INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume3, Issue1)
Available online at: www.ijariit.com

Secure Convergent key and Deduplication using Distributed Convergent key Management

Nilam Patil
Dept. of Computer Engineering PVPIT, Bavdhan, Pune, India.
patil.nilam125@gmail.com

Prof. (Dr) B. K. Sarkar
Dept. of Computer Engineering PVPIT, Bavdhan, Pune, India.
dr.bksarkar2003@yahoo.in

***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 servers.

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 point- ers 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 behavior makes block deduplication far more efficient. However, block deduplication takes more processing power and uses a much larger index to track the individual pieces.

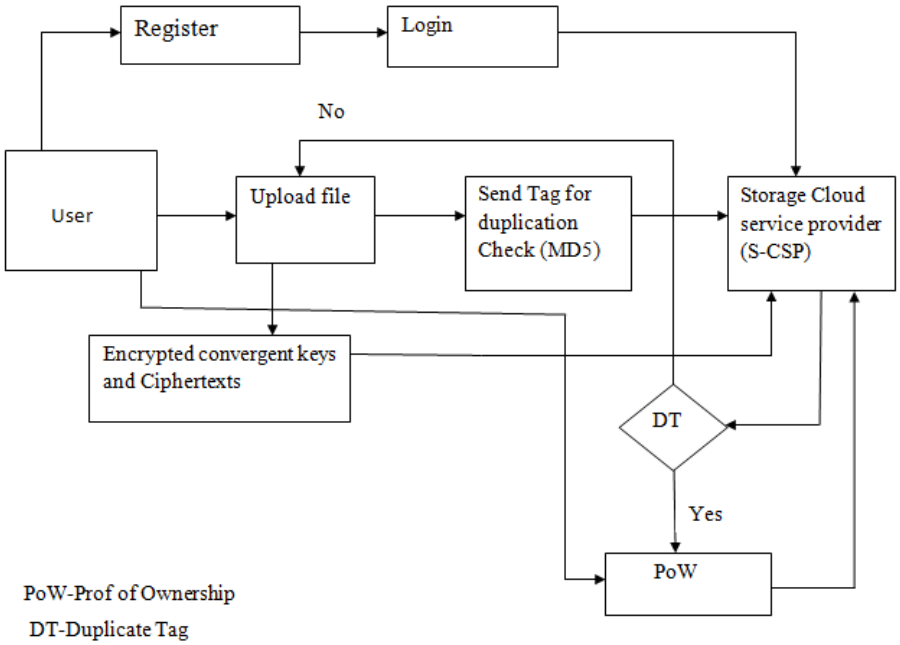
From a user's perspective, data outsourcing raises security and privacy concerns. We must trust third-party cloud providers to properly enforce confidentiality, integrity check- ing, and access control mechanisms against any insider and outsider attacks. Traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different ciphertexts, mak- ing deduplication impossible [1]. Convergent encryption [2] has been proposed to enforce data condentiality 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 orig- inal data copy is first encrypted with a convergent key and convergent key is encrypted by a master key that will be kept 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 is a need to keep by each user.

II. 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



(a)

Fig. 1: Architecture of Upload files for Baseline Approach

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.

1) 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 $PoWF$ and $PoWB$ *to* denote POW for a file F and block B respectively. Steps for check the Ownership of file:

- User send φ(f) to server side.
- Server derives φ' (f).
- Proof is correct if φ(f)=φ' (f).

B. 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.

1) System setup: The system setup phase initializes the necessary parameters in the following two steps:

The following entities are initialized: 1) a symmetric encryption scheme with the primitive functions (KeyGenSE , EncryptSE , DecryptSE) and the user's master key k = KeyGenSE(1^{λ})for some security parameter 1; 2)a convergent encryption scheme with the primitive functions ($KeyGen_{CE}$, $Encrypt_{CE}$,$Decrypt_{CE}$, $TagGen_{CE}$);and 3)a PoW algorithm $PoWF$ for the file and a PoW algorithm for the block , which is denoted by $PoWB$.

