ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 1, January 2018

Study of Data Deduplication Techniques on Cloud

Pooja Kharade[1], Zaheed Shaikh[2]

P.G. Student, Department of Computer Engineering, KJ Somaiya College of Engineering, Mumbai, India[1]

Professor, Department of Computer Engineering, KJ Somaiya College of Engineering, Mumbai, India[2]

ABSTRACT: Cloud storage is a remote storage service, where users can upload and download their data from anywhere and anytime. In the recent years, there had been a tremendous increase in the amount of digital data. This increased data occupies more storage space on cloud. Thus, the cloud storage server performs data compression i.e. data deduplication which eliminates the duplicate data and also saves the storage space. Also, cloud storage service poses challenges with respect to privacy and confidentiality of the data stored by the different users.

KEYWORDS: Cloud storage, data deduplication, confidentiality.

I. INTRODUCTION

Data deduplication is widely used data compression techniques. It is used to eliminate redundant copies of data stored on cloud. This saves the amount of storage space on the cloud. The data deduplication techniques also take into consideration the privacy and confidentiality of the stored data so that more number of users will be willing to store their data on cloud.

Some deduplication approaches operate at the file level, while others go deeper to examine data at a sub-file, or block, level. File-level data deduplication compares a file to be backed up or archived with those already stored by checking its attributes against an index. If the file is unique, it is stored and the index is updated; if not, only a pointer to the existing file is stored. The result is that only one instance of the file is saved.Block-level deduplication looks at the data block itself to see if another copy of this block already exists. If so, the second (and subsequent) copies are not stored on the disk/tape, but a link/pointer is created to point to the original copy. There are pros and cons associated with both file-level and block-level deduplication.

II.RELATED WORK

1.Hybrid Cloud Approach

.The hybrid cloud approach aims at solving the problem of data deduplication by using the concept of differential privileges of users [1]. A hybrid cloud architecture is taken into account in this approach which consists of a public as well as private cloud.In this system, the private cloud is incorporated as a proxy to allow data owner/users to securely perform duplicate data check with different privileges.

The system model of hybrid cloud approach consists of the following entities:

Storage cloud service provider (S-CSP): This entity provides storage service for data in the private cloud. It stores the data on the account of the users and provides the data on demand of the users. It eradicates storage of duplicate data copies via deduplication.

Data users: A user is the entity who wants to outsource their data to the S-CSP and demand it later. In this system, each user is provided with a set of privileges. Each and every file is secured with the convergent encryption key and privilege keys to identify the authorized deduplication with differential privileges.

Private cloud: This is new entity has been introduced into the system to promote user's secure usage of cloud storage. As the resources at data user/owner side are curbed and the public cloud is not fully trusted party in practice, private

DOI: 10.15680/IJIRCCE.2018. 0601047

cloud acts like an interface between user and the public cloud. It also provides execution environment to the data owner.The private cloud manages the private keys for the privileges and answers the file token requests from the users.

Convergent Encryption:

A user (or owner of data) formulates a convergent key from the genuine data and encrypts it with the same convergent key. The user also generates a tag of the data copy which is used to recognize duplicate copies. If two data copies are indistinguishable, then their tags are the same. To identify duplicate copies, first the user sends the generated tag to the server in order to check if the identical copy has been already stored. Convergent key as well as the tag will be stored on the server side and are independently formulated i.e. the tag cannot be used to formulate the convergent key.

A convergent key consist of the following four primary functions [1]:

- KeyGen(M): K is the key generation algorithm that generates a convergent key K from a data copy M.
- Enc(K,M): C is the symmetric encryption algorithm that takes both the convergent key K and the data copy M as inputs and then output a ciphertext C.
- Dec(K,C): M is the decryption algorithm that takes both the convergent key K and the ciphertext C as inputs and then outputs the original data copy M.
- TagGen(M): T(M) is the tag generation algorithm that takes input as the original data copy M and outputs a tagT(M).

