A secure deduplication scheme for encrypted data

Vishal Passricha[1], Ashish Chopra[2], Pooja Sharma[3], Shubhanshi Singhal[4]
[1,2]Assistant Professor, National Institute of Technology, India
[3]Lecturer, Government College for Women, India
[4]Assistant Professor, TERii., Kurukshetra University, India

Article Info

Article history:

Received Nov 22, 2018
Revised Mar 25, 2019
Accepted Apr 20, 2019

Keywords:

Cloud Storage
Convergent Encryption
Deduplication
Proof-of-Ownership

ABSTRACT

Cloud storage (CS) is gaining much popularity nowadays because it offers low-cost and convenient network storage services. In this big data era, the explosive growth in digital data moves the users towards CS to store their massive data. This explosive growth of data causes a lot of storage pressure on CS systems because a large volume of this data is redundant. Data deduplication is a most-effective data reduction technique that identifies and eliminates the redundant data. Dynamic nature of data makes security and ownership of data as a very important issue. Proof-of-ownership schemes are a robust way to check the ownership claimed by any owner. However to protect the privacy of data, many users encrypt it before storing in CS. This method affects the deduplication process because encryption methods have varying characteristics. Convergent encryption (CE) scheme is widely used for secure data deduplication, but it destroys the message equality. Although, DupLESS provides strong privacy by enhancing CE, but it is also found insufficient. The problem with the CE-based scheme is that the user can decrypt the cloud data while he has lost his ownership. This paper addresses the problem of ownership revocation by proposing a secure deduplication scheme for encrypted data. The proposed scheme enhances the security against unauthorized encryption and poison attack on the predicted set of data.

Corresponding Author:

Shubhanshi Singhal,
Department of Computer Science and Engineering,
Technology Education and Research Integrated Institute,
Kurukshetra, Haryana, India.
Email: Shubhanshi17@gmail.com

1. INTRODUCTION

In this digital era, the data is growing at an enormous rate hence the word 'Big Data' is introduced. Social networks and mobile computing systems are major players in generating Big Data. The 16.1 ZB of data was produced in 2016, and it is estimated that nearly 163.2 ZB of data will be produced in 2025 [1]. To store this huge volume of data, cloud services provide the cloud storage (CS) that are set-up at different geographical locations. CS is highly prevalent nowadays because it offers low-cost and location independent virtual storage and processing resources [2]. The volume of data is rising exponentially. Hence, cloud service providers (CSP) are required to adopt some techniques for managing disk space and enhancing reliability. Data deduplication is a technique that identifies and eliminates the redundant data and stores an only single instance of it. It improves both savings of storage space and network bandwidth [3-5]. Dropbox [6], Google Drive [7], and IDrive [8] have also adopted deduplication techniques to reduce resource consumption.

CS security is also a challenging issue. CS adopts various data integrity checking schemes for security, but client-side deduplication always arises security issues. Malicious users may download any file by cheating the cloud server (CSS). Proof-of-ownership is the highly adaptable approach for secure client-side deduplication. By this, the owner can effectively prove to cloud server his ownership that he indeed holds the whole file. The customer usually encrypts his private data with his own encryption key and uploads the encrypted data to CS to preserve his privacy. Randomization in encryption, i.e., different keys generate different ciphertext for the same data leads to infeasibility in deduplication.

Convergent encryption (CE) is a good solution to this problem. In CE, the encryption key is derived from the data itself for encryption [9]. Therefore, every user having the same data will get the same ciphertext. By this, deduplication of encrypted data is made possible. The major security flaw in CE is that key derivation process is deterministic, i.e., same files always derive the same encryption key. If attackers get the key, then CE becomes worthless [9]. The problem of CE is solved by DupLESS [10] by providing a robust security scheme. However, in early proposed CE schemes including DupLESS, a user may decrypt the data after losing his ownership of uploaded data, i.e., ownership revocation.

In this paper, a secure deduplication scheme is proposed to store encrypted data. It ensures the authorized access to the shared data which is examined as a major challenge for secure and efficient CS services [11]. The ownership changes dynamically hence each ownership group is handled by a group key management mechanism. The proposed scheme guarantees no leaking of any confidential information. The paper is organized as follows: related work is given in section 2. Section 3 describes the system model and security requirements. Chapter 4 explains the preliminaries and notations. The proposed secure deduplication scheme is demonstrated in section 5. The security results of the proposed scheme are described in section 6. Finally, section 7 concludes the paper.