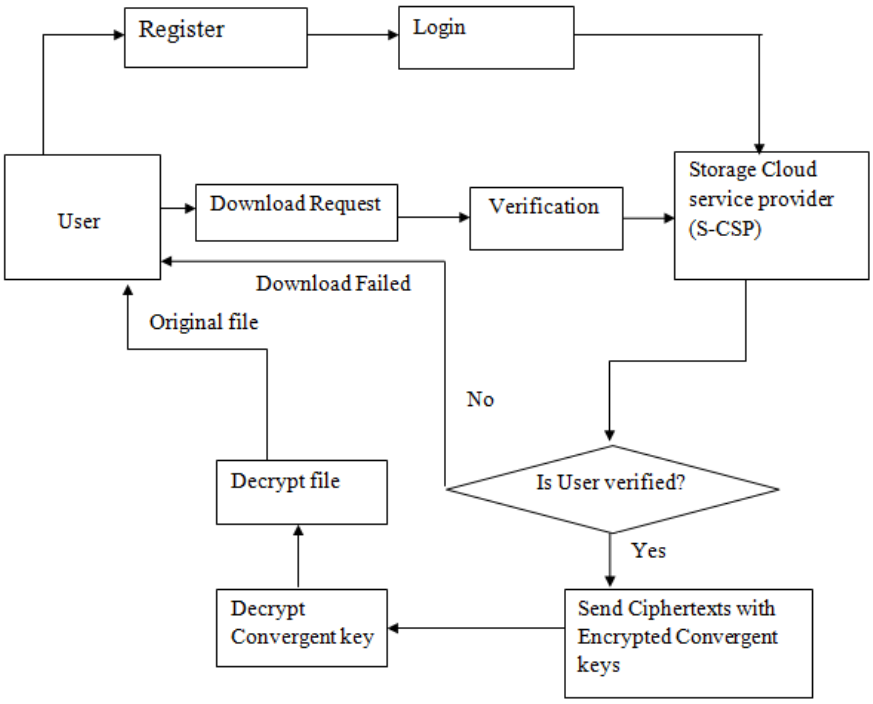
The S-CSP initializes two types of storage systems: a rapid storage system for storing the tags for efficient duplicate checks, and a file storage system for storing both encrypted data copies and encrypted convergent keys. Both storages systems are initialized to be $\perp$.

2) File upload: Suppose that a user uploads a file F. First, it performs file level and block level deduplication as see Fig.1

a) : User computes and sends the file tag to server.

b) : Server checks whether their exists the same tag. If so server replies the user with response "file duplicate" or "no file duplicate" otherwise.

c) : If response is "file duplicate", then user runs $PoWF$ on F with server. If response is "no file duplicate", then users it proceeds with block level duplication.



(a)

Fig. 2: Architecture of download files for Baseline Approach

d) : Divide the input file F into set of block B_i (where i= 1, 2, . . .). Compute block tag and send to server for duplicate checks.

Server checks whether their exists the same tag for each block. If so server replies the user with response "block duplicate" or "no block duplicate" otherwise.

e) : If response is "block duplicate", then user runs $PoWB$ *on* Bi with server. Otherwise it encrypts the block with convergent key of that block.

f) : Encrypt all convergent keys with master key

g) : User uploads unique blocks, encrypted convergent keys and file tag to the server.

3) 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.

a) : User sends request for file F along with file name.

b) : Sever 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.

c) : User uses master keys to recover each convergent key. Then it uses each convergent key to recover original block.

C. Limitations

The baseline approach suffers two major problems. The first problem is the enormous storage overhead in key management.Each user has encrypted convergent keys with his own master key for each data copy.The encrypted convergent keys (i.e,$CK^r s$) 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.

