated file. To effectively address this problem, variable-size chunking, which is also known as content-based chunking, has been proposed. The Basic Sliding Window algorithm can be used for chunking [10].

B. Working of Deduplication Technology

Deduplication Technology can work in the following manner:

a. Firstly divide the input data into chunks or blocks.
b. Hash value for each of the block need to be calculated.
c. The values that are used to determine whether the blocks of same data is already stored.
d. Replace the redundant data with the reference or pointers to the block that is already in database.

Once the data is divided into chunks, by the results that is obtained index can be created and the redundant data that will be found is removed. Only a single copy of every chunk will be stored. Hash collisions are potential problem with de-duplication. The piece of data when it receives the hash number, the number is compared with the index of other already existing hash numbers. Some of the algorithms can be used to find the hash numbers for example, SHA, MD than it is encrypted using the AES algorithm and then data is outsourced to the cloud environment.

SHA-1 creates the required cryptographic signatures for security applications. A 160-bit value of SHA-1 creates that is unique for each piece of data. *MD5* - is also designed for cryptographic purposes it creates a 128-bit hash. Hash collisions usually will occur when two different chunks will produce the same hash value. The chances of this are very less indeed, but SHA-1 is considered to be most secure of the two algorithms that are above said.

16 KB Data Block 1	2181f384e82f5b5df72177853e093ad3
16 KB Data Block 2	a3745ae5bcdc3f53f87ce4beb792b123
16 KB Data Block 3	50cc0b0aec1c121ad3aa6b7b6d1d9e0d
16 KB Data Block 4	9e293ae056e53f4cc4c3b768de0cce9f
16 KB Data Block 5	2181f384e82f5b5df72177853e093ad3
16 KB Data Block 6	1c96f7a1e9bf91f9175fba4db912d3f0
16 KB Data Block 7	fb4d6a63feb1cda57a1be200037f4092
16 KB Data Block 8	13634ce69fb2380915e17ae30a9fbebf

Block 1 and 5 are the same, so one can be eliminated.

Figure 1. Generation of Hash values for data blocks.

C. Examples of deduplication Storage System [1]:

Data deduplication is commonly performed on secondary storage systems such as archival and backup storage.

1. *Venti:* It is a network storage system. It applies identical hash values to find block contents so that it decreases the data occupation of storage area. Venti generates blocks for huge storage applications and inspire a write-once policy to avoid collision of the data. This network storage system emerged in the early stages of network storage, so it is not suitable to deal with avast data, and the system is not scalable.
2. *HYDRAstor:* It is a scalable, secondary storage solution, which includes a back-end consisting of a grid of storage nodes with a decentralized hash index, and a traditional file system interface as a front-end. The back-end of HYDRAstor is based on Directed Acyclic Graph, which is able to organize large-scale, variable-size, content addressed, immutable, and highly-resilient data blocks. HYDRAstor detects duplications according to the hash table. The ultimate target of this approach is to form a backup system. It does not consider the situation when multiple users need to share files.
3. *Extreme Binning [11***]:** It is a scalable, paralleled deduplication approach aiming at a non-traditional backup workload which is composed of low-locality individual files. Extreme Binning exploits file similarity instead of locality, and allows only one disk access for blocks lookup per file. Extreme Binning arranges similar data files into bins and removes duplicated chunks inside each bin. Duplicates exist among different bins. Extreme Binning only keeps the primary index in memory in order to reduce RAM occupation.
4. *MAD2 [11]:* It is an accurate deduplication network backup service, which works at both the file level and the block level. This approach is designed for backup service not for a pure storage system.
5. *Duplicate Data Elimination (DDE***):** DDE applies a combination of content hashing, then copy-on-write, and lazy updates to get the functions of discovering and coalescing similar data blocks in a storage area network file system. It always works in the background.

D. A Secure Authorized Data Deduplication System:

In an authorized deduplication system, each user is issued a set of privileges during system initialization [8]. Each file uploaded to the cloud is also bounded by a set of privileges to specify which kind of users is allowed to perform the duplicate check and access the files.

There are three entities in the system that is, Owners/Users, Private cloud and Storage Cloud Service Provider (CSP) in Public cloud. The CSP performs deduplication by checking if the contents of two files are the same and stores only one of them. The access right to a file is defined based on a set of privileges. Each privilege is represented in the form of a short message called token. A user computes and sends duplicate-check tokens to the public cloud for authorized duplicate check. Users have access to the private cloud server, a semi-trusted third party which will aid in performing Deduplicable encryption by generating file tokens for the requesting users. The private keys for the privileges are managed by the private cloud, who answers the file token requests from the users. Users would try to access data either within or out of the scopes of their privileges. They have proposed a new Deduplication system supporting for Differential Authorization and Authorized Duplicate Check.