2. RELATED WORK

Data deduplication is an advanced data compression technique which identifies the redundant data by comparing the hash values of data chunks. It stores only the unique copy and creates logical links for redundant copies instead of storing them [9, 12]. By this, it reduces the volume of data stored and as a result, the running cost of the system decreases. Generally, it is applied where the data is stored or transmitted in CS [13]. It is successfully adopted by backup storage in clouds to manage physical capacity and network traffic [14, 15]. It can be divided into two categories: deduplication over unencrypted data and deduplication over encrypted data. Data privacy is also an important issue in CS. It protects the data from malicious attackers outside and inside the CS. Harnick et al. [16] use data deduplication as a side channel to reveal information to attackers. They use a randomized threshold to perform an attack on CS services. When a malicious user temporarily embraces a server and gets the hash values of data in CS, then he can download all these data. A similar attack scenario was brought forward on CS that uses deduplication across multiple users. Hence by obtaining an only hash value for a specific data, the malicious user becomes so powerful that he can access all data that is stored in CS. Dropbox was accessed using manipulation of the hash value attack by Mullazani et al. [11]. He showed that if the malicious user retrieves the SHA-256 hash values of file's chunks then other Dropbox user's data may be accessed by spoofing the hash value attack.

The notion of proof-of-ownership is proposed by Halevi et al. [17] to overcome these attacks. In this, the file is held using Merkle trees instead of hash value. Deduplication over unencrypted data stores the plain data in CS and plain data is sent to the user after verifying his ownership hence CS should be trusted. So users are always more concern about their privacy. To keep their data privacy, users may encrypt their data before storing to CS. Different users use different mechanisms to encrypt the data. Due to this, different ciphertexts are generated for the same data which makes the deduplication infeasible. To overcome this problem, Douceur et al. [5] proposed a new approach known as convergent encryption. In CE, data owner generates its own key from the data M , $K \leftarrow H(M)$ where H be a hash function. The key K and data M are mapped into ciphertext $C \leftarrow E(K,M)$, where E is a block cipher. Then only K is maintained, M is deleted and C is uploaded to CS. If another user encrypts the same data then the same ciphertext will be generated. Hence if CS receives the C from another user then CS will not store it, only meta-data will be updated. Only authorized users can request and download ciphertext C and decrypt it with K .

However, CE suffers from the poison attack [18]. It mainly arises the tag consistency problem in which malicious user uploads maliciously generated ciphertext and its tag. When another user tries to store his ciphertext; the server finds the same tag and deletes it. When a user downloads this ciphertext and decrypts it, the message is not found original. By this, the integrity of data and CS is lost. CE does not offer required security, and it is found vulnerable to poison attack [5]. This security flaw is resolved by oblivious pseudo-random function (O-PRF) based DupLESS [10]. It replaced the data derived encryption key with the key generated by the interaction between the key server and the user. It provides security against poison attack, but the key is still deterministic.

3. SYSTEM MODEL AND SECURITY REQUIREMENT

In this section, data deduplication architecture and security requirements are described. Based on granularity, deduplication models are divided into two categories: course grained and fine grained. For the sake of simplicity, coarse-grained deduplication is preferred.

3.1. System Description

Figure 1 shows the system model of the data deduplication system for cloud storage, having the following entities:

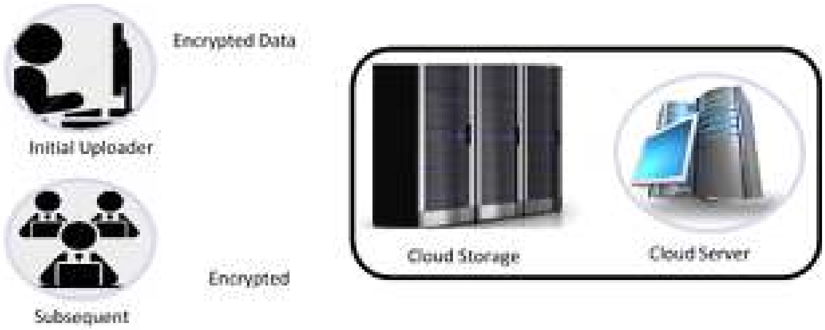

Figure 1. The model of data deduplication system

i. Data owner: The data owner uploads the data to CS. He may encrypt his data before uploading and uploads ciphertext to CS with its index information. If the data uploaded by data owner do not already reside in CS, then data owner is known as initial uploader otherwise he is called as the subsequent uploader. An ownership group is a group of data owners who share the same data in CS.

ii. Cloud Server (CSS): Cloud server and cloud storage are two different entities. CSS performs the deduplication process on uploaded data and if this data is not present in CS then CSS stores this in CS. It manages the list of owners for stored data and also controls the access of stored data using the list of ownership. It is a group key authority that maintains group key for each ownership group. It executes assigned tasks honestly but learns as much information about the encrypted contents as possible. When the data owners request to download a file, after authenticating the owner's identity, it provides the file's access.

iii. Cloud Storage (CS): CS is a logical server managed by the cloud service provider over the internet. It offers similar functionality as a traditional server. It offers distributed storage with immediate access. It is highly configurable and interchangeable.

3.2. Security Requirements

The following security requirements are tried to achieve in this paper.

