Advances in Science, Technology and Engineering Systems Journal Vol. 3, No. 5, 267-276 (2018)
www.astesj.com
Special Issue on Multidisciplinary Sciences and Engineering

ASTESJ
ISSN: 2415-6698

A Study on Improving Security and Efficiency using Dynamic Ownership Management in Client-Side Deduplication Environments

Won-Bin Kim, Im-Yeong Lee*

Department of Computer Science & Engineering, Soonchunhyang University, 31538, Republic of Korea

A R T I C L E I N F O
Article history:
Received: 21 August, 2018
Accepted: 24 September, 2018
Online: 12 October, 2018

Keywords:
Data Deduplication
Data Encryption
Cloud Storage

A B S T R A C T
Data deduplication technology is used to improve the spatial efficiency of cloud storage. This technology is used for storing data on a cloud and omitting data uploading if the data are already present. However, various security threats may occur during the deduplication process. These security threats include poison attacks and user identity exposure through ownership. In addition, in an environment in which ownership changes in real time, there is a problem in renewing ownership information that has already been issued. Therefore, various studies have been conducted to solve these problems. In this study, a poison attack, real-time ownership management, and ownership anonymization are provided through MLE and dynamic ownership management.

1. Introduction

The rapid development of information communication and technology (ICT) in the recent times has led to many changes in data storage environment. According to a Dell EMC report in Korea, the amount of data produced in 2013 was 4.4 trillion GB, and this value is expected to increase to 44 trillion GB by 2020. As the amount of data produced increases, there is a demand for storage media with sufficient capacity to accommodate the data. Accordingly, in the future, the basic size unit of the storage medium will exceed terabytes and will be of the order of petabytes. However, portable hard disk drives (HDDs) and universal serial bus (USB) memories, which have been used in the past, have to be carried along always, and there exists the risk of losing them.

Cloud storage is a storage service that is available remotely over a network. It provides an environment wherein multiple users can access the storage simultaneously. Therefore, it is necessary to accommodate the data of a large number of users, and it involves maintenance and expansion costs, such as the costs for periodic storage space expansion, because it is necessary to ensure availability always. However, in general, much of the data stored by users are the same. Therefore, some of the storage space of the cloud storage is wasted in storing the same data repeatedly. To solve this problem, a data deduplication technique is proposed.

Data deduplication is a technique that reduces the amount of data stored in the data storage by preventing duplication of the stored data[1]. Because most of the data stored in the data storage are stored as the same data repeatedly, storage space is wasted. Data deduplication technology allows confirming that the data to be added are stored when the data are added to the storage. At this time, if the data to be added are already stored, the data are not stored, and the ownership of the data is given to the user. Therefore, the use of data deduplication technology can prevent repeated storage of the same data and improve storage space efficiency.

The cloud storage is a remote server. Therefore, data can always be leaked because of internal or external threats. The data stored in the storage must be encrypted so that the contents of the data can't be accessed by unauthorized users. However, common encryption techniques can't be applied simultaneously with deduplication because they don't know whether the two encrypted data originated from the same source. To solve this problem, a secure data deduplication technology using various technologies such as convergent encryption (CE) has been developed.

The initial technique for secure data deduplication was developed to enable simple deduplication of encrypted data. However, during the process of secure data deduplication, various security threats such as poison attack and ownership forgery attack occurred. In addition, a number of techniques were studied to solve such threats, but these techniques created additional problems such as the inconvenience of ownership management and excessive operation. In this paper, we propose an improved method of secure data deduplication. This paper is an extension of work originally presented in 2017 4th International Conference on Computer

*Corresponding Author. - Im-Yeong Lee, Email: imylee@sch.ac.kr

Applications and Information Processing Technology (CAIPT) [2].

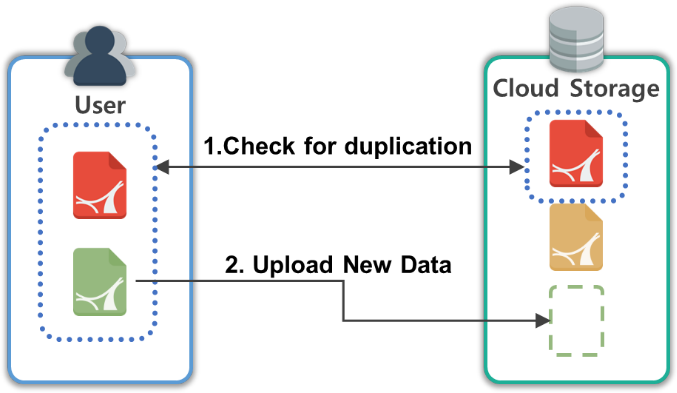

Figure 1. Data deduplication

2. Related Works

2.1. Data Deduplication

Data deduplication is a technique that prevents the same data from being repeatedly stored. To achieve this, when data are stored, it is necessary to check whether the same data are already present in the storage system or device as shown in Fig. 1. Therefore, with data deduplication a comparison of the data is conducted to check whether the same data are already stored. To do so, the data source is hashed and compared, and the data are stored according to the comparison results. Various methods can be applied during this process, and various types of systems can be designed according to the method used.

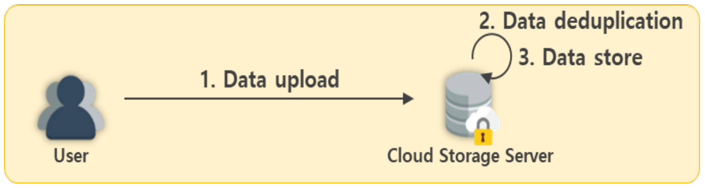

Figure 2. Server-side deduplication

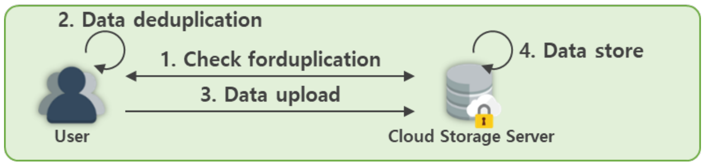

Figure 3. Client-side deduplication

2.2. Client-Side Deduplication (CSD)

Data deduplication uses a method for determining whether the data to be uploaded have already been stored. During this process, it is the responsibility of the storage server to determine whether the data already exist. However, the process of removing redundant data may take various forms. In the initial data deduplication method, all data to be uploaded are transmitted to the server, and the deduplication process is conducted in the server, as shown in Fig. 2. This is called server-side deduplication. However, because all data including redundant data are transmitted, a large amount of data transmission traffic occurs irrespective of the ratio of redundant data. In addition, bottlenecks may be incurred when data are uploaded from many different users