Differential Authorization: Each authorized user is able to get his/her individual token of his file to perform duplicate check based on his privileges [8]. Under this assumption, any user cannot generate a token for duplicate check out of his privileges or without the aid from the private cloud server.

Authorized Duplicate Check: Authorized user is able to use his/her individual private keys to generate query for certain file and the privileges he/she owned with the help of private cloud, while the public cloud performs duplicate.

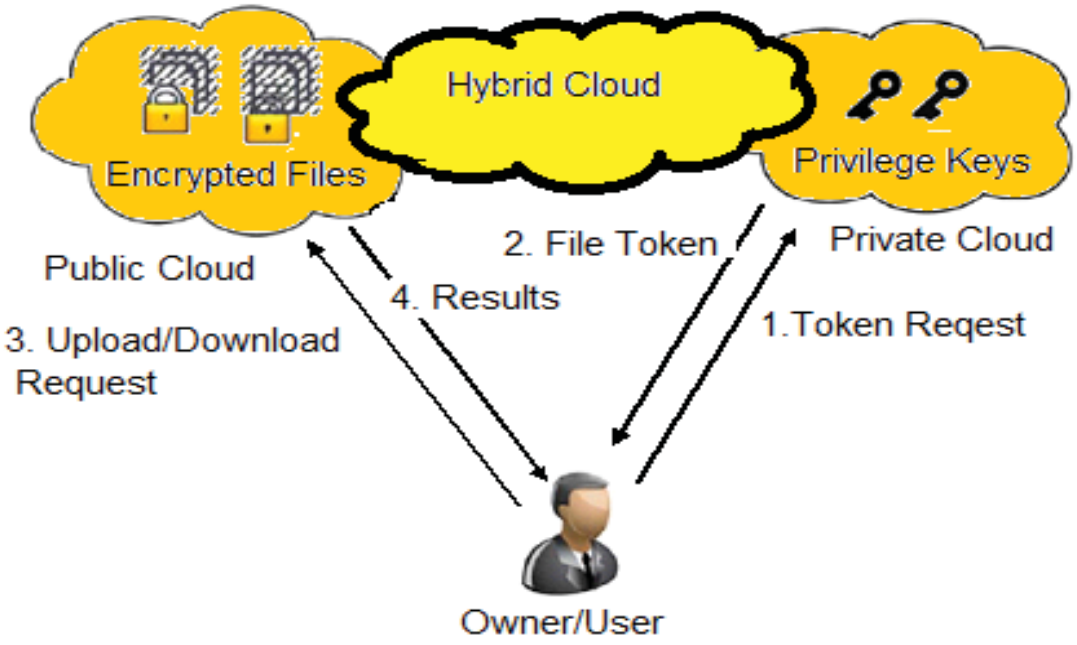

Figure 2. Architecture for Secure Data Deduplication

A Client program is used to model the data users to carry out the file upload process.

A Private Server program is used to model the private cloud which manages the private keys and handles the file token computation.

A Storage Server program is used to model the CSP which stores and Deduplicates files.

This can be implemented using cryptographic operations of hashing and encryption with the OpenSSL library [12] and communication between the entities based on HTTP, using GNU Libmicrohttpd [13] and libcurl [14]. Thus, users can issue HTTP Post requests to the servers.

User can upload and download the files from public cloud but private cloud provides the security for that data. That means only the authorized person can upload and download the files from the public cloud. User generates the key and stored that key onto the private cloud.

Data Users/Owner: This entity wants to outsource his/her data and later access it. User only uploads unique data but does not upload any duplicate data to save the upload bandwidth. This data may be owned by the same user or different users. In the authorized deduplication system, each user is issued a set of privileges in the setup of the system [8]. To realize the authorized deduplication with differential privileges, the files/blocks are protected with the convergent encryption keys and privilege keys.

At the Users/Owner:

- *FileTag(File):*Tag of a File is generated using SHA-1 algorithm.
- *TokenReq(Tag, UserID) :* This is request for the token by providing the tag and user ID to the private cloud.
- *DupCheckReq (Token):* To check the file/block is duplicated on public cloud or not.
- *ShareTokenReq(Tag,{Priv.}):* Request the private cloud for sharing the tag and providing the privileges to other users.
- *File Encrypt(File):* By using AES algorithm, The file is encrypted using convergent key which is the hash value of the original file and generated using SHA-256.
- *FileUploadReq(FileID,File,Token):* If file is not duplicate on public cloud then request to store the file on public cloud by providing FileID, Encrypted file and Token.

