



# Secure Deduplication Using Distributed Convergent Key Management

Nilam Patil<sup>1</sup>, B. K. Sarkar<sup>2</sup>

M.E. Student, Dept. of Computer Engineering, PVPIT, Bavdhan, Pune, India<sup>1</sup>

Professor, Dept. of Computer Engineering, PVPIT, Bavdhan, Pune, India<sup>2</sup>

**ABSTRACT:** Data deduplication is a specific form of compression where redundant data is eliminated, and has been used in cloud computing to improve storage utilization. In the deduplication process, duplicate data is deleted, leaving only one copy of the data to be stored. In the traditional encryption, different users required to encrypt their data with own keys. So different ciphertexts are produced, it makes difficult to deduplication. The new scheme has been proposed, convergent encryption, to protect confidentiality of sensitive data while making deduplication is possible. In this paper this convergent key manage efficiently and reliability. Firstly in this paper baseline approach is used in which a master key means encrypted convergent key is kept individually and outsourcing them to cloud. Each user has master key, increase the enormous number of keys. And users also want to protect the master key. A new approach is proposed in which Dekey is used in which user share key on multiple server, he do not require manage the key. Ramp Secret Sharing scheme is used to implement Dekey in which each key distributed on multiple server.

**KEYWORDS:** Convergent encryption, private sharing, storage system, cloud environment.

## I. INTRODUCTION

Today days many enterprises and organization outsource data to cloud environment. It is critical challenge of today's cloud storage to manage large volume of data. According to the analysis report, the volume of data in the wild is expected to reach 40 trillion gigabytes in 2020 [3]. Data deduplication is one of the techniques for reducing the amount of storage space an organization needs to save its data. In most organizations, the storage systems contain duplicate copies of many pieces of data. For example, the same file may be saved in several different places by different users, or two or more files that aren't identical may still include much of the same data. Deduplication eliminates these extra copies by saving just one copy of the data and replacing the other copies with pointers that lead back to the original copy. Data deduplication can generally operate at the file or block level. File deduplication eliminates duplicate files (as in the example above), but this is not a very efficient means of deduplication. Block deduplication looks within a file and saves unique iterations of each block. Each chunk of data is processed using a hash algorithm such as MD5 or SHA-1. This process generates a unique number for each piece which is then stored in an index. If a file is updated, only the changed data is saved. That is, if only a few bytes of a document or presentation are changed, only the changed blocks are saved; the changes don't constitute an entirely new file. This behaviour makes block deduplication far more efficient. However, block deduplication takes more processing power and uses a much larger index to track the individual pieces.

From a user's perspective, data outsourcing raises security and privacy concerns. We must trust third-party cloud providers to properly enforce confidentiality, integrity checking, and access control mechanisms against any insider and outsider attacks. Traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different ciphertexts, making deduplication impossible [1]. Convergent encryption [2] has been proposed to enforce data confidentiality while making deduplication feasible. Data copy is encrypts/decrypts using convergent key. Convergent key is computed by cryptographic hash value of the content of the data copy. When key generated and data encrypted, users retain the keys and send the cipertext to the cloud. Since the encryption operation is deterministic and is derived from the data content, identical data copies will generate the same convergent key and hence the same cipertext [1]. User consider Baseline approach in which original data copy is first encrypted with a convergent key and convergent key is encrypted by a master key that will be kept



# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirce.com](http://www.ijirce.com)

Vol. 5, Issue 6, June 2017

locally and securely by each user. The master key can be used to recover the encrypted keys and hence the encrypted files. In this way, the master key and the metadata about the outsourced data are needs to keep by each user.

## II. RELATED WORK

J. Li, X. Chen, M. Li, J. Li[1], new approach to securely share convergent key on different cloud service provider with secure deduplication of data. G. Wallace, F. Douglis [2], a secure overlay cloud storage system that achieves fine-grained, policy-based access control and file assured deletion. It associates outsourced files with file access policies, and assuredly deletes files to make them unrecoverable to anyone upon revocations of file access policies. Q. Wang, C. Wang [3]. Data integrity auditing and storage deduplication are achieved simultaneously.

Our proposed scheme is also characterized by constant real-time communication and computational cost on the user side. Cloud computing based storage services have rapidly spread in the market due to their promising capabilities and features. However, the security challenge of outsourcing sensitive data for sharing on the cloud which is not fully controlled by the data owners is still open. W. Wang, B. Bargava [4], we design a cipher text-policy attribute-based encryption (ABE) scheme and a proxy re-encryption scheme. Based on them, we further propose a secure, efficient and fine-grained data Access Control mechanism for P2P storage cloud named ACPC.