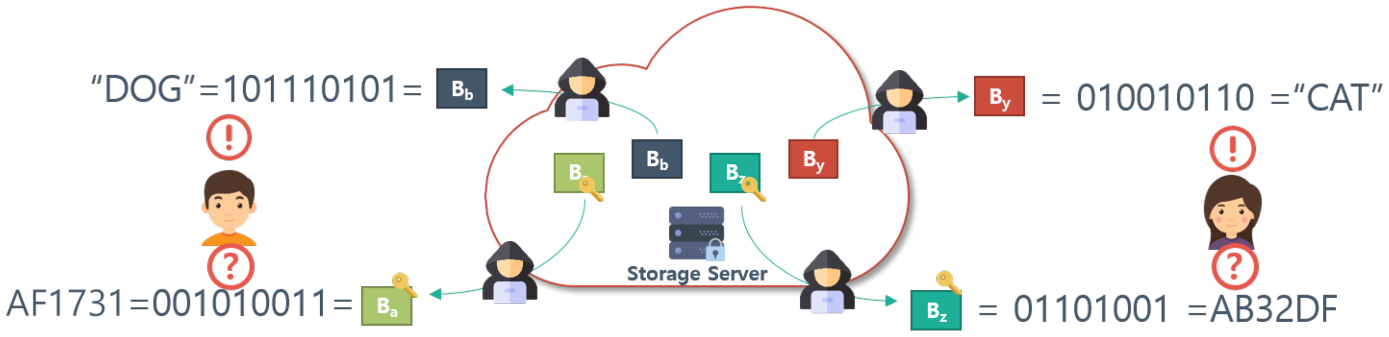

Figure 4. Necessity of data encryption

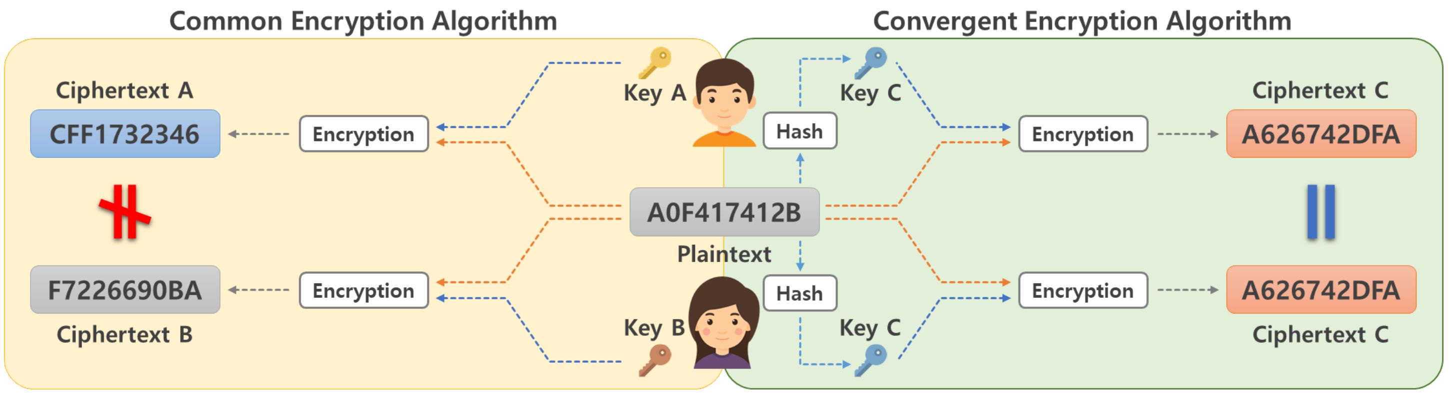

Figure 5. Comparison of common encryption and convergent encryption

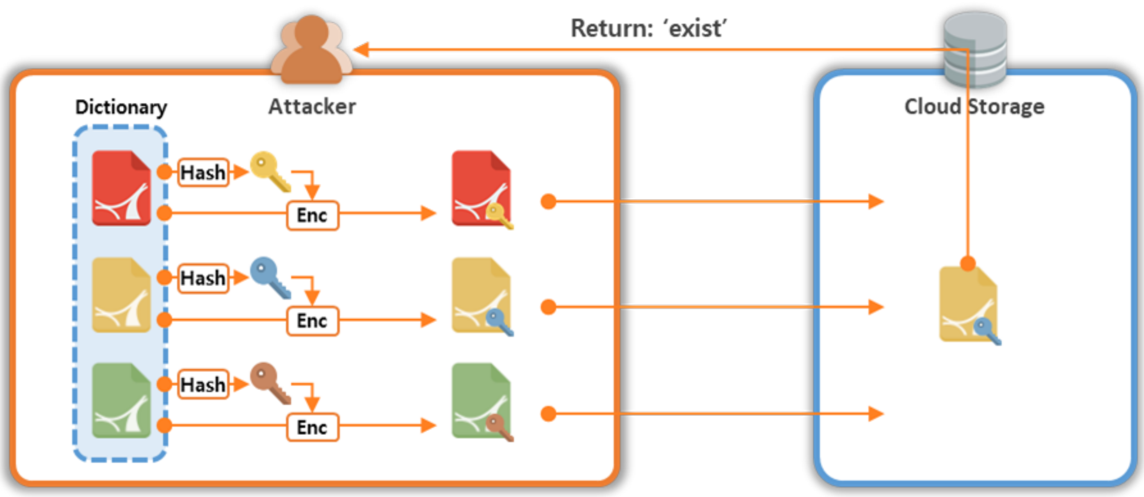

Figure 6. Dictionary attack scenario

concurrently. Therefore, client-side deduplication (CSD) technology has been proposed to solve this problem.

CSD is a method for determining redundancy by comparing a list of stored data with a list of identifiers from the data to be uploaded by the client to the server, as shown in Fig. 3. The server determines whether the data are redundant and transmits a list of non-duplicated data to the user. The user conducts deduplication using the duplicated data list received from the server, and transmits only the deduplicated data to the cloud storage server. CSD can reduce the amount of data transmission traffic and load on the server side. In this study, we used the CSD method [1-3].

2.3. Convergent Encryption (CE)

Data deduplication is conducted by comparing whether two data are the same. Therefore, we use hashed data for the data comparison. However, cloud storage is provided at a remote server, and thus has an honest-but-curious attribute. Therefore, threats continuously occur from insiders or from outside the system, and an encryption technique is thus required to prevent the data source from being known even when the data are leaked, as shown in Fig. 4. However, even if encrypted data are the same as the original data, the result depends on the encryption key used for encryption, and thus a data comparison based on the data deduplication technique cannot be conducted. To solve this problem, convergent encryption (CE), shown in Fig. 5, has been proposed [4].

CE is a technique for hashing a data source so as to generate hash data of a fixed size and using the source as an encryption key. Therefore, even if different users encrypt the data, the same encryption key and ciphertext are always generated. M. Storer proposed a technique for the deduplication of secure data using this approach [3]. First, key generation and encryption are conducted through the CE process. Thereafter, it is determined whether the data have been duplicated using the identifier of the encrypted data, and the data are additionally stored according to the result. Therefore, encryption and deduplication can be conducted concurrently. CE is used in most data deduplication environments, and various types of technologies have been proposed to utilize this technique. However, CE incurs a threat of poison attacks caused by different data and their identifier, as well as dictionary attacks that infer the data sources.