Private Cloud: To secure usage of cloud service, the entity is introduced in Traditional deduplication technique. Computing resources at data user/owner side are restricted and the public cloud is not fully trusted party. Private cloud provides data user/owner with an environment and infrastructure working as an interface between user/owner and the public cloud. The private keys for the privileges are managed by the private cloud, who answers the file token requests from the users [8]. The interface allows user to submit files/sub-files and queries to be securely stored and computed respectively.

At the Private Server:

- *TokenGen (Tag, UserID):* It loads the associated privilege keys of owner/user and generates a token by using HMAC-SHA-1 algorithm.
- *ShareTokenGen (Tag, {Priv.}):* It generates the share token with privilege key and sharing privileges set using HMAC-SHA-1 algorithm.

Cloud Service Provider (CSP): This entity provides data storage service in public cloud. CSP eliminate the duplicate data using deduplication and keep the unique data. For reducing the storage cost, the CSP eliminates the storage of redundant data via deduplication and keeps only unique data into the public cloud.

Cloud Storage Server:

- *DupCheck (Token):* It maps the generated token with existing tokens for duplicate check. If duplicate found then reply with "file duplicate" otherwise reply with "No File Duplicate" to owner/user.
- *FileStore(FileID, File, Token) :* It store the encrypted file , file ID and Token and maps the update.

Data Uploading- Suppose that a data owner U with privilege set P(U) wants to upload and share a file F with users who have the privilege set P(F) = { Pj}. The user computes and sends the file token $\phi'(F, p) = TagGen(F, kp)$ for all $p \in P(F)$ to the CSP.

1. If a duplicate is found by the CSP, the user proceeds proof of ownership of this file with the CSP. If the proof is passed, the user will be assigned a pointer, which allows him to access the file.

2. Otherwise, if no duplicate is found, the user computes the encrypted file CF = EncCE(kF , FILE) with the convergent key kF = KeyGenCE(FILE) and uploads the encrypted file along with File ID, Token and Privileges to the cloud server. The convergent key kF is stored by the user locally.

Data Retrieving- Suppose a user wants to download a file F. It first sends a request and the file name to the CSP. Upon receiving the request and file name, the CSP will check whether the user is eligible to download File F. If failed, the CSP sends back an abort signal to the user to indicate the download failure. Otherwise, the CSP returns the corresponding cipher text CF. upon receiving the encrypted data from the CSP; the user uses the key kF stored locally to recover the original file F. In this, each user will be issued private keys {kpi}pi ∈ P(U) for their corresponding privileges, denoted by P(U). These private keys can be applied by the user to generate file token for duplicate check.

F. Efficient and Reliable Convergent Key Management [7]:

Convergent encryption has been extensively adopted for secure deduplication; a critical issue of making convergent encryption practical is to efficiently and reliably manage a huge number of convergent keys. The first attempt to formally address the problem of achieving efficient and reliable key management in secure deduplication is a baseline approach, in which each user holds an independent master key for encrypting the convergent keys and outsourcing them to the cloud. However, the baseline approach suffers two critical deployment issues:

a. Inefficient:

As it will generate an enormous number of keys with the increasing number of users, although different users may share the same data copies, they must have their own set of convergent keys so that no other users can access their files. As a result, the number of convergent keys being introduced linearly scales with the number of blocks being

stored and the number of users. This key management overhead becomes more prominent if they exploit fine-grained block-level deduplication. For example, suppose that a user stores 1 TB of data with all unique blocks of size 4 KB each, and that each convergent key is the hash value of SHA-256, which is used by Dropbox for deduplication. Then the total size of keys will be 8 GB. The number of keys is further multiplied by the number of users. The resulting intensive key management overhead leads to the huge storage cost, as users must be billed for storing the large number of keys in the cloud under the pay-as-you-go model.

b. Unreliable:

As it requires each user to dedicatedly protect his own master key. If the master key is accidentally lost, then the user data cannot be recovered, if it is compromised by attackers, then the user data will be leaked.