Data Confidentiality: The unauthorized and malicious user should be restricted to change and know about the encrypted data. If data stored in CS are maliciously modified, then its integrity may be lost. The deduplication algorithms should be able to protect it from any poison attacks. It only allows authentic users to access data and guarantees that the data uploaded in CS are not altered.

Collusion Resistance: The data owners want to keep their privacy. However, malicious users who do not have valid ownership of data may access it from CS. Hence CSS must restrict them to decrypt or access it.

Backward and Forward Secrecy: To restrict any user from accessing plaintext of the outsourced data before uploading is known as backward secrecy. To restrict any user, who has deleted or modified the data in CS from accessing the outsourced data after deletion or modification is known as forward secrecy.

4. PRELIMINARIES

In this section, the basic definitions of a secure deduplication scheme for encrypted data are described.

4.1. Notations

The random and uniform selection of the element x from the finite set S is denoted by $x \xleftarrow{\wp} S$. It means one value from S is randomly assigned to x . $y \leftarrow Al(x_1, ..., x_n)$ is denoted as an algorithm Al which runs on the input set $(x_1,...,x_n)$ and its result is assigned to y. 1^α means a string of length α having all 1s. The concatenation of two strings a and b is expressed as $a \, \text{P} b$.

$(u_1,...,u_n)$ is a world of user denoted by U. The identity of the i^th^ user u_i is defined as ID_i. Let $G_a \subset U$ is a group of users having data M_a and $L_a = \{T_a, G_a\}$ is the ownership list of M_a. Here T_a denotes the tag and CSS maintains the ownership list. K_{G_a} is an ownership group key, generally shared by valid owners of G_a.

4.2. Definitions

A secure deduplication scheme for encrypted data is defined in this section. The algorithms used in this scheme are described below:

a. $KEY \xleftarrow{\wp} KEYGen(u)$: The set of user u is given as input to the $KEYGen(\ )$ algorithm. It generates *KEYs* corresponding to each user set of u for secure ownership group key distribution.

b. $Ci \xleftarrow{\wp} Enc(M, 1^\alpha)$: The data M and security parameter α is given as input to $Enc(\ )$ function to encrypt it into the corresponding ciphertext Ci. Ci is composed of encrypted data and tag information indexing.

c. $Ci' \xleftarrow{\wp} \text{Re}-Enc(Ci, G)$: The ciphertext Ci and ownership group G is given as input to $\text{Re}-Enc(\ )$ for mapping the ciphertext Ci into corresponding encrypted ciphertext Ci'. The characteristic of encrypted ciphertext Ci' is that valid owners in G only do its decryption.

d. $M \longleftarrow Dec(Ci', K, Pk)$: The encrypted ciphertext Ci', message encryption key K, and a set of keys Pk is supplied as input to $Dec(\ )$ function to decrypt Ci' into the corresponding message M. Note that K should be derived from data M.

5. PROPOSED DEDUPLICATION SCHEME

In this paper, a secure deduplication scheme for encrypted data is proposed that offers dynamic ownership management capability. Randomized CE algorithm [19] is applied in the proposed scheme to randomize the encrypted data. It powered the proposed system to defend against poison attacks. Re-encryption module is also combined in the proposed scheme for owner revocation. The outsourced ciphertext is re-encrypted using this module for owner revocation. It also distributes the re-encryption key to authentic owners using CSS. Figure 2 illustrates the architecture of the proposed scheme. The ownership list for each data is maintained by the CSS to handle dynamic ownership management. As CSS gets ownership list, it does not disobey the security requirement. It only allows the re-encrypted ciphertext, and rest information regarding data encryption is meaningless.

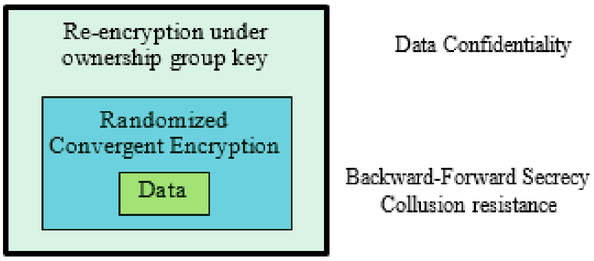
Re-encryption under ownership group key
Randomized Convergent Encryption
Data
Data Confidentiality
Backward-Forward Secrecy
Collusion resistance

Figure 2. Scheme overview and corresponding security

5.1. System Construction

Let $E_K(M)$ denotes symmetric encryption [20] with the key K of data M where $E_K(\ )$ is a block cipher. A cryptographic hash function $H: \{0,1\}^k$ is used to derive an encryption key and tag from the message and k denotes the length of the key K.