System Architecture:

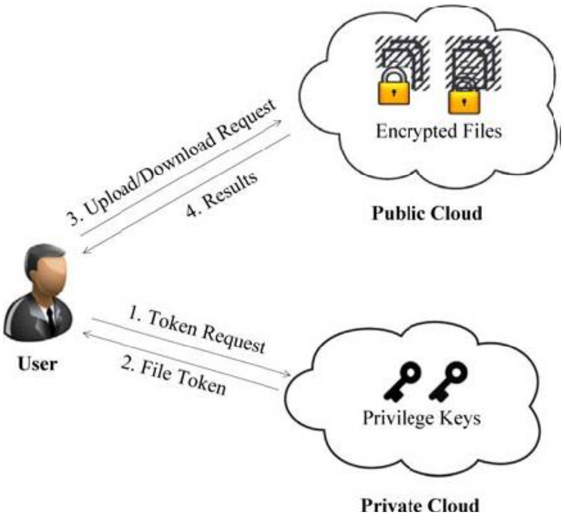

Figure 1. Hybrid Cloud Approach [1]

2. ClouDedup

ClouDedup provides block level data deduplication by coping with the inherent security exposures of convergent encryption. The security provided by ClouDedup confides on its new architecture which has a metadata manager and an additional server in addition to the basic storage provider. The server performs additional encryption to avoid well-known attacks against convergent encryption. The metadata manager is responsible for the key management task since block-level deduplication requires storage of huge number of keys.

DOI: 10.15680/IJIRCCE.2018. 0601047

ClouDedup has of two basic components: a server which is in charge of access control and that achieves the main protection against COF and LRI attacks. The metadata manager i.e. MM is in charge of the data deduplication and key management operations.

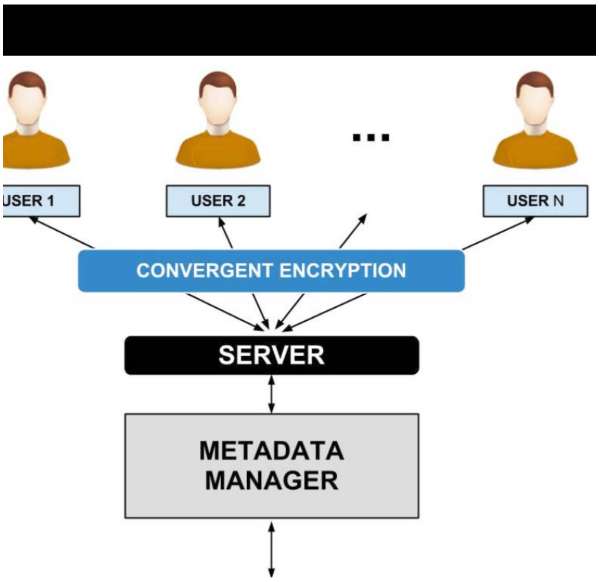

Figure 2.ClouDedup Architecture [2]

User:
The role of the user is limited to splitting files into blocks, encrypting them with the convergent encryption technique, signing the resulting encrypted blocks and creating the storage request. In addition, the user also encrypts each key derived from the corresponding block with the previous one and his secret key in order to outsource the keying material as well and thus only store the key derived from the first block and the file identifier.

Server:
The server has three main roles: authenticating users during the storage/retrieval request, performing access control by verifying block signatures embedded in the data, encrypting/ decrypting data traveling from users to the cloud and viceversa. The server takes care of adding an additional layer of encryption to the data (blocks, keys and signatures) uploaded by users. Before being forwarded to MM, data are further encrypted in order to prevent MM and any other component from performing dictionary attacks and exploiting the well-known weaknesses of convergent encryption. During file retrieval, blocks are decrypted and the server verifies the signature of each block with the user’s public key. If the verification process fails, blocks are not delivered to the requesting user.

Metadata Manager(MM):
MM is the component responsible for storing metadata, which include encrypted keys and block signatures, and handling deduplication. Indeed, MM maintains a linked list and a small database in order to keep track of file ownerships, file composition and avoid the storage of multiple copies of the same data segments.