To efficiently and reliably manages enormous convergent keys, while still achieving secure deduplication. A new construction proposed called Dekey in which users do not need to manage any keys on their own but instead securely distribute the convergent key shares across multiple servers. Security analysis demonstrates by that Dekey is secure in terms of the definitions specified in the proposed security model. Convergent encryption is the technique in cloud computing to remove duplicate files from storage without the provider having access to the encryption keys. Only the first user who uploads the data is required to compute and distribute such secret shares, while all following users who own the same data copy need not compute and store these shares again. To recover data copies, a user must access a minimum number of key servers through authentication and obtain the secret shares to reconstruct the convergent keys. In other words, the secret shares of a convergent key will only be accessible by the authorized users who own the corresponding data copy. This significantly reduces the storage overhead of the convergent keys and makes the key management reliable against failures and attacks.

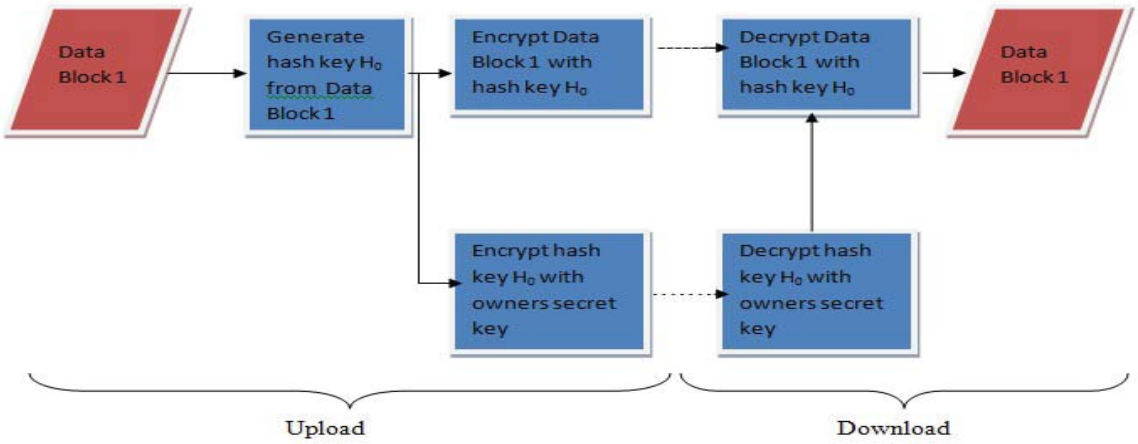

Figure 3. Baseline approach (keeping the hash key with an encryption scheme).

Dekey supports both file level and block level deduplication. Dekey is implemented using the Ramp secret sharing scheme that enables the key management to adapt to different reliability and confidentiality levels.

Ramp Secret Sharing (RSSS):

Dekey uses the RSSS [15], [16] to store convergent keys. Specifically, the (n > k > r) - RSSS (where n > k > r >= 0) generates n shares from a secret such that:

1. The secret can be recovered from any k shares but cannot be recovered from fewer than k shares, and
2. No information about the secret can be deduced from any r shares.

The (n > k > r)-RSSS builds on two primitive functions:

1. Share divides a secret S into (k – r) pieces of equal size, generates r random pieces of the same size, and encodes the k pieces using a non-systematic k-of-n ensure code into n shares of the same size;

2. Recover takes any k out of n shares as inputs and then outputs the original secret S.

Dekey uses RSSS to provide a key management mechanism to balance among confidentiality, reliability, storage overhead, and performance.

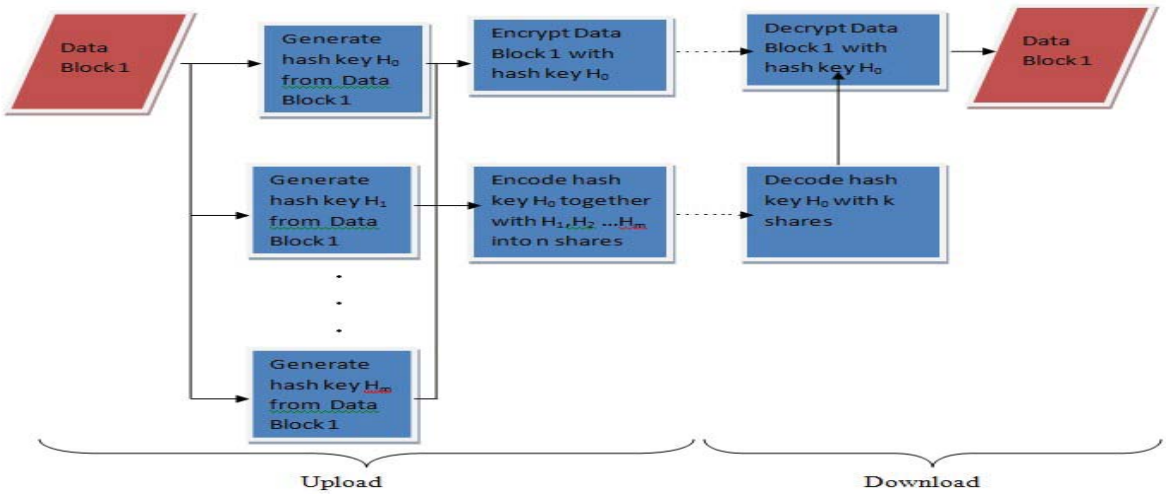

Figure 4. Dekey (keeping the hash key with (n, k, r)-RSSS).