Key Generation: The *KEYs* for the users in u are generated by CSS by running $KEYGen(u)$. Initially, a binary *KEY* tree is designed for the world of users u by CSS. Figure 3 shows the binary *KEY* tree and users u are getting ownership group keys through this tree. Each user (node) v_j has a *KEY*, represented as KEY_j. Path keys are the set of keys from leaf to root, and it is separately maintained by each user (node). Figure 3 shows how u_3 keeps KEY_6, KEY_3, and KEY_1 as its path keys Pk_3. For $u_j \in U$, the path keys of u_j is denoted by Pk_j. CSS constructs the *KEY* tree in the following manners:

1. Each user of U gets a leaf node. Randomly generated keys are assigned to each node of the tree.
2. Each user $u_j \in U$ gets securely the path key Pk_j. In the re-encryption phase, CSS uses the path keys as *KEYs* to encrypt the ownership group keys.

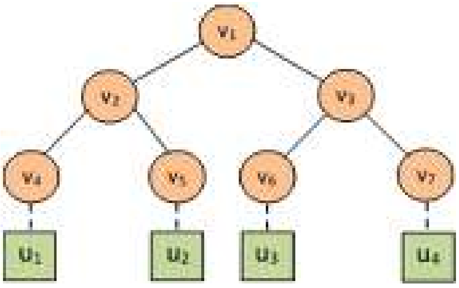

Figure 3. KEY tree for ownership group key distribution

Data Encryption: If a data owner u_j wants to upload data M to CS then he runs $Enc(M, 1^{\alpha})$ module to encrypt M. $L \leftarrow \{0,1\}^{k(\alpha)}$ denotes the random encryption key and algorithm $k(\alpha)$ drives the length of the encryption key. $K_a \leftarrow H(M_a)$ is computed from M_a. It is used to encrypt message encryption key L. $T_a \leftarrow H(K_a)$ is computed from K_a as indexing information of data. $Ci_a^1 \leftarrow E_L(M_a)$ and $Ci_a^2 \leftarrow L \oplus K_a$ encrypt the data and encryption key respectively. The ciphertext Ci_a is constructed by concatenation of Ci_a^1 and Ci_a^2, i.e., $Ci_a = Ci_a^1 \parallel Ci_a^2$.

After generation of Ci_a^1, *uploads* $\parallel T_a \parallel Ci_a \parallel ID_a$ are sent to CS. After that M_a is deleted to save storage space and K_a is kept. If u_a is the first uploader then cloud server embeds ID_a into G_a and generates $L_a = \{T_a, G_a\}$, and store Ci_a in CS otherwise it embeds ID_a to G_a and discards Ci_a.

Data Re-Encryption: Cloud server runs $\text{Re}-Enc(Ci_a, G_a)$ to re-encrypt the ciphertext Ci_a using the ownership group information before sharing. The main objective of the re-encryption algorithm is to control the dynamically controlled owners of outsourced data. The re-encryption is done in the following steps:

1. $Ci_a^{1'} = E_{Gk_a}(Ci_a^1)$ is generated by re-encrypting the ciphertext Ci_a^1 with the random ownership group Gk_a.
2. Select the root nodes in such a way that it should cover all leaf nodes associated with users in G_a. $KEY(G_a)$ represents a set of *KEYs* of root nodes of subtrees of G_a. In Figure 3, if $G_a = \{u_1, u_3, u_4\}$ then $KEY(G_a) = \{KEY_3, KEY_4\}$. It is because root nodes (v_3, v_4) cover all the members of G_a. All the users in G_a are covered by this set. No user except $u \in G_a$ cannot know about any *KEY* in $KEY(G_a)$.
3. Derive $C_a^3 = \{E_k(Gk_i)\}_{k \in KEY(G_a)}$ ownership group key are encrypted by this algorithm before distributing to valid owners. When a user u_j requests the data then query for $T_j \parallel ID_j$ is received by the CSS. CSS validates L_j and then $T_j \parallel Ci_j^{'}$ is sent to the user otherwise request declined.

Data Decryption: If $u_a \in G_a$, then the module $Dec(C{i}_a^{'}, K_a, Pk_a)$ is run to decrypt the ciphertext C_{i_a} which is received by the user u_a from the CSS. Here ownership group key decryption is also performed by parsing C_{i_a} into $T_a, C{i}_a^{1}, C{i}_a^{2}, C{i}_a^{3}$. The ownership group key is extracted by $Gk_a = D_{KEY\in(KEY(G_a)\mathrm{I}\ Pk_a)}(C{i}_a^{3})$. Only the valid owner can decrypt the ownership group key Gk_a by formula $KEY \in KEY(G_a) \cap Pk_a$.

If the user u_a is not a member of G_a then, he cannot decrypt Gk_a . When Gk_a is retrieved, the ciphertext is decrypted into a message using formula.

$$C{i}_a^{'} \leftarrow D_{Gk_a}(C{i}_a^{1}), \quad L^{i} \leftarrow C{i}_a^{2} \oplus K_i, \qquad M_a \leftarrow D_L(C{i}_a^{1}), \quad T_i^{1} \leftarrow H(M_a)$$

Any modification made by poison attack is detected by validating tag. If $T_a^1 = T_a$, then it means tag inconsistency is present so drop the message otherwise M_a is the original data outsourced by him, and it is accepted. It is to be noted that if a valid owner requests the cloud server to delete and modify his data in CS, then his ownership must be canceled out from the ownership list and restrict him to access the data. By this step, forward secrecy is maintained. When the subsequent user uploads the same data. It is re-encrypted to offer backward secrecy. By this, subsequent users are prevented from accessing previously encrypted data. The working procedure of the proposed secure scheme shown in Figure 4.