The tables used by MM are structured as follows:
• File table. The file table contains the file id, file name, user id and the id of the first data block.
• Pointer table. The pointer table contains the block id and the id of the block stored at the cloud storage provider.

- Signature table. The signature table contains the block id, the file id and the signature.

Cloud Service Provider:
SP is the most simple component of the system. The only role of SP is to physically store data blocks. SP is not aware of the deduplication and ignores any existing relation between two or more blocks. Indeed, SP does not know which file(s) a block is part of or if two blocks are part of the same file.

3. Message Locked Encryption

Message Locked Encryption i.e MLE was introduced by Douceur et al. [3]. In this system, the user first generates a key K by applying a cryptographic hash function H to M (M is the file's data), and then generates the ciphertext C by using symmetric encryption scheme. A second client B encrypting the same file M will produce the same C, enabling deduplication.

An MLE scheme MLE = $(\mathcal{P},\mathcal{K},\mathcal{E},\mathcal{D},\mathcal{T})$ is a five-tuple algorithm. On input 1^λ the parameter generation algorithm $\mathcal{P}$ returns a public parameter P. On input P and a message M, the key-generation algorithm $\mathcal{K}$returns a message-derived $K \leftarrow_{\$} \mathcal{K}_P(M)$. On inputs P,K,Mthe encryption algorithm $\mathcal{E}$ returns a ciphertext $C \leftarrow_{\$} \mathcal{E}_P(K,M)$. On inputs P,Kand a ciphertextC, the decryption algorithm $\mathcal{D}$ returns $\mathcal{D}_P(K,C)$. On inputs P,Cthe tag generation algorithm returns a tag $T \leftarrow \mathcal{T}_P(C)$ [3].

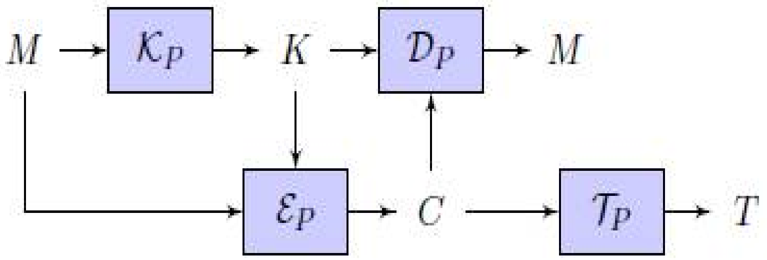

Figure 3. Depiction of syntax of MLE [3]

4. DupLESS

DupLESS [4] was proposed by Bellare et al. This technique provides secure deduplicated storage which resists brute force attacks. In Dup-LESS, a group of users/clients (e.g., company employees) encrypt their data by using a Key Server (KS). This KS that is distinct from a Storage Service (SS), which stores data. Clients do not expose any information about their data to the KS. So, as long as the KS remains obscured to attackers, high security can be assured.

DupLESS starts with the observation that brute-force ciphertext recovery in a CE-type scheme can be dealt with by using a key server (KS) to derive keys, instead of setting keys to be hashes of messages. Access to the KS is preceded by authentication, which stops external attackers. The increased cost slows down brute-force attacks from compromised clients, and now the KS can function as a (logically) single point of control for implementing rate-limiting measures. It is expected that by scrupulous choice of rate-limiting policies and parameters, brute-force attacks originating from compromised clients will be rendered less effective, while normal usage will remain unaffected [4].

A secret-parameter MLE which is an extension to MLE which endows all clients with a systemwide secret parameter *sk*(see Section 4). The rationale here is that if *sk*is unknown to the attacker, a high level of security can be achieved (semantic security, except for equality), but even if *sk*is leaked, security falls to that of regular MLE. A server-aided MLE scheme then is a transformation where the secret key is restricted to the KS instead of being available to all clients. One simple approach to get server-aided MLE is to use a PRF F, with a secret key *K* that never leaves the KS. A client would send a hash *H* of a file to the KS and receive back a *message-derived* key *K′* ← F(*K*,*H*). The other steps are as in CE. However, this approach proves unsatisfying from a security perspective. The KS here becomes a single point of failure, violating our goal of compromise resilience: an attacker can obtain hashes of files after gaining access to the KS, and can recover files with brute-force attacks. Instead, DupLESS employs an oblivious PRF (OPRF) protocol between the KS and clients, which ensures that the KS learns nothing about the client inputs or the resulting PRF outputs, and that clients learn nothing about the key [4].