2.4. Dictionary Attack

A dictionary attack is an attack in which a ciphertext is decrypted, or related information is obtained, using pre-existing data. CE uses a method for hashing the data source to generate an encryption key, and encrypting it using the encryption key. Therefore, if an attacker knows the data source, the attacker can find the encrypted data and the encryption key, as shown in Fig. 6. Therefore, the attacker attempts an attack using a data list that is guessed as the data source. First, the attacker obtains the key by hashing the data that are guessed as the data source. When encryption is conducted using an encryption key and a data source, password data are generated. By comparing the generated secure data with a ciphertext maintained in storage, it is possible to confirm that the data generated by the attacker matches the original stored data. Such threats are quite serious in environments where data are predictable [5].

2.5. Poison Attack

A poison attack is a threat that occurs when the data stored do not match the actual data, as shown in Fig. 7 [6]. When a poison attack occurs in data M, two types of threats can be created:

- **Loss of data source:** If the user who acquired the ownership by performing a subsequent upload to the data where the poison attack occurred deletes the data source from the local storage, the original data can't be acquired again.
- **Damage due to malicious code or modified data:** Failure to detect tampering of data when downloading poison attack data may result in damage due to malicious code contained in the tampered data or property damage due to tampering with sensitive information.

To protect user's data from poison attacks, user need to be able to check whether stored data and metadata are generated from the same data. The RCE used in this study compares the tag with the data source and performs integrity verification after downloading

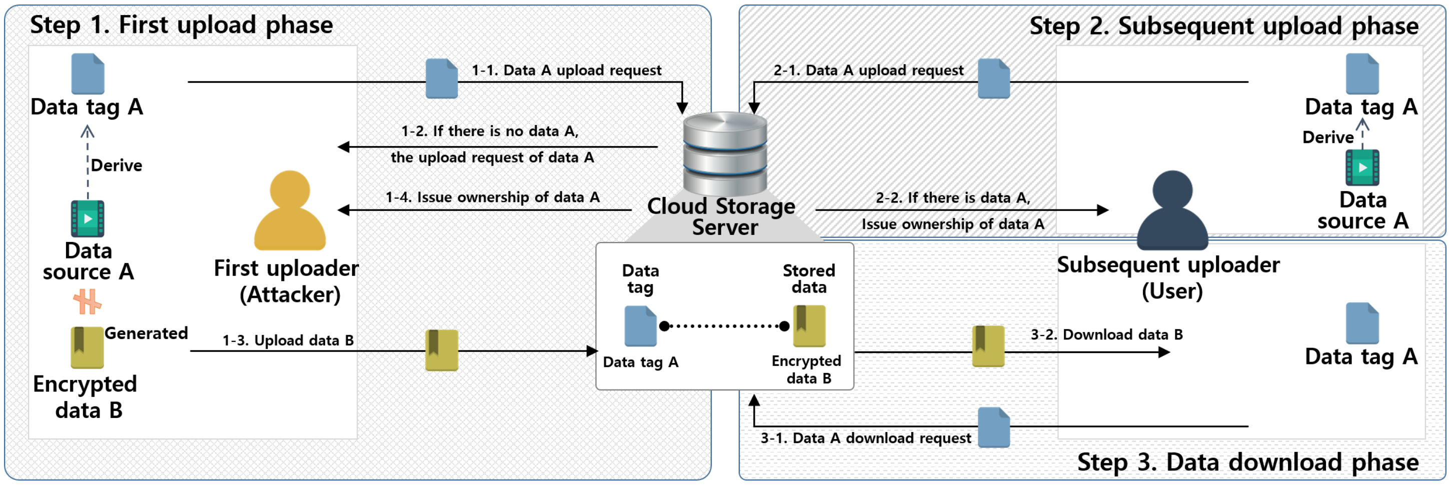
Step 1. First upload phase
Data tag A
Derive
Data source A
Generated
Encrypted data B
First uploader (Attacker)
1-1. Data A upload request
1-2. If there is no data A, the upload request of data A
1-4. Issue ownership of data A
1-3. Upload data B
Cloud Storage Server
Data tag
Stored data
Data tag A
Encrypted data B
Step 2. Subsequent upload phase
2-1. Data A upload request
2-2. If there is data A, Issue ownership of data A
Data tag A
Derive
Data source A
Subsequent uploader (User)
3-2. Download data B
Data tag A
3-1. Data A download request
Step 3. Data download phase

Figure 7. Poison attack scenario

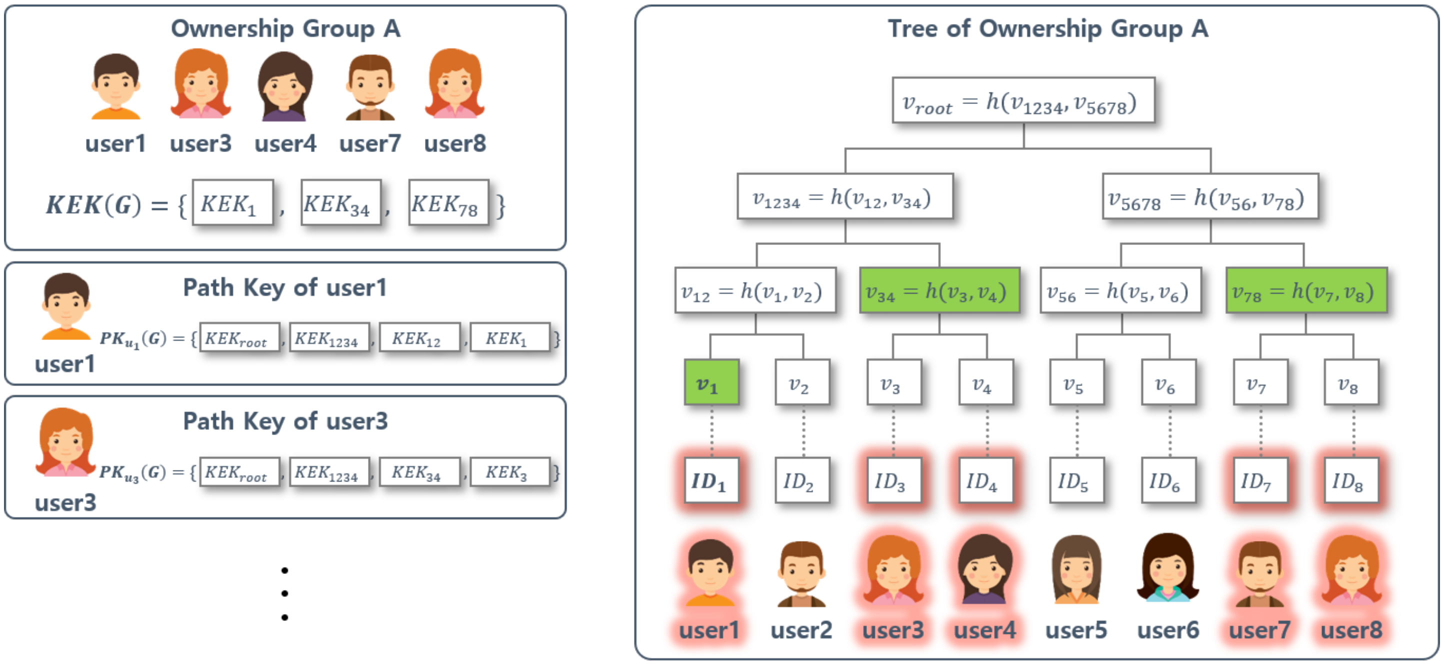
Ownership Group A
user1
user3
user4
user7
user8
$KEK(G) = \{KEK_1, KEK_{34}, KEK_{78}\}$
Path Key of user1
$PK_{u_1}(G) = \{KEK_{root}, KEK_{1234}, KEK_{12}, KEK_1\}$
user1
Path Key of user3
$PK_{u_3}(G) = \{KEK_{root}, KEK_{1234}, KEK_{34}, KEK_3\}$
user3
Tree of Ownership Group A
$v_{root} = h(v_{1234}, v_{5678})$
$v_{1234} = h(v_{12}, v_{34})$
$v_{5678} = h(v_{56}, v_{78})$
$v_{12} = h(v_1, v_2)$
$v_{34} = h(v_3, v_4)$
$v_{56} = h(v_5, v_6)$
$v_{78} = h(v_7, v_8)$
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
ID_1 ID_2 ID_3 ID_4 ID_5 ID_6 ID_7 ID_8
user1 user2 user3 user4 user5 user6 user7 user8