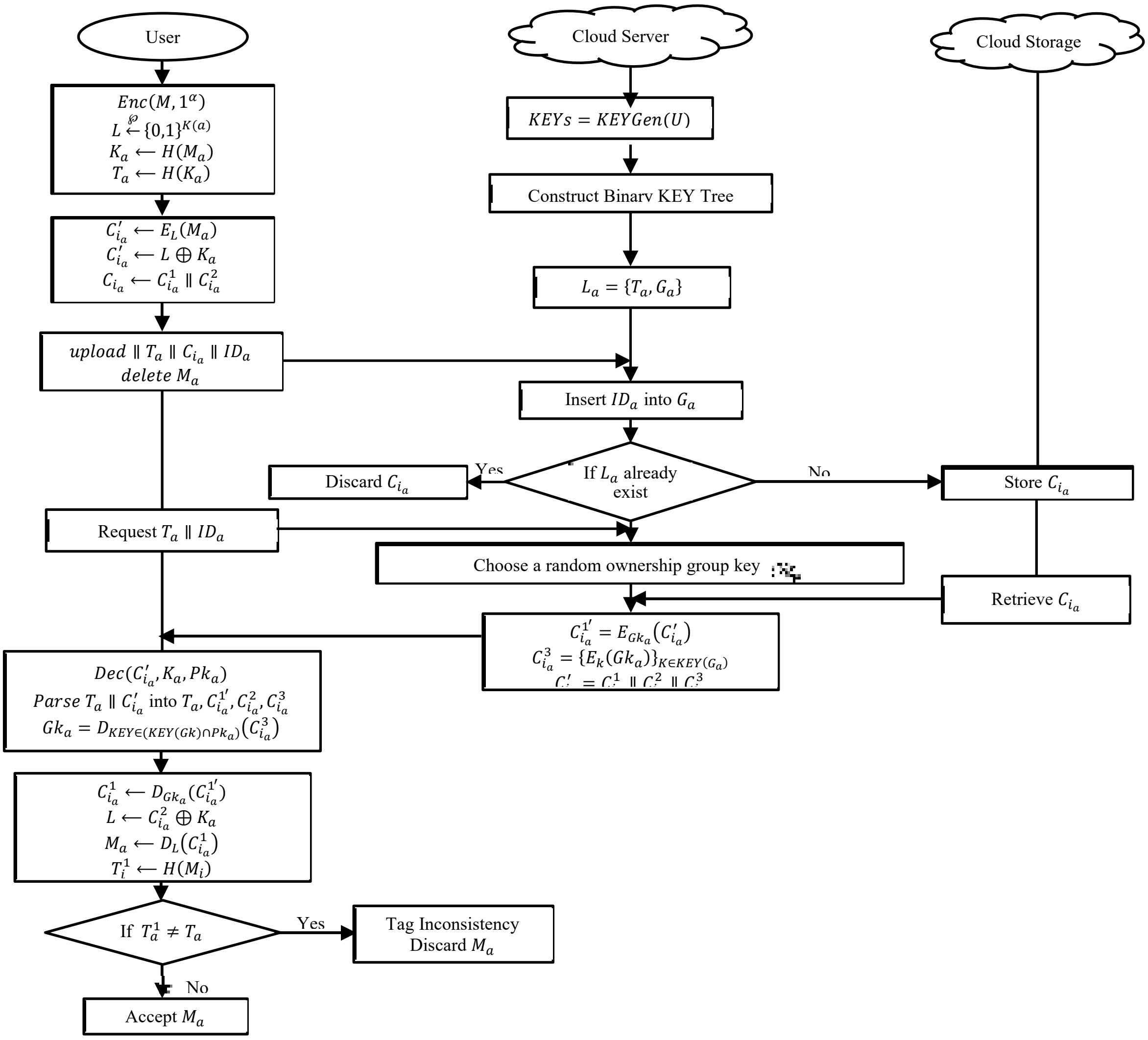
User
Cloud Server
Cloud Storage
$Enc(M, 1^{\alpha})$
$L \overset{\wp}{\leftarrow} \{0,1\}^{K(a)}$
$K_a \longleftarrow H(M_a)$
$T_a \longleftarrow H(K_a)$
$C'_{i_a} \longleftarrow E_L(M_a)$
$C'_{i_a} \longleftarrow L \oplus K_a$
$C_{i_a} \longleftarrow C^1_{i_a} \parallel C^2_{i_a}$
$upload \parallel T_a \parallel C_{i_a} \parallel ID_a$
$delete\ M_a$
$KEYs = KEYGen(U)$
Construct Binarv KEY Tree
$L_a = \{T_a, G_a\}$
Insert ID_a into G_a
Discard C_{i_a}
Yes
If L_a already exist
No
Store C_{i_a}
Request $T_a \parallel ID_a$
Choose a random ownership group key
Retrieve C_{i_a}
$C^{1'}_{i_a} = E_{Gk_a}(C'_{i_a})$
$C^3_{i_a} = \{E_k(Gk_a)\}_{K \in KEY(G_a)}$
$C'_{i_a} = C^1_{i_a} \parallel C^2_{i_a} \parallel C^3_{i_a}$
$Dec(C'_{i_a}, K_a, Pk_a)$
$Parse\ T_a \parallel C'_{i_a}$ into $T_a, C^{1'}_{i_a}, C^2_{i_a}, C^3_{i_a}$
$Gk_a = D_{KEY\in(KEY(Gk)\cap Pk_a)}(C^3_{i_a})$
$C^1_{i_a} \longleftarrow D_{Gk_a}(C^{1'}_{i_a})$
$L \longleftarrow C^2_{i_a} \oplus K_a$
$M_a \longleftarrow D_L(C^1_{i_a})$
$T^1_i \longleftarrow H(M_i)$
If $T^1_a \neq T_a$
Yes
Tag Inconsistency Discard M_a
No
Accept M_a