## III. EXISTING SYSTEM

### A. Introduction:

In existing system data De-duplication is not scalable. It suffers in two issues first; it is inefficient, as it will generate an enormous number of keys with the increasing number of users. Specifically, each user must associate an encrypted convergent key of each block with encrypted data copies. 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. 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 Drop box for deduplication [4]. 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 [1] Second, the baseline approach is unreliable, each user dedicatedly want to secure his own master. 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.

### B. Proof of Ownership

The proof of ownership (PoW) with users to prove their ownership of data copies to the storage server. This proof mechanism in PoW provides a solution to protect the security in client-side deduplication. In this way, a client can prove to the server that it indeed has the file. Notations of  $PoW_F$  and  $PoW_B$  to denote PoW for a file F and block B respectively. Steps for check the Ownership of file:

- User send  $\phi(f)$  to sever side
- Server derives  $\phi'(f)$
- Proof is correct if  $\phi(f)=\phi'(f)$

## IV. BASELINE APPROACH

The baseline approach involves only the user and the S-CSP (i.e., no KM-CSPs are required). Its idea is that each user has all his data copies encrypted by the corresponding convergent keys, which are then further encrypted by an independent master key. The encrypted convergent keys are outsourced to the S-CSP, while the master key is securely maintained by the user.

### A. System Setup

The system setup phase initializes the necessary parameters in the following steps:

S1: The following entities are initialized: 1) a symmetric encryption scheme with the primitive functions ( $KeyGen_{SE}$ ,  $Encrypt_{SE}$ ,  $Decrypt_{SE}$ ) and the user's master key  $k = KeyGen_{SE}(1^\lambda)$  for some security parameter  $1^\lambda$ ; 2) A convergent

# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirce.com](http://www.ijirce.com)

Vol. 5, Issue 6, June 2017

encryption scheme with the primitive functions ( $KeyGen_{CE}$ ,  $Encrypt_{CE}$ ,  $Decrypt_{CE}$ ,  $TagGen_{CE}$ ); and 3) A PoW algorithm  $PoW_F$  for the file and a PoW algorithm for the block, which is denoted by  $PoW_B$ .

S2: The S-CSP initializes two types of storage systems: a rapid storage system for storing the tags for efficient duplicate checks, and a file storage system for storing both encrypted data copies and encrypted convergent keys. Both storage systems are initialized to be  $\perp$ .

## B. File Upload

Suppose that a user uploads a file  $F$ . First, it performs file level and block level deduplication as see Fig.1.

- 1) User computes and sends the file tag to server.
- 2) Server checks whether the same tag exists. If so server replies the user with response “file duplicate” or “no file duplicate” otherwise.
- 3) If response is “file duplicate”, then user runs  $PoW_B$  on  $F$  with server. If response is “no file duplicate”, then users it proceeds with block level duplication.

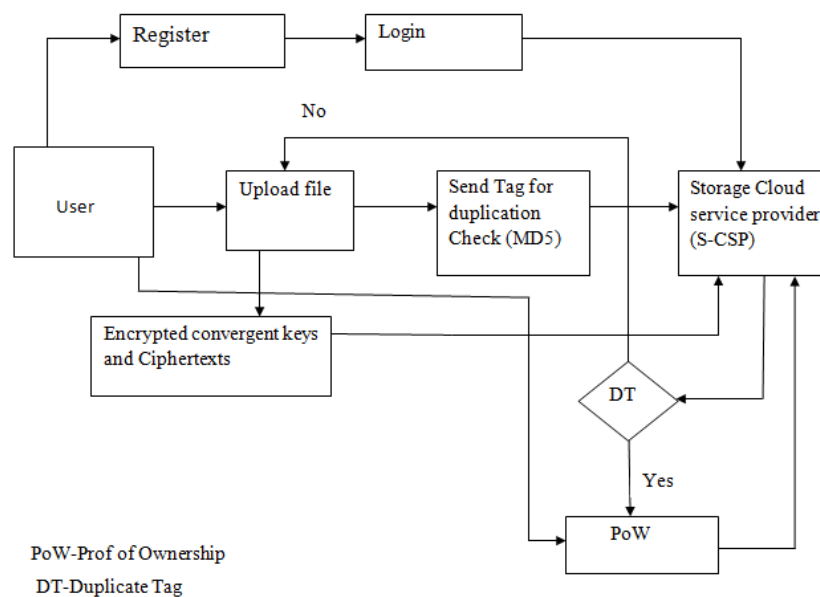


Fig. 1: Architecture of Upload file for Baseline Approach

- 4) If response is “file duplicate”, then user runs  $PoW_B$  on  $F$  with server. If response is “no file duplicate”, then users it proceeds with block level duplication.
- 5) Divide the input file  $F$  into set of block  $B_i$  (where  $i=1, 2, \dots$ ). Compute block tag and send to server for duplicate checks. Server checks whether the same tag exists for each block. If so server replies the user with response “block duplicate” or “no block duplicate” otherwise.
- 6) If response is “block duplicate”, then user runs  $PoW_B$  on  $B_i$  with server. Otherwise it encrypts the block with convergent key of that block.
- 7) Encrypt all convergent keys with master key.
- 8) User uploads unique blocks, encrypted convergent keys and file tag to the server.