Thus, a client, to store a file *M*, will engage in the RSA OPRF protocol with the KS to compute a messagederived key *K*, then encrypt *M* with *K* to produce a ciphertext*C*data. The client's secret key will be used to encrypt *K* to produce a key encapsulation ciphertext*C*key. Both *C*key and *C*data are stored on the SS. Should two clients encrypt the same file, then the message-derived keys and, in turn, *C*data will be the same , the key encapsulation *C*key will differ, but this ciphertext is small [4].

5. SecDep

User Aware Convergent Encryption is used in SecDep [5]. This technique decreases computation overhead and resists brute force attack. It also uses Multi-Level Key Management to eliminate the key space overheads (as file level key used for block level encryption) and splits file-level keys into share-level keys which is share with multiple servers to assure security and reliability of file-level keys.

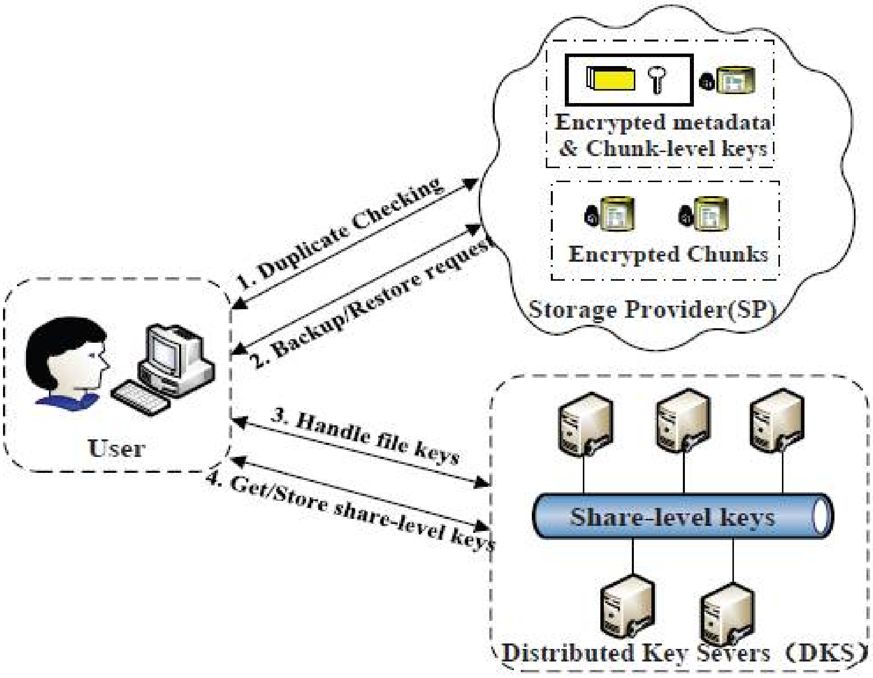

Figure 4.System Model of SecDep [5]

DOI: 10.15680/IJIRCCE.2018. 0601047

As shown in Figure 4.,SecDep consists of Users, a Storage Provider (SP), and Distributed Key Servers (DKS). When a user wants to access the DKS and the SP, his/her passwords and credentials should be verified at first. Chunks and chunk-level keys are encrypted and stored on the SP. Filelevel keys are securely divided into share-level keys via Shamir Secret Sharing Scheme (SSSS). Share-level keys are stored separately on the DKS. Data stored on the DKS and the SP are protected by some access control policies [5].

User : user is an entity who wants to upload data to (download data from) the Storage Provider (SP). The user applies variants of CE to protecting the data chunks and keys. To resist brute-force attacks and provide data confidentiality, the user accesses the DKS to add secret for generating the random file-level keys.