Figure 4. The working procedure of the proposed secure scheme

6. SCHEME ANALYSIS

This section presents the comparisons of the proposed scheme with earlier deduplication schemes. Table 1 compares the proposed scheme with CE [5], weak-leakage resilient deduplication [18], randomized CE [19] over encrypted data. These all scheme guarantee the data privacy in CSS by allowing the data owner to encrypt their data. But these all schemes do not manage dynamic ownership management.

Table 1. The Comparison of the Proposed Scheme in Terms Of Encrypted Deduplication, Tag Consistency, and Dynamic Ownership Management

Scheme	Encrypted Deduplication	Tag Consistency	Dynamic Ownership Management
CE [5]	YES	NO	NO
WLR [18]	YES	YES	NO
RCE [19]	YES	YES	NO
Proposed	YES	YES	YES

The notations used are as follows:

S_c: Size of accepted data

S_k: Size of a key

S_T: Size of a tag

S_{ID}: Size of an identity of a user

S_r: Size of a node value in Merkle hash tree

S_{PoW}: Size of exchange message for PoS

The 'key size' and 'tag size' are used to calculate the storage overhead. They are the size of keys and tag information which are stored by each owner respectively. Table 2 shows the comparisons of all these techniques in terms of storage overhead

Table 2. The Comparison of the Proposed Scheme in Terms of Storage Overhead

Scheme	Storage Overhead	
	Key size	Tag Size
CE [5]	S_k	S_t
WLR [18]	$2S_k+S_M*S_t$	S_t
RCE [19]	S_k	S_t
Proposed Scheme	$(\log n + 1)$	S_t

Table 3 shows the comparisons of all these techniques in terms of communication overhead. The communication cost of the proposed scheme is evaluated by network simulation.

Table 3. The Comparison of the Proposed Scheme in Terms of Communication Overhead

Scheme	Communication Overhead	
	Upload Message Size	Download Message Size
CE [5]	$S_c + S_r + S_{ID}$	S_c
WLR [18]	$S_c + 3S_k + S_r + S_{ID}$	$S_c + S_{pow} + S_k + S_T$
RCE [19]	$S_c + S_k + S_r + S_{ID}$	$S_c + S_k + S_T$
Proposed Scheme	$S_c + S_k + S_t + S_{ID}$	$S_c + S_k + S_T$

Figure 5 shows the encryption and decryption time for proposed scheme w.r.t. data size. Figure 6 shows the encryption time of the proposed scheme for large size data chunks for different attributes and Figure 7 shows the decryption time of the proposed scheme for large size data chunks for different attributes.

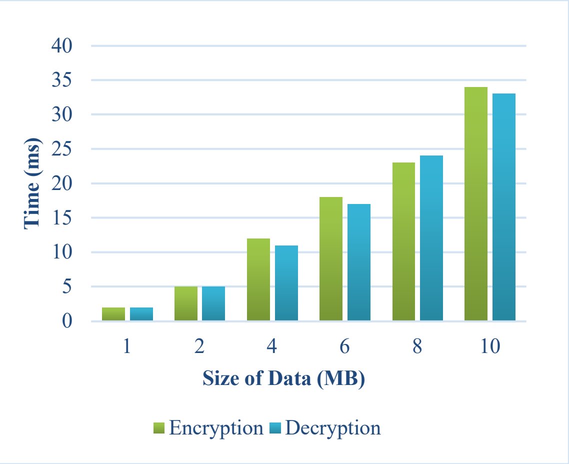

Figure 5. Encryption and decryption time w.r.t. size of data for the proposed scheme