## C. File Download

Suppose a user wants to download a file  $F$ . It first sends a request and the file name to the S-CSP and performs the following steps with the help of Fig. 2.

- 1) User sends request for file  $F$  along with file name.

# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirce.com](http://www.ijirce.com)

Vol. 5, Issue 6, June 2017

- 2) Server will check whether the user is eligible to download F. If user is authenticated, it returns corresponding Ciphertexts and encrypted convergent keys to the user.
- 3) User uses master keys to recover each convergent key. Then it uses each convergent key to recover original block.

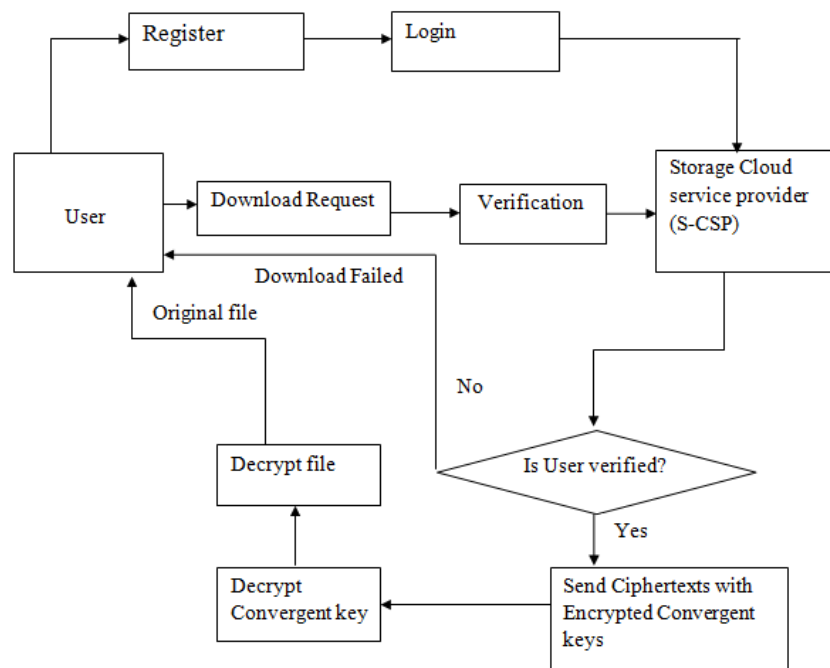


Fig. 2: Architecture of download file for Baseline Approach

## D. Limitations

The first problem is the enormous storage overhead in key management. Each user has encrypted convergent keys with his master key for each data copy. The encrypted convergent keys are different across users due to the different master keys. Thus the number of convergent keys increases linearly with the number of unique data copies being stored and the number of users, here by imposing heavy storage overhead. Another problem is that the master key presents the single-point-of failure and needs to be securely and reliably maintained by the user.

## V. PROPOSED SYSTEM

### A. Introduction

Propose a new construction called Dekey, which provides efficiency and reliability, guarantees for convergent key management on both user and cloud storage sides. The idea is to apply deduplication to the convergent keys and leverage secret sharing techniques. This paper makes the following contributions:

- A new construction Dekey is proposed to provide efficient and reliable convergent key management through convergent key deduplication and secret sharing.
- Security analysis demonstrates that Dekey is secure in terms of the definitions specified in the proposed security Model.
- Dekey using the Ramp secret sharing scheme that enables the key management to adapt to different reliability and confidentiality levels.

## B. Architecture

DeKey concept do not relay on Master Key Concept. The concept of deduplication effectively and security is achieved by means of Proof of Ownership of the file. The convergent keys outsource to third party key Management server securely. Architecture of DeKey is shown in Fig. 3.