Figure. 8. Method of Ownership Management with Re-encryption

the data. However, since the direct association between the tag generated from M and the cipher text C can't be found in the data uploading stage, it is vulnerable to a poison attack.

2.6. Message-Locked Encryption

Message-locked encryption (MLE) offers four types of cryptographic techniques for the deduplication of encrypted data using technology proposed by Bellare et al. [7]. As a common MLE approach, data source M generates an encryption key K by computing K←H(M). Four types of MLE have been developed: CE, hash and CE without a tag check (HCE1), hash and CE with a tag check (HCE2), and randomized convergent encryption (RCE). The CE and HCE1 methods conduct only encryption and decryption. The HCE2 method conducts encryption and decryption along with tag integrity verification. The RCE method is based on HCE2, and the key generation entropy is improved through the generation of a randomized key in a one-time pad during the key generation process. In this study, a protocol design using RCE was applied.

2.7. Proof of Ownership (PoW)

In client-side deduplication technology, if the uploaded data are duplicated, ownership is issued without uploading the data. However, a problem arises if an attacker attempts to upload data by forging a data identifier. In this case, the attacker can take ownership of the data without the data source. To solve this problem, a technique is proposed to verify data ownership without transferring the data source. This technique is referred to as proof of ownership (PoW) [8].

2.8. Dynamic Ownership Management

A variety of data are stored on a cloud, and each data group has its own ownership group. However, if a change occurs, such as the issuing and discarding of ownership, a problem arises in that all ownership of the ownership group must be renewed. In general, it is relatively easy to issue ownership to a new user. However, if ownership information for a particular portion of data changes, it is difficult to change the ownership previously issued to the user. Therefore, dynamic ownership management technology is necessary to manage changes in ownership information in real time. In this study, we use proxy re-encryption for dynamic ownership management.

2.9. Hur et al.'s scheme

Hur et al.'s scheme was proposed in 2016 [9]. The authors proposed a method for achieving dynamic ownership management in a secure data deduplication environment. This scheme addresses the problem of identifying the data owner using the ownership information of the user by providing anonymity of ownership. As a way to provide anonymity of ownership, a general method for confirming ownership through the ownership group was developed. However, this approach complicates the ownership management because the ownership of the entire group must be changed at the time of ownership issue and renewal. re-encryption was proposed to solve this problem, along with a dynamic ownership management technology providing better efficiency. However, this scheme is based on a server-side deduplication environment, is continuously affected by the redundancy rate of the data, and always involves the same number of computations. Therefore, in the present study, we researched an improved technology by applying the idea of Hur et al.'s scheme.

In Her et al.'s scheme, ownership management is performed using the Merkle Hash Tree (MHT) for dynamic management of data ownership. In the leaf node of the MHT, the identification information of the users is located, and the MHT is configured as shown in Fig. 8 using this identification information. Ownership group G_i contains a list of users having ownership of data M_i. Also, $KEK(G_i)$ is configured as shown in the left side of Fig. 8 so that users included in G_i can be included in the minimum number of nodes. Therefore, user1, user3, user4, user7, user8 belong to group G and $KEK(G)$ consists of KEK_1, KEK_{34} and KEK_{78} in Fig. 8. Also, for each user, a node included in the user's own identifier path from the root node is provided as a path key(PK). In Fig. 8, PK_{u_1} (G) is PK of user1. And it includes $KEK_{root}, KEK_{1234}, KEK_{12}$, and KEK_1.

The ownership manager generates $KEK(G_i)$, which is a list of KEK of the ownership group G_i, and provides $PK_{u_1}(G_i)$ to the user. Then, after generating the group key GK_i of the group G_i, the data M_i is encrypted to generate the cipher text $C_i = E_{GK_i}(M_i)$. Finally, the group key GK_i is encrypted with $KEK(G_i)$ to complete the update of the ownership group for the data M_i.

Through the above process, the user belonging to the ownership group can obtain the data $M_i = D_{PK_{u_1}(G_i) \cap KEK(G_i)}(C_i)$ by using the $PK_{u_1}(G_i)$ held by the ownership group and the $KEK(G_i)$ provided by the ownership manager.

3. System Requirements

3.1. System configuration

The system structure required in this study is shown in Fig. 9. Users access the cloud storage, which includes storage and metadata servers. Therefore, the user stores the metadata server,

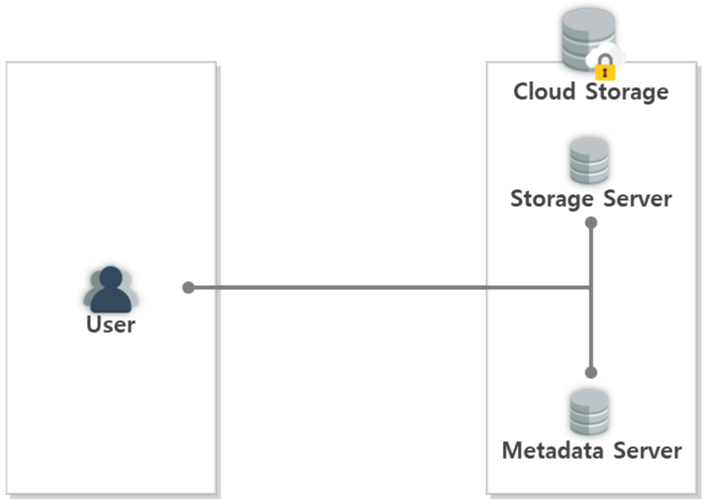

Figure 9. System configuration

data tag, and verification data, and the encrypted data are stored in the storage server.