Storage Provider (SP): The storage providers mainly offer computation and storage services. The SP maintains tag indices for chunk-level and file-level duplicate checking. The SP also stores ciphertexts of chunks and chunk-level keys.

Distributed Key Servers (DKS): The DKS is built on a quorum of key servers via Shamir Secret Sharing Scheme (SSSS) to ensure security of keys. The user splits filekeys into *w* shares via SSSS (*w*, *t*). Each key server is a standalone entity that adds secret for key generations and stores users' key shares.

SecDep employs User-Aware Convergent Encryption (UACE) and Multi-Level Key management (MLK) approaches. First, UACE performs cross-user deduplication at file-level that encrypts files with the server-aided CE key. Meanwhile, UACE performs inside-user deduplication at chunk level that encrypts chunks with the user-aided CE key. Second, MLK encrypts chunk-level keys with the corresponding file-level key and splits file-level key into secure key shares via Shamir Secret Sharing Scheme, and sends them to the Distributed Key Servers [5].

User-Aware Convergent Encryption :

In order to resist brute-force attacks and reduce computation (time) overheads, UACE combines cross-user file-level and inside-user chunk-level secure deduplication, and exploits different secure policies for better performance. In general, (1) UACE firstly performs cross user file-level Hash Convergent Encryption (HCE) when each user backups their files. For each file, the user computes a file-level key and file tag via server-aided HCE. The user sends file tag to the SP and searches file tag in the global file-tag index, and then the SP returns the file deduplication results to the user. (2) If it is not a duplicate file, it will be divided into several data chunks. The CE keys and tags of these chunks will be computed by performing user-aided Convergent Encryption (CE). The user then sends the chunk tags to the SP. The SP will check whether the chunk tags are existed in the tag index of this user, and return the duplicate-checking results to the user. Then the user encrypts all the unique chunks and sends ciphertexts of chunks to the SP [5].

Role of the DKS:

The function of the DKS are twofold, namely, aiding secret information to generate random file-level keys, and storing share-level keys. (1) To resist brute-force attacks, key servers aid the users to generate file-level keys by adding secret information via RSA-OPRF. Users do not trust the DKS completely. Key servers do not know the file hash *HF* and file key *KF* because the user blinds the above data. (2) In order to ensure key security and avoid singlepoint- of-failure of file-level keys, MLK splits file-level keys into share-level keys and sends them to the DKS via a secure channel [5].