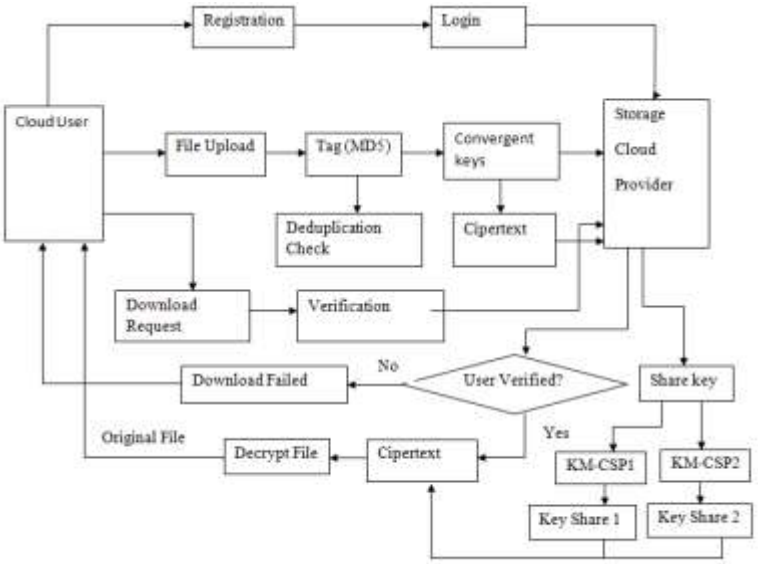
III. PROPOSED SYSTEM

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 effi- cient 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
- We implement Dekey using the Ramp secret sharing scheme that enables the key management to adapt to different reliability and confidentiality levels.



(a)

Fig. 3: Architecture of Dekey

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. 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. Scal- ability 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.

1) 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 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.Now the details of Dekey as follows.

2) 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.

a) : On input security parameter 1λ, the user initializes
a convergent encryption scheme, and two PoW protocols POWF and POWB for the file ownership proof and block ownership

proof, respectively.

b) : The S-CSP initializes both the rapid storage system and the file storage system and set them to be ⊥.

c) : 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 ⊥.

3) File upload:

a) : User computes and sends the file tag to KM-CSP.

b) : Server checks whether their exists the same tag. If so server replies the user with response "file duplicate" or "no file duplicate" otherwise.

c) : If response is "file duplicate", then user runs PoWF on F with jth KM-CSP. If response is "no file duplicate", then user it proceed with block level duplication.

d) : Divide the input file F into set of block Bi (where i= 1, 2, . . .). Compute block tag and send to each KM-CSP for duplicate checks.

Server checks whether their exists the same tag for each block. If so server replies the user with response "block duplicate" or "no block duplicate" otherwise. If response is "block duplicate", then user runs PoW_B on Bi 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.

e) : Upon receiving results for a block Bi 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, ..., K_ik 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, . . . , n via a secure channel.

f) : 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.

4) File download: To download file F, the user first down- loads 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.

5) Ramp Secret Sharing:

a) : 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

b) : After implementation of DeKey algorithm, we will check the feasibility of implementation of Ramp Secret Shar- ing Algorithm on data. This will help to reduce single point of failure of Storage Cloud Service Provider. Blocks are divided into n shares and distributed across n Storage cloud servers. To recover the data, at least k ($n > k > r > 0$)shares are required and those can be obtained from k servers. Here different aspects like execution time, performance will be checked and compared with previous approach.

CONCLUSION

The propose DeKey, an efficient and reliable convergent key management scheme for secure deduplication. DeKey applies deduplication among convergent keys and distributes conver- gent 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 up- load/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"* in Proc. IEEE Trans. Parallel Distrib. Syst.,http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.284, 2014.

[2] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Cham- ness, and W. Hsu., "*Characteristics of Backup Workloads in Production Systems"* in Proc. 10th USENIX Conf. FAST, 2012, pp. 1-16..

[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, May 2011.

[4] W. Wang, Z. Li, R. Owens, and B. Bhargava, *"Secure and Efficient Access to Outsourced Data"* in Proc. CM CCSW, Nov. 2009, pp. 55-66.