3.2. Security requirements

The six security requirements are poison attack resistance, dictionary attack resistance, confidentiality, integrity, anonymity, and efficiency.

- **Confidentiality:** Data stored in cloud storage must be encrypted and archived in case of data leakage. Encryption should also be considered to enable data deduplication.
- **Integrity:** Data stored in cloud storage should be preserved without modification. If the data stored in the cloud storage are modified, the user should be able to detect the deformation when downloading the data.
- **Anonymity :** Ownership information stored in the server can only be used to verify the user's ownership, and information that can identify the user's identity should not be included.
- **Efficiency:** In the case of data deduplication, a large number of operations and data transmissions can occur. In this case, the computation and data traffic efficiency must be provided during the process because the benefits of the data deduplication are offset.
- **Resistance to poison attack:** A tag is derived from the data source and must be able to verify the integrity of the original data. This prevents poison attacks that may occur owing to the different tag sources and data stored in the server.
- **Resistance to dictionary attack:** An attacker should not be able to attack a dictionary. Therefore, an attacker should not be able to derive data or data encryption keys through the CE using an analogy with the data source.

4. Proposed Scheme

The proposed technique is based on Hur et al.'s concept of dynamic ownership management. In the proposed method, the data upload request phase is initially applied, and the first and subsequent data upload phases are conducted depending on whether the data are duplicated. If a poison attack is detected in the

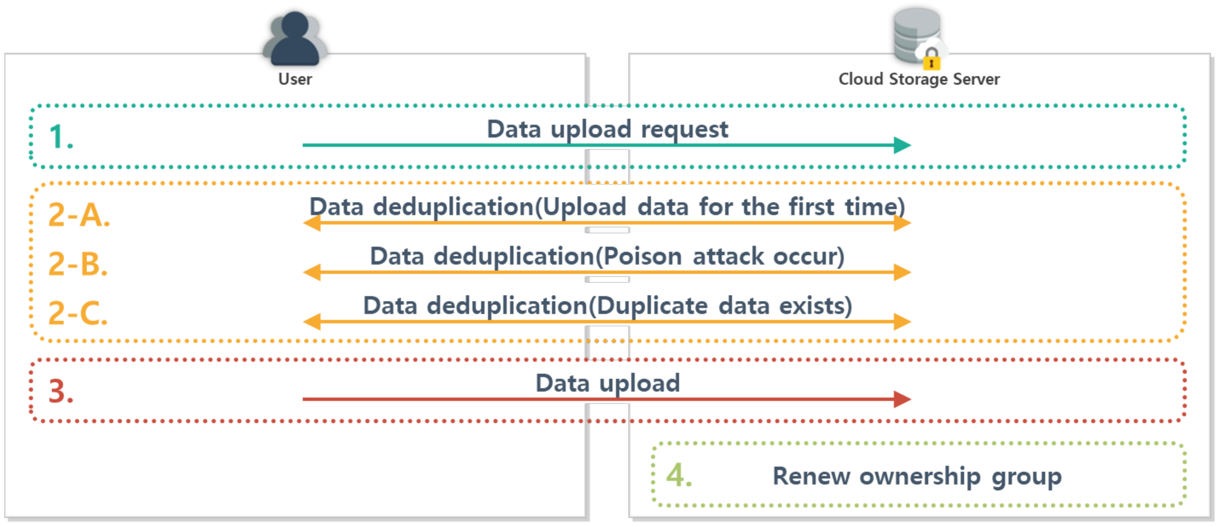
User
Cloud Storage Server
1.
Data upload request
2-A.
Data deduplication(Upload data for the first time)
2-B.
Data deduplication(Poison attack occur)
2-C.
Data deduplication(Duplicate data exists)
3.
Data upload
4.
Renew ownership group

Figure 10. Overview of uploading data in the proposed scheme

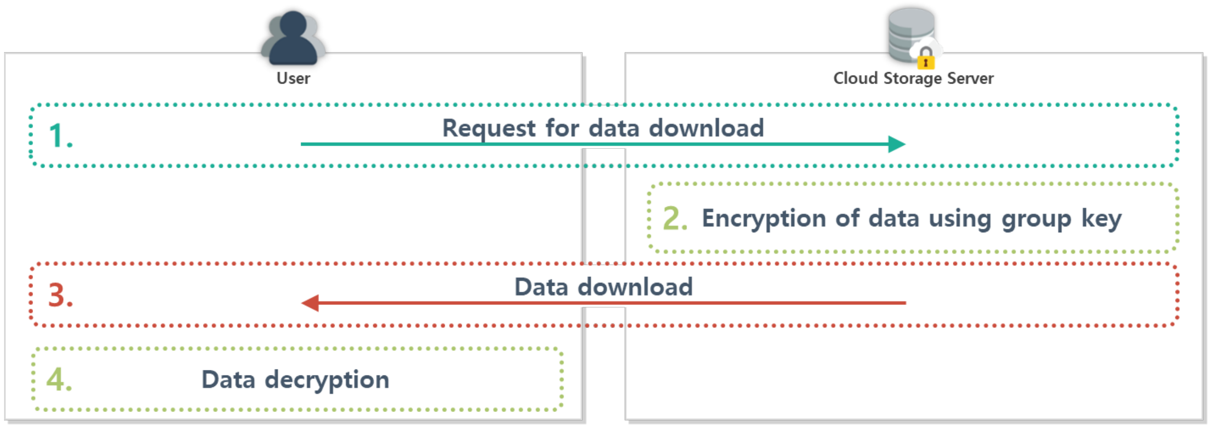

Figure 11. Overview of downloading data in the proposed scheme

$$K_i \leftarrow H(M_i) \tag{1}$$

4.1. Data Upload Request Phase

This step is a common step in the data upload process. At this stage, the user uses the identifier of the data to be uploaded, and requests the server to confirm whether the corresponding data exists, as shown in Fig. 12.

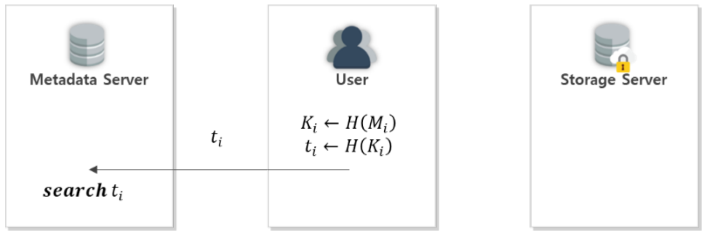

Figure 12. Data upload request phase

The user converts data M_i to be uploaded into a hash algorithm for generating K_i and t_i. The generated t_i is then transmitted to the metadata server to request confirmation of storage.