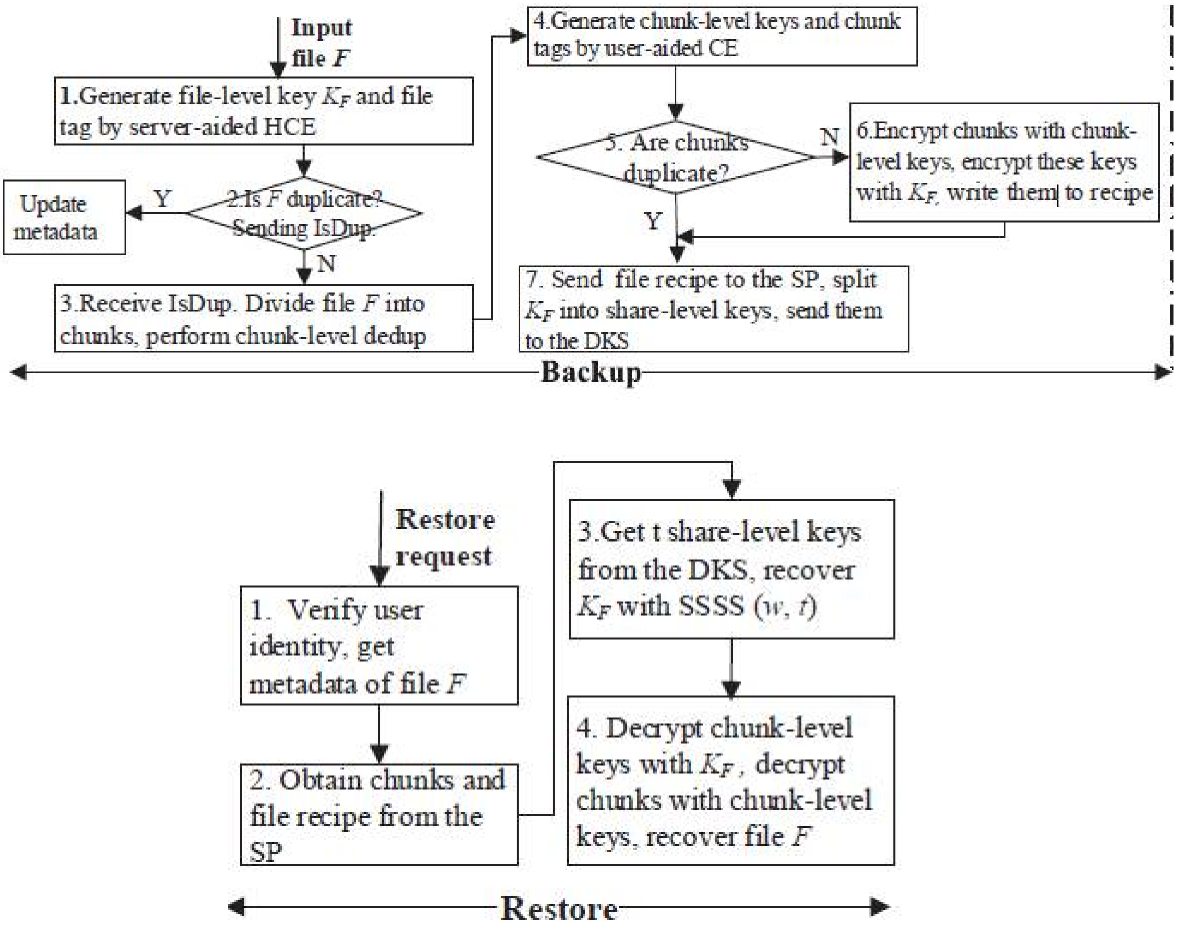
Input file F
1.Generate file-level key K_F and file tag by server-aided HCE
2.Is F duplicate? Sending IsDup
Y
Update metadata
N
3.Receive IsDup. Divide file F into chunks, perform chunk-level dedup
4.Generate chunk-level keys and chunk tags by user-aided CE
5. Are chunks duplicate?
N
6.Encrypt chunks with chunk-level keys, encrypt these keys with K_F, write them to recipe
Y
7. Send file recipe to the SP, split K_F into share-level keys, send them to the DKS
Backup
Restore request
1. Verify user identity, get metadata of file F
2. Obtain chunks and file recipe from the SP
3.Get t share-level keys from the DKS, recover K_F with SSSS (w, t)
4. Decrypt chunk-level keys with K_F, decrypt chunks with chunk-level keys, recover file F
Restore

Figure 5.Workflow of SecDep including backup and restore protocol [5]

6. Dekey

Dekey [6] is a technique which overcomes the problem of achieving efficient and reliable key management. Ramp secret sharing scheme is used in this technique.Thus key management at different reliability and confidentiality levels is achieved. It also uses key management and convergent encryption method to provide security to the data. Dekey provides both file as well as block level deduplication.

Dekey is designed to efficiently and reliably maintain convergent keys. Its idea is to enable deduplication in convergent keys and distribute the convergent keys across multiple KM CSPs. Instead of encrypting the convergent keys on a per-user basis, Dekey constructs secret shares on the original convergent keys (that are in plain) and distributes the shares across multiple KM-CSPs. If multiple users share the same block, they can access the same corresponding convergent key. This significantly reduces the storage overhead for convergent keys. In addition, this approach provides fault tolerance and allows the convergent keys to remain accessible even if any subset of KMCSPs fails [6].