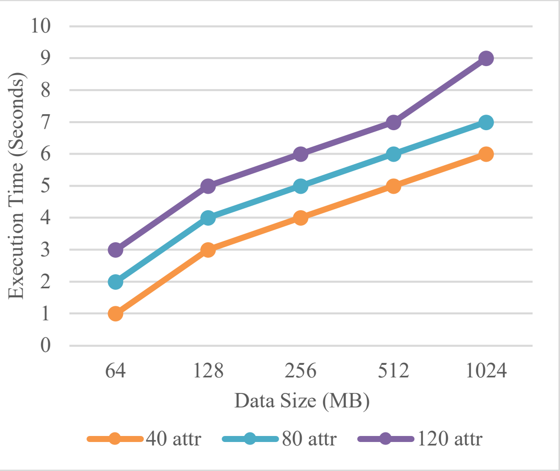

Figure 6. Encryption process (data size vs time)

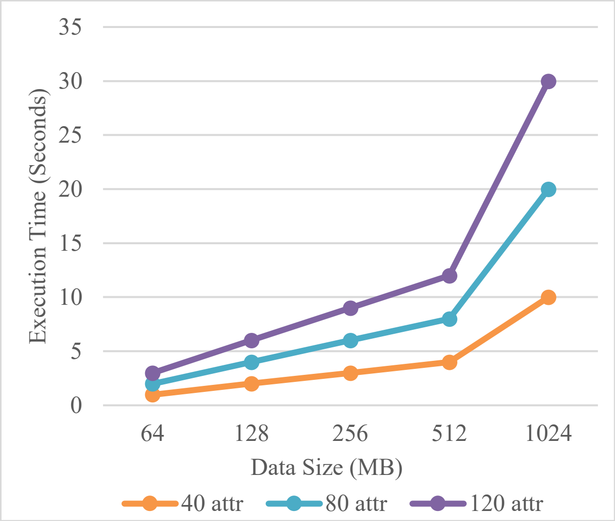

Figure 7. Decryption time (data size vs time)

6.1. Security

The proposed scheme is proved more secure in terms of data confidentiality, collusion resistance, and backward & forward secrecy.

Data Confidentiality: In the proposed scheme, the CSS is not fully trusted even if honest. So there is a need to keep plain data secret from the CSS and unauthorized users. Without loss of generality, a malicious user may request for data M_i with a tag T_i and a valid user's identity $ID_t \in G_i$. He gets the ciphertext $T_i \, \mathrm{P} C_i^{1'} \, \mathrm{P} C_i^{2} \, \mathrm{P} C_i^{3}$. If a malicious user is not the owner of data M_i then it is impossible for him to retrieve K_i and the data encryption key L because of the cryptographic hash function. Therefore, the proposed scheme guarantees data confidentiality against unauthorized users.

Collusion Resistance: The unauthorized users without valid ownership should be restricted to decrypt data even if they collude. In this model, the data encryption key L and ownership group key G_K are necessary to decrypt ciphertext into plain data. Luckily, a malicious user may get the encrypted key L but G_K , the ownership group is not possible to get. Hence, the proposed scheme offers proper security against collusion.

Backward and Forward Security: If a user tries to upload already unloaded data in CS, the ownership group key is updated randomly for G_K to $G_{K'}$. $G_{K'}$ is sent confidentially and rapidly to the data's owner. $C^{1'} = E_{Gk}(C^{1})$ is encrypted with G_K and in parallel C' is re-encrypted with the updated ownership group key $G_{K'}$ by CSS. Hence, the user cannot decrypt the previous ciphertext. By this, the proposed algorithm offers backward and forward secrecy.

7. CONCLUSION

Deduplication is an efficient technique that is successfully employed in various large scale and cloud storage. Recently, the conflicts that arise when deduplication is applied on encrypted data has been resolved. However, the ownership of outsourced data is still an issue. Dynamic ownership management is also a very challenging issue. In this paper, a secure deduplication scheme for encrypted data has been proposed. The re-encryption techniques provide power to manage any ownership changes in CSS. As the ownership changes in the ownership group of outsourced data, rapidly the data are re-encrypted using ownership group key, and this key is also updated among valid owners. By this, data confidentiality in CS is strengthened. Tag consistency is also used to take advantages of efficient data deduplication over encrypted data. The proposed scheme is more effective than earlier schemes in terms of communication cost. Therefore, the proposed scheme offers secure and efficient data deduplication in cloud storage.

REFERENCES

[1] Reinsel D, Gantz J, Rydning J. *"Data Age 2025: The Evolution of Data to Life-Critical".* An IDC White Paper. Seagate, November 2018.

[2] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. "A view of cloud computing." *Commun. ACM,* vol. 53, no 4 (April 2010), pp. 50-58.

[3] Bolosky WJ, Douceur JR, Ely D, Theimer M. "Feasibility of a serverless distributed file system deployed on an existing set of desktop." *ACM SIGMETRICS Performance Evaluation Review*. Vol. 28, no. 1 (2000), pp. 34-43.