4.2. Data Upload Phase (First Upload)

This step is conducted when the corresponding data do not exist as a result of the data upload request phase, as shown in Fig. 13.

The metadata server determines that there are no data that a user has requested to upload and transmits the result to the user.

A user who receives a response from the metadata server generates L_i, C_i^1, C_i^2, and C_i^3.

$$L_i \overset{\$}{\leftarrow} \{0,1\}^{\lambda(K)} \tag{3}$$

$$C_i^1 \leftarrow E_{L_i}(M_i) \tag{4}$$

$$C_i^2 \leftarrow L_i \oplus K_i \tag{5}$$

$$C_i^3 \leftarrow H(M_i \parallel L_i) \tag{6}$$

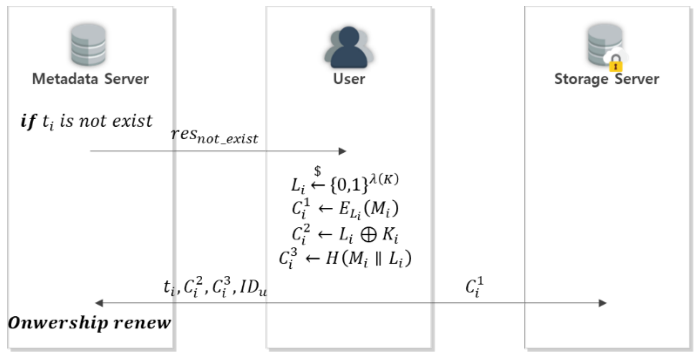
Metadata Server
User
Storage Server
if t_i is not exist
res_{not_exist}
$L_i \xleftarrow{\$} \{0,1\}^{\lambda(K)}$
$C_i^1 \leftarrow E_{L_i}(M_i)$
$C_i^2 \leftarrow L_i \oplus K_i$
$C_i^3 \leftarrow H(M_i \parallel L_i)$
t_i, C_i^2, C_i^3, ID_u
C_i^1
Onwership renew

Figure 13. Data Upload Phase (First Upload)

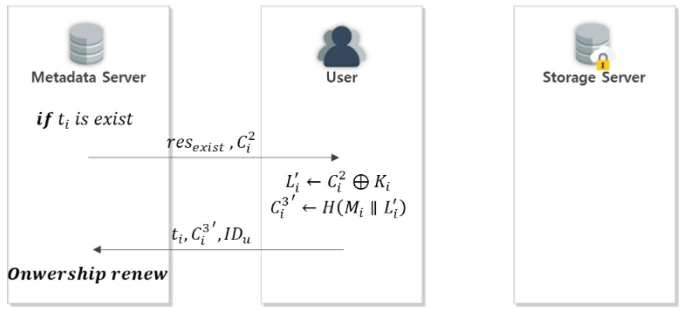
Metadata Server
User
Storage Server
if t_i is exist
res_{exist}, C_i^2
$L_i' \leftarrow C_i^2 \oplus K_i$
$C_i^{3\prime} \leftarrow H(M_i \parallel L_i')$
$t_i, C_i^{3\prime}, ID_u$
Onwership renew

Figure 14. Data Upload Phase (Subsequent Upload)

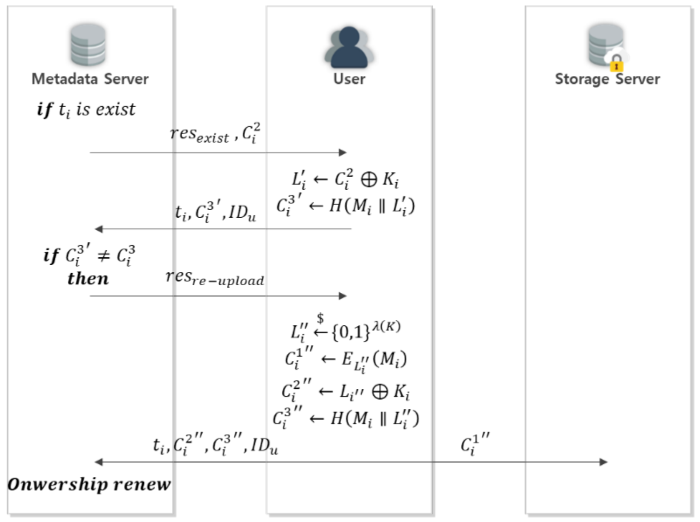
Metadata Server
User
Storage Server
if t_i is exist
res_{exist}, C_i^2
$L_i' \leftarrow C_i^2 \oplus K_i$
$C_i^{3\prime} \leftarrow H(M_i \parallel L_i')$
$t_i, C_i^{3\prime}, ID_u$
if $C_i^{3\prime} \neq C_i^3$ then
$res_{re-upload}$
$L_i'' \xleftarrow{\$} \{0,1\}^{\lambda(K)}$
$C_i^{1\prime\prime} \leftarrow E_{L_i''}(M_i)$
$C_i^{2\prime\prime} \leftarrow L_{i''} \oplus K_i$
$C_i^{3\prime\prime} \leftarrow H(M_i \parallel L_i'')$
$t_i, C_i^{2\prime\prime}, C_i^{3\prime\prime}, ID_u$
$C_i^{1\prime\prime}$
Onwership renew

Figure 15. Data Upload Phase (when a Poison attack occur)

The user obtains L_i by calculating data C_i^2 obtained from the metadata server. Then, $C_i^{3\prime}$ is generated using the obtained L_i.

$$L_i' \leftarrow C_i^2 \oplus K_i \tag{7}$$

$$C_i^{3\prime} \leftarrow H(M_i \parallel L_i') \tag{8}$$

The user sends the created t_i, C_i^3 and ID_u to the metadata server. The ownership group renewal is then conducted.

4.4. Data Upload Phase (when a Poison attack occur)

This step is conducted when it is determined that a poison attack has occurred on the data uploaded during the data upload phase (subsequent upload), as shown in Fig. 15.

The metadata server determines that there are data that the user has requested to upload, and transmits the result and C_i^2 to the user.

The user obtains L_i by calculating data C_i^2 obtained from the metadata server, as shown in (7). Then, $C_i^{3\prime}$ is generated using the obtained L_i, as shown in (8).

The user sends the created t_i, C_i^3, and ID_u to the metadata server.

The metadata server determines that a poison attack has occurred when the user-uploaded $C_i^{3\prime}$ is different from C_i^3 stored in the metadata server.