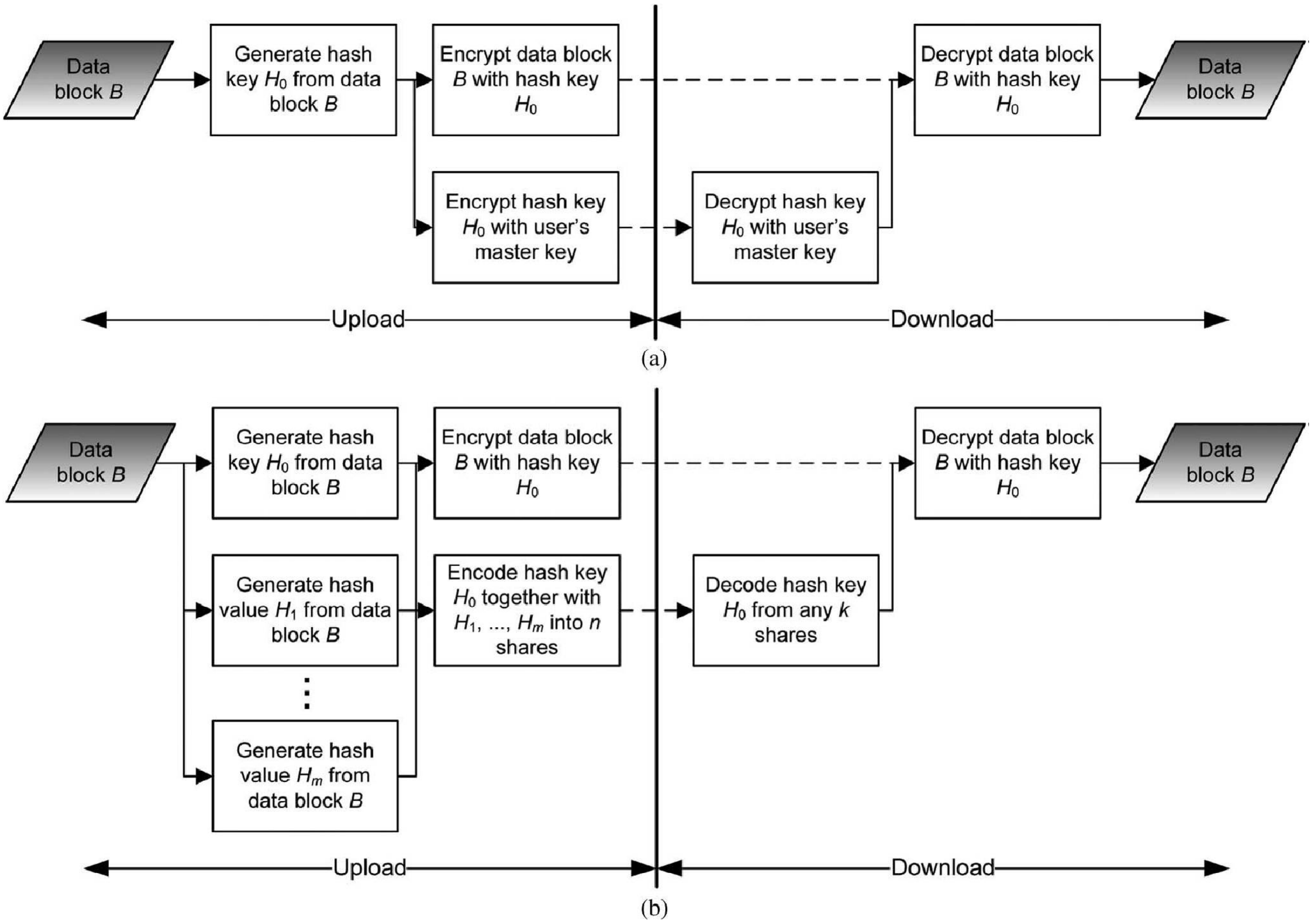
Data block B
Generate hash key H_0 from data block B
Encrypt data block B with hash key H_0
Encrypt hash key H_0 with user's master key
Decrypt hash key H_0 with user's master key
Decrypt data block B with hash key H_0
Data block B
Upload
Download
(a)
Data block B
Generate hash key H_0 from data block B
Generate hash value H_1 from data block B
Generate hash value H_m from data block B
Encrypt data block B with hash key H_0
Encode hash key H_0 together with H_1, ..., H_m into n shares
Decode hash key H_0 from any k shares
Decrypt data block B with hash key H_0
Data block B
Upload
Download
(b)