Proof of Ownership (PoW): The notion of proof of ownership is to solve the problem of using a small hash value as a proxy for the entire file in client-side deduplication, PoW is implemented as an interactive algorithm run by a user and a storage server. The storage server derives a short value ∏ from a data copy M. User needs to send ∏' and run a proof algorithm with the storage server. If ∏=∏' then storage server allows otherwise abort the communication.

III. SUMMARY

Thus, the notion of secure data deduplication is to protect the data security by including differential privileges of users in the duplicate check. The owner/user is allowed to perform the duplicate check for files marked with the corresponding privileges. Deduplication can be performed on files and can also be performed on blocks. Deduplication can be done at client side, which reduces the upload bandwidth on the network. It can also perform at Target side. It can be done at online i.e. when owner uploading data that time also deduplication performed or can be offline, means after storing data on storage system deduplication is performed. By using deduplication technique, storage space can be saved by eliminating duplicate copies in the cloud storage system. Dekey applies deduplication among convergent keys and distributes convergent key shares cross multiple key servers, while preserving semantic security of convergent keys and confidentiality of outsourced data. Dekey Implementation using the Ramp secret sharing scheme that incurs small encoding/decoding overhead in the regular upload/download operations.

REFERENCES

[1] Deepak Mishra, Dr. Sanjeev Sharma, "Comprehensive study of data de-duplication", International Conference on Cloud, Big Data and Trust 2013, Nov 13-15.
[2] Gaurav Kakariya, Prof. Sonali Rangdale, "A Hybrid Cloud Approach For Secure Authorized Deduplication", International Journal of Computer Engineering and Applications, Volume VIII, Issue I, October 14.
[3] Daehee Kim, Sejun Song, Baek-Young Choi, "SAFE: Structure-Aware File and Email Deduplication for Cloud-based Storage Systems".
[4] Pooja S Dodamani, Pradeep Nazareth, "A Survey on Hybrid Cloud with De-Duplication", International Journal of Innovative Research in Computer and Communication Engineering, December 2014.
[5] Boga Venkatesh, Anamika Sharma, Gaurav Desai, Dadaram Jadhav, "Secure Authorised Deduplication by Using Hybrid Cloud Approach", November 2014.
[6] M. Bellare, S. Keelveedhi, and T. Ristenpart, " Message-locked encryption and secure deduplication", in Proc. IACR Cryptology ePrint Archive, 2012
[7] Jin. Li, Xiaofeng 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,June- 2014.
[8] Jin. Li, Yan Kit Li, Xiaofeng, P. Lee, and W. Lou., "A Hybrid Cloud Approach for Secure Deduplication" ,In IEEE Transactions on Parallel and Distributed Systems, 2014.
[9] Jan Stanek, Alessandro Sorniotti, Elli Androulakiy, Lukas Kencl, "A Secure Data Deduplication Scheme for Cloud Storage".
[10] Jaehong Min, Daeyoung Yoon, and Youjip Won, "Efficient Deduplication Techniques for Modern Backup Operation", IEEE Transactions on Computers, Vol. 60, No. 6, June 2011.
[11] T.Y.J. NagaMalleswari, D.Malathi, "Deduplication Techniques: A Technical Survey", International Journal for Innovative Research in Science & Technology, December 2014.
[12] OpenSSL Project. http://www.openssl.org/, April-2015.
[13] GNU Libmicrohttpd http://www.gnu.org/software/libmicrohttpd/, April-2015.
[14] libcurl. http://curl.haxx.se/libcurl/, April-2015.

[15] G.R. Blakley and C. Meadows, ‘‘Security of Ramp Schemes,’’ in Proc. Adv. CRYPTO, vol. 196, Lecture Notes in Computer Science, G.R. Blakley and D. Chaum, Eds., 1985, pp. 242-268.
[16] A.D. Santis and B. Masucci, ‘‘Multiple Ramp Schemes,’’ IEEE Trans. Inf. Theory, vol. 45, no. 5, pp. 1720-1728, July 1999.