The metadata server then requests the user to re-upload the data. Ownership group renewal is then conducted. A user who receives a response from the metadata server generates L_i'', $C_i^{1\prime\prime}$, $C_i^{2\prime\prime}$, and $C_i^{3\prime\prime}$.

$$L_i'' \xleftarrow{\$} \{0,1\}^{\lambda(K)} \tag{9}$$

$$C_i^{1\prime\prime} \leftarrow E_{L_i''}(M_i) \tag{10}$$

$$C_i^{2\prime\prime} \leftarrow L_i'' \oplus K_i \tag{11}$$

$$C_i^{3\prime\prime} \leftarrow H(M_i \parallel L_i'') \tag{12}$$

The user sends the generated t_i, $C_i^{2\prime\prime}$, $C_i^{3\prime\prime}$ and ID_u to the metadata server and transmits $C_i^{1\prime\prime}$ to the storage server. Then, the ownership group renewal is conducted.

4.5. Ownership Group Renewal Phase

This step is executed after the data upload phase ends. Include the user who has been issued ownership in the ownership group. In this process, the metadata server encrypts and archives the GK_i using the added KEK. Fig. 16 shows the formation and management of the user's tree and ownership group.

The owner group G_i of data M_i has a key $KEK(G_i)$ for encrypting the group key GK_i. The $KEK(G_i)$ term is a KEK list of the minimum nodes that can include all user nodes included in G_i. For example, when $G_{data1} = \{u_1, u_3, u_4, u_7, u_8\}$, $KEK(G_{data1}) = \{KEK_1, KEK_{34}, KEK_{78}\}$ is established. The group update procedure is as follows.

The metadata server applies a data deduplication process with the user, and issues ownership of data M_i to user u_2.

The metadata server includes the user in ownership group G_i of data M_i. Then, $KEK(G_i)$ is updated. The added KEK is assumed to be KEK_2.

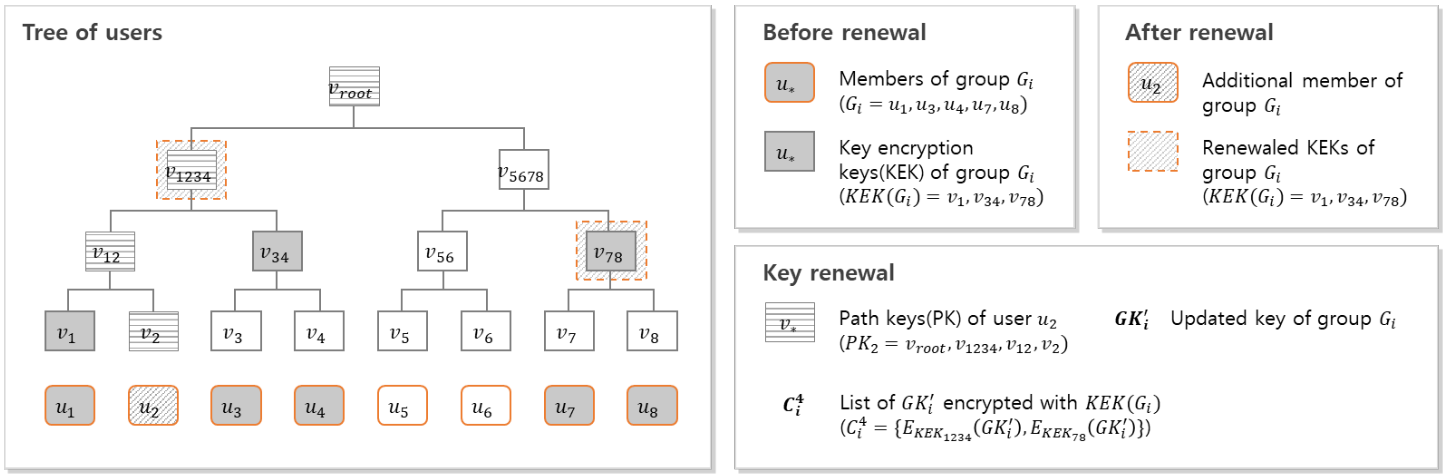
Tree of users
v_{root}
v_{1234}
v_{5678}
v_{12}
v_{34}
v_{56}
v_{78}
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8
Before renewal
Members of group G_i $(G_i = u_1, u_3, u_4, u_7, u_8)$
Key encryption keys(KEK) of group G_i $(KEK(G_i) = v_1, v_{34}, v_{78})$
After renewal
Additional member of group G_i
Renewaled KEKs of group G_i $(KEK(G_i) = v_1, v_{34}, v_{78})$
Key renewal
Path keys(PK) of user u_2 $(PK_2 = v_{root}, v_{1234}, v_{12}, v_2)$
GK_i' Updated key of group G_i
C_i^4 List of GK_i' encrypted with $KEK(G_i)$ $(C_i^4 = \{E_{KEK_{1234}}(GK_i'), E_{KEK_{78}}(GK_i')\})$

Figure 16. Method of renew ownership

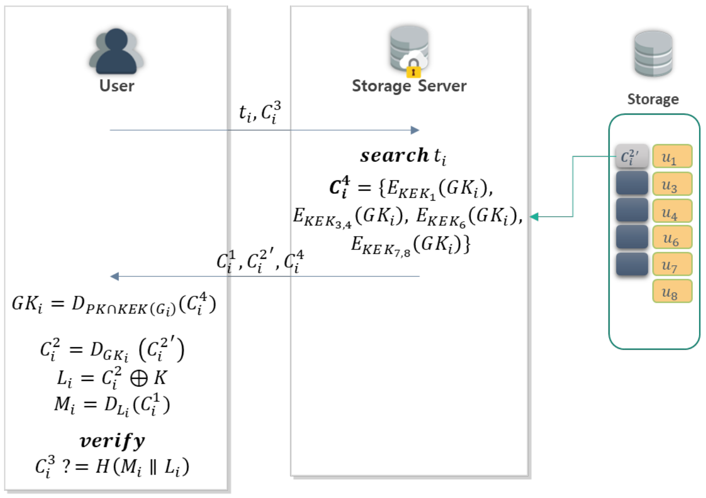
User
Storage Server
Storage
t_i, C_i^3
search t_i
$C_i^4 = \{E_{KEK_1}(GK_i), E_{KEK_{3,4}}(GK_i), E_{KEK_6}(GK_i), E_{KEK_{7,8}}(GK_i)\}$
$C_i^1, C_i^{2'}, C_i^4$
$GK_i = D_{PK \cap KEK(G_i)}(C_i^4)$
$C_i^2 = D_{GK_i}(C_i^{2'})$
$L_i = C_i^2 \oplus K$
$M_i = D_{L_i}(C_i^1)$
verify
$C_i^3 ?= H(M_i \parallel L_i)$
$C_i^{2'}$ u_1 u_3 u_4 u_6 u_7 u_8

Figure 17. Data download phase

$\mathrm{KEK(G_i)}$ is composed of KEK, which can cover all users included in group $\mathrm{G_i}$ with the minimum KEK by combining the added and existing KEK.

The metadata server encrypts C_i^4 with GK_i, and encrypts GK_i with KEK_{1234}.

4.6. Data Download Phase

This procedure corresponds to Fig. 17, and is conducted when a user who has been issued ownership of data $M_i$requests data to be downloaded from the server.