Figure 6. (a) Baseline Approach (b) Dekey [6]

III.SUMMARY

Summary of data deduplication techniques:

Sr.No.	Data Deduplication Technique	Methodology Used	Summary
1.	Message Locked Encryption	Convergent encryption i.e. the key under which encryption and decryption are performed is itself derived from the message.	Provides definitions both for a form of integrity and privacy which is called tag consistency. Convergent encryption leads to significant number of convergent keys which are difficult to manage with increasing user. Affected by brute-force attack.

2.	ClouDedup	Uses Convergent encryption (SHA-256 hash function) with an additional level of encryption and metadata manager for key management.	Supports block level deduplication. The key server suffers from a single point of failure risk.
3.	DupLESS	Employs convergent encryption (Uses a key server to obtain the message-based keys) and RSA-OPRF protocol	Resists brute force attack. Operations are time consuming and thus has large computational overheads for chunk level
4.	Dekey	Uses convergent encryption m and Ramp secret sharing scheme that enables key management	Supports both file-level and block level deduplication. Convergent keys are distributed across Multiple servers but the key servers are limited. Suffers Key space overhead.
5.	SecDep	Employs User Aware Convergent Encryption and Multi-Level Key Management	Resists brute force attack . Time overhead comes with multi-level key management.
6.	Hybrid Cloud Approach	SHA-1 algorithm is used to identify duplicate data Convergent encryption by using SHA-256 hash key using AES algorithm	Proof Of Ownership (PoW scheme) is used. Convergent encryption is prone to dictionary attacks, confirmation-of-file attack, learning remaining information attack

Table 1 :Summary of data deduplication techniques

IV.CONCLUSION

Deduplication is a method available in cloud storage for saving bandwidth and storage capacity. But, deduplication is less feasible with encrypted data since, different key encryptions convert same data into different formats. In this paper various methods are discussed where deduplication methods are carried out on encrypted data in a large storage area. Most of the methods studied here work on the basis of convergent encryption. In this information dense world, we cannot compromise on both security and duplication of data across storage areas. A strategy needs to be formulated which will enhance storage optimization without negotiating on encryption method; by providing deduplication technique in data storage servers where the available data is encrypted.

REFERENCES

[1] Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P.C. Lee, and Wenjing Lou, "A Hybrid Cloud Approach for Secure Authorized Deduplication," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 26, NO. 5, MAY 2015.

[2] P. Puzio, R. Molva, M. Onen, and S. Loureiro, "ClouDedup: Secure deduplication with encrypted data for cloud storage," in Proc. IEEE Int. Cof. Cloud Comput. Technol. Sci., 2013, pp. 363–370, doi:10.1109/CloudCom.2013.54.

[3] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in Proc. Cryptology—EUROCRYPT, 2013, pp. 296–312, doi:10.1007/978-3-642-38348-9_18.

[4] M. Bellare, S. Keelveedhi, and T. Ristenpart, "DupLESS: Server aided encryption for deduplicated storage," in Proc. 22nd USENIX Conf. Secur., 2013, pp. 179–194.

[5] Yukun Zhou, Dan Feng, Wen Xia, Min Fu, Fangting Huang, Yucheng Zhang, Chunguang Li, "SecDep: A User-Aware Efficient Fine-Grained Secure Deduplication Scheme with Multi-Level Key Management", IEEE Mass Storage Systems and Technologies (MSST) 2015 31st Symposium, Year - 2013

[6] Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management", IEEE Transactions on Parallel and Distributed Systems vol. 25(6), Year – 2014

[7] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A secure data deduplication scheme for cloud storage," Tech. Rep. IBM Research, Zurich, ZUR 1308-022, 2013.