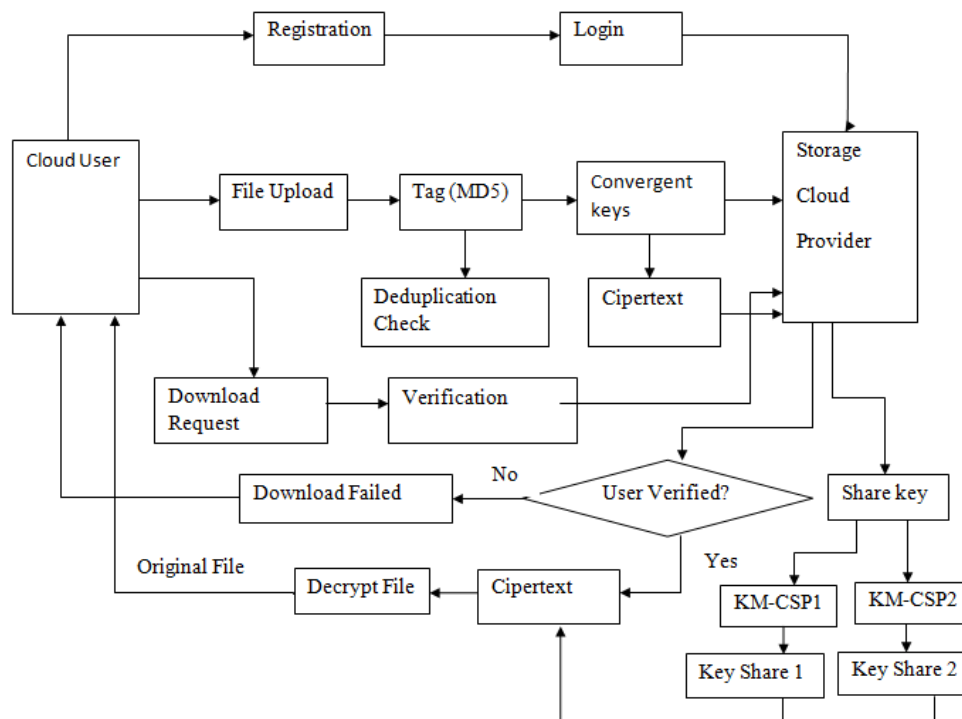


Fig. 3: Architecture of Dekey

DeKey supports both file-level and block level deduplication. Key Management overhead is avoided and provides fault tolerance guarantees for key management, while preserving the required security properties of secure deduplication. Instead of using normal encryption and decryption, both SHA-256 and AES-256 algorithms are used as the plain text is encrypted with the convergent key so that our data will be secured. Scalability increases as DeKey achieved efficiently. Cost efficiency is achieved as multiple users of same date is just referred and not newly added. Deleting content of shared file of different user will allow deleting only convergent keys references not content stored in server.

## C. DeKey

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 over head 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.

## D. System Setup

The system setup phase in Dekey is similar to that in the baseline approach, but involves an additional step for initializing the key storage in KM-CSPs. In Dekey, we assume that the number of KM-CSPs is  $n$ .



# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirce.com](http://www.ijirce.com)

Vol. 5, Issue 6, June 2017

S1: On input security parameter  $1\lambda$ , the user initializes a convergent encryption scheme, and two PoW protocols PoWF and PoWB for the file ownership proof and block ownership proof.

S2: The S-CSP initializes both the rapid storage system and the file storage system and set them to be  $\perp$ .

S3: Each KM-CSP initializes a rapid storage system for block tags and a lightweight storage system for holding convergent key shares, and sets them to be  $\perp$ .

## E. File Upload

To upload file  $F$ , the user and the S-CSP perform both file level and block-level de-duplications. The file-level deduplication operation is identical to that in the baseline approach. After both file-level and block-level duplicate checks, an additional stage called key distribution is performed. File upload includes following steps-

- 1) User computes and sends the file tag to server.
- 2) Server checks whether their exists the same tag. If so server replies the user with response “file duplicate” or “no file duplicate” otherwise.
- 3) If response is “file duplicate”, then user runs  $PoW_F$  on  $F$  with  $j$ -th KM-CSP. If response is “no file duplicate”, then user proceeds with block level duplication.
- 4) Divide the input file  $F$  into set of block  $B_i$  (where  $i= 1, 2 \dots$ ). Compute block tags and send to each KM-CSP for duplicate checks. Server checks whether the same tag exists for each block. If so server replies the user with response “block duplicate” or “no block duplicate” otherwise.
- 5) If response is “block duplicate”, then user runs  $PoW_B$  on  $B_i$  with  $j$ -th KM-CSP. If proof is pass, the  $j$ -th KM-CSP will return a pointer for the secret share stored for the convergent key  $K_i$  to the user; otherwise  $j$ -th KM-CSP keeps  $T(B_i)$  and sends back a signal to ask for the secret share on the convergent key.
- 6) Upon receiving results for a block  $B_i$  returned from KM-CSPs, if it is a valid pointer, the user stores it locally; otherwise the user computes the secret shares  $K_{i1}, K_{i2}, \dots, K_{in}$  by running  $Share(K_i)$  using the  $(n, k, r)$ -RSSS. It then sends the share  $K_{ij}$  and  $T_j(B_i)$  to the  $j$ -th KM-CSP for  $j = 1, 2, \dots, n$  via a secure channel.
- 7) Upon receiving  $K_{ij}$  and  $T_j(B_i)$ , the  $j$ -th KM-CSP stores them and sends back the pointer for  $K_{ij}$  to the user for future access.