[4] Clements AT, Ahmad I, Vilayannur M, Li J, editors. *"Decentralized Deduplication in SAN Cluster File Systems."* USENIX annual technical conference; 2009; San Diego.

[5] Douceur JR, Adya A, Bolosky WJ, Simon P, Theimer M. *"Reclaiming space from duplicate files in a serverless distributed file system."* 22nd International Conference on Distributed Computing Systems; 2002; Vienna, Austria.

[6] Dropbox. www.dropbox.com. 2014.

[7] Google Drive. https://drive.google.com. 2016.

[8] IDRIVE. https://www.idrive.com. 2017.

[9] Storer MW, Greenan K, Long DD, Miller EL, editors. *"Secure data deduplication."* 4th ACM international workshop on Storage security and survivability; 2008; Alexandria, USA.

[10] Bellare, Mihir & Keelveedhi, Sriram & Ristenpart, Thomas. (2013). *"DupLESS: Server-aided encryption for deduplicated storage."* 22nd USENIX Conference on Security. 179-194.

[11] Mulazzani M, Schrittwieser S, Leithner M, Huber M, Weippl ER, editors. *"Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space."* USENIX security symposium; 2011; San Francisco, CA.

[12] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," *2013 IEEE Conference on Communications and Network Security (CNS)*, National Harbor, MD, 2013, pp. 145-153.

[13] Shin, Youngjoo & Koo, Dongyoung & Hur, Junbeom. "A Survey of Secure Data Deduplication Schemes for Cloud Storage Systems." *ACM Computing Surveys*. vol. 49, no. 74. pp. 1-38. January 2017.

[14] V. Javaraiah, "Backup for cloud and disaster recovery for consumers and SMBs," *2011 Fifth IEEE International Conference on Advanced Telecommunication Systems and Networks (ANTS)*, Bangalore, 2011, pp. 1-3.

[15] Y. Tan, H. Jiang, D. Feng, L. Tian, Z. Yan and G. Zhou, "SAM: A Semantic-Aware Multi-tiered Source De-duplication Framework for Cloud Backup," *2010 39th International Conference on Parallel Processing*, San Diego, CA, 2010, pp. 614-623.

[16]. D. Harnik, B. Pinkas and A. Shulman-Peleg, "Side Channels in Cloud Services: Deduplication in Cloud Storage," in *IEEE Security & Privacy*, vol. 8, no. 6, pp. 40-47, Nov.-Dec. 2010.

[17] Halevi, Shai & Harnik, Danny & Pinkas, Benny & Shulman-Peleg, Alexandra. *"Proofs of ownership in remote storage systems".* Proceedings of the ACM Conference on Computer and Communications Security. 491-500. 2011.

[18] Xu, Susan Jia, Ee-Chien Chang and Jianying Zhou. "Weak leakage-resilient client-side deduplication of encrypted data in cloud storage." *AsiaCCS* (2013).

[19] Bellare M, Keelveedhi S, Ristenpart T, editors. *"Message-locked encryption and secure deduplication."* International Conference on the Theory and Applications of Cryptographic Techniques; 2013; Athens, Greece.

[20] A. Russell and Hong Wang, "How to fool an unbounded adversary with a short key," in *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 1130-1140, March 2006.

BIOGRAPHIES OF AUTHORS

Vishal Passricha received his B.Tech. with honors in Computer Engineering from Kurukshetra University, India, in 2010, He received his M.Tech. with honors in Computer Engineering from YMCAUST, Faridabad India in 2012. He is currently working as an assistant professor in the Computer Engineering Department, National Institute of Technology, Kurukshetra. He is also pursuing Ph.D. from the Computer Engineering Department, National Institute of Technology, Kurukshetra. His current research focuses on Data deduplication and Machine Learning.

Ashish Chopra is currently an Assistant Professor at the National Institute of Technology, Kurukshetra, India. He holds his Ph.D. degree and MCA degree in Department of Computer Application from the NIT, Kurukshetra in 2013 and 2006 respectively. His Ph.D. research work mainly focused on Mobile Adhoc Networks. His current research focuses on Wireless Sensor Networks and Data Deduplication.

Pooja Sharma received her Bachelor of Science in Computer Application from Kurukshetra University, Kurukshetra, India in 2009. She received her master in computer application from Kurukshetra University, Kurukshetra in 2012. She is working as a lecturer in Government college for Women, Karnal since August 2013. She has been guided many undergraduates and postgraduates project.

Shubhanshi Singhal received her B. Tech in computer science and engineering from utter pradesh technical university, Lucknow, India. She did her Master in Technology in Computer Engineering from Kurukshetra University, Kurukshetra. Presently she is working as Assistant Professor in Computer Science and Engineering Department, TERii, Kurukshetra. Her area of Interest is data deduplication and machine learning.