## F. File Download

To download file  $F$ , the user first downloads the encrypted blocks  $C_i$  from the S-CSP as described in the baseline scheme. It needs to decrypt these encrypted blocks by recovering the convergent keys. Specifically, the user sends all the pointers for  $F$  to  $k$  out of  $n$  KM-CSPs and fetches the corresponding shares  $K_{ij}$  for each block  $B_i$ . After gathering all the shares, the user continues to reconstruct the convergent key  $K_i = Recover(K_{ij})$  for  $B_i$ . Finally, the encrypted blocks  $C_i$  can be decrypted with  $K_i$  to obtain the original file  $F$ .

## G. Ramp Secret Sharing

Dekey uses the Ramp secret sharing scheme (RSSS) to store convergent keys. The  $n, k, r$ -RSSS (where  $n > k > r$ ) 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, 2) no information about the secret can be deduced from any  $r$  shares.

## VI. SIMULATION RESULTS

All experiments were performed on an Intel Core i5, 4th Generation (2.40 GHz) machine with Windows 10 OS. The default data block size is 4 KB. For better results, a larger data block size can be considered (e.g., 8 KB instead of 4 KB). Hash function SHA-256 is used to generate hash key of size 32 bytes. For encryption purpose, AES-128 algorithm is used in Electronic Codebook (ECB) mode. Open source version of Secret sharing algorithm is used to generate shares of convergent. Open source library is used to implement Ramp Secret Sharing.

Table 1 shows the average time in milliseconds for each operation file upload and file download for respective block size in baseline approach. Time required for file upload is more than that of file download because file upload includes block division, convergent key computation, block encryption and convergent keys are also encrypted with master key.



# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: [www.ijirce.com](http://www.ijirce.com)

Vol. 5, Issue 6, June 2017

Data Size (KB)	Avg. Time For Upload (Milliseconds)	Avg. Time For Upload (Milliseconds)
50	285	38
100	301	69
500	638	169

Table 1: Result Analysis for Baseline Approach

Table 2 shows the average time in milliseconds for each operation file upload and file download for respective block size in DeKey approach. Time required for both operations in DeKey approach is more than that of baseline approach because additional time is needed to distribute convergent keys to KM-CSPs.

File Size(Kb)	Avg. Time For Upload (Milliseconds)	Avg. Time For Upload (Milliseconds)
50	4161	501
100	8060	1103
500	19506	3207

Table 2: Result Analysis for DeKey Approach

## VII. CONCLUSION

The propose DeKey, an efficient and reliable convergent key management scheme for secure deduplication. DeKey applies deduplication among convergent keys and distributes convergent key shares across multiple key servers, while preserving semantic security of convergent keys and confidentiality of outsourced data. DeKey using the Proof of ownership scheme incurs small encoding/decoding overhead in the regular upload/download operations.

## REFERENCES

1. J. Li, X. Chen, M. Li, J. Li, P. Lee and W. Lou, 'Secure deduplication with efficient and reliable convergent key management', Proc. IEEE Trans. Parallel Distrib.Syst., Vol. 25, No. 6, Issue 1, pp. 1615-1625, 2014.
2. G. Wallace, F. Dougliis, H. Qian, P. Shilane, S. Smaldone, M. Chamness and W. Hsu, 'Characteristics of Backup Workloads in Production Systems', Proc. 10th USENIX Conf. FAST, pp. 1-16, 2012.
3. Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, 'Enabling Public Audit ability and Data Dynamics for Storage Security in Cloud Computing', IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 847-859, 2011.
4. W. Wang, Z. Li, R. Owens, and B. Bhargava, 'Secure and Efficient Access to Outsourced Data', Proc. ACM CCSW, pp. 55-66, 2009.

## BIOGRAPHY

**Nilam Patil** is a Post Graduate (M.E.) student in the Computer Engineering Department, PVPIT, and Pune University.

**B. K. Sarkar** is a Professor and HOD, Computer Engineering Department, PVPIT, Pune University.