The user transmits identifier t_i of the data to be downloaded and verification data C_i^3 to the metadata server.

The server searches for data corresponding to $\mathrm{t_i}$, and then transmits $\mathrm{C_i^1, C_i^{2'}}$, and $\mathrm{C_i^4}$ stored in the server to the user.

The user obtains $\mathrm{M_i}$ using $\mathrm{C_i^1, C_i^{2'}}$, $\mathrm{C_i^4}$, and the user's own $\mathrm{PK_u}$ and $\mathrm{k_i}$.

$$GK_i \leftarrow D_{PK \cap KEK(G_i)}(C_i^4) \qquad (13)$$

$$C_i^2 \leftarrow D_{GK_i}(C_i^{2'}) \qquad (14)$$

$$M_i \leftarrow D_{L_i}(C_i^1) \qquad (16)$$

$$L_i \leftarrow C_i^2 \oplus K \qquad (15)$$

The user compares $\mathrm{C_i^3}$ and $C_i^{3'}$. This allows verification of the data integrity.

$$C_i^3 ?= H(M_i \parallel L_i) \qquad (17)$$

5. Analysis of Proposed Scheme

In this paper, we propose a more efficient and secure method for deduplication of encrypted data based on Hur et al.'s dynamic ownership idea [9]. Therefore, it has similar features to the Hur et al. method in certain areas, but differs in the following ways.

Confidentiality: Cloud storage is continuously exposed to data breaches. Therefore, data stored in the cloud must be encrypted and stored. However, because data deduplication and data encryption have opposite characteristics, we used CE to apply them concurrently. CE is a technique for obtaining an encryption key K by hashing the data source M as in (18). Therefore, the same encryption key and ciphertext are always generated for users who own the same data source. Therefore, using CE, it is possible to deduplicate the encrypted data. In this proposed scheme, data encryption is conducted using RCE mode of MLE based on CE.

$$K \leftarrow H(M) \qquad (18)$$

- **Integrity:** The user who has downloaded the data can verify whether the downloaded data have been transformed through the RCE tag validation. With this process, the user obtains data $chunk_i$ and conducts a hash operation using the encryption key L_i to create C_i^3, as shown in (19), which can be verified.

$$C_i^3 ?= H(M_i \parallel L_i), \qquad (19)$$

- **Anonymity :** This method is based on Hur et al.'s approach. When a user requests a data download, the cloud storage server sends the encrypted group key to the user. A user with legitimate ownership can decrypt the key and can verify that

Table. 1. Comparison with other technologies

				Hur, et al.	Kim, et al.	Proposed Scheme
Deduplication Location				Server Side	Client Side	Client Side
Poison Attack Resistance				△	○	○
Proof of ownership				○	X	○
Dynamic Ownership Management				○	X	○
User Anonymity				○	X	○
Compu-tation	Upload	First upload	User	2H + 1SE + 1⊕	3H + 1SE	3H + 1SE + 1⊕
		Subsequent upload	User	2H + 1SE + 1⊕	3H + 1SE	3H + 1⊕
		Poison attack occurred	User	2H + 1SE + 1⊕	3H + 1SE	4H + 1SE + 1⊕
	Download		User	1H + 3SE + 1⊕	1H + 1SE	1H + 3SE + 1⊕
	Ownership group renewal		Server	1H + 3SE + 1⊕	1H + 1SE	1H + 3SE + 1⊕

○: Offer; X: Not offer; △: limited offer;
H: Hash algorithm; SE: Symmetric key encryption; ⊕ : XOR Operation;

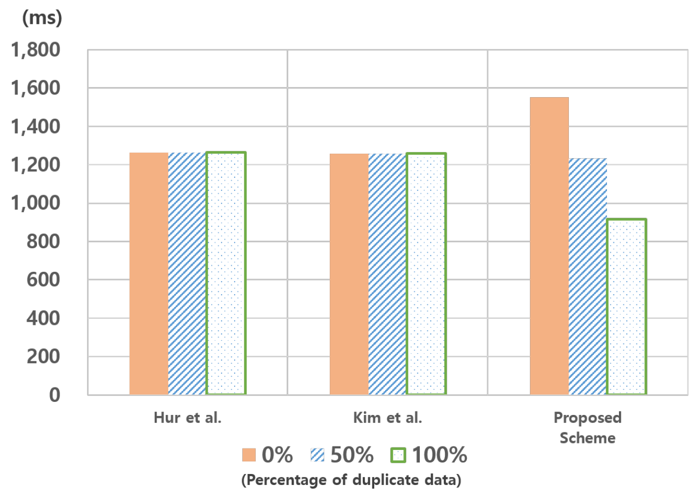

Figure 18. Computation time of 250k blocks

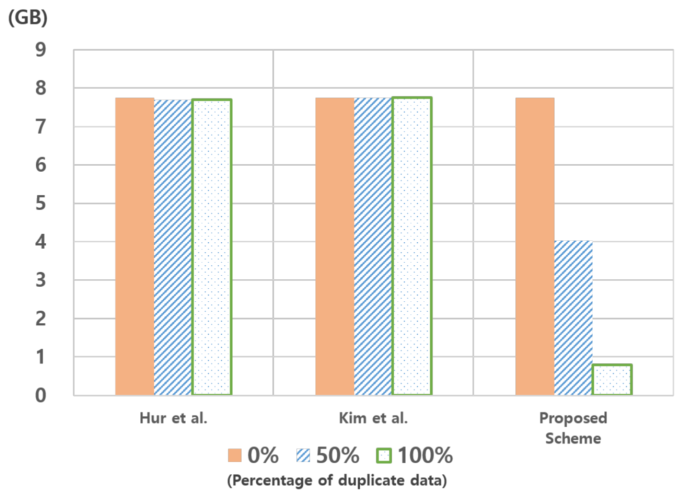

Figure 19. Data traffic of during deduplication of 250k blocks

it has legitimate ownership without identifying the user. Accordingly, the user has ownership of the corresponding data through this method, and thus downloads and decrypts the data through the process shown in (13-16).

- **Efficiency:** The proposed method shows that the total computational load decreases as the duplication ratio increases, the cause of which can be observed in Table 1. With the proposed method, one more hash operation per block is generated than in Hur et al.'s and Kim et al's method during the upload process when the first upload and poison attack occur[11, 9]. However, in the proposed scheme, the 3H + 1SE operation is omitted when redundant data exist. Therefore, the total number of operations can be reduced as the ratio of redundant data increases. In addition, the proposed scheme uses client-side deduplication. This method can conduct data uploads and deduplication with less computational and transmission overhead than server-side deduplication methods when uploading duplicated data, as shown in Fig 18, 19[10].

- **Resistance to poison attack:** In Hur et al.'s scheme, when a poison attack occurs, its occurrence can be checked at the download phase. However, data lost owing to a poison attack cannot be recovered. Therefore, the proposed scheme detects the poison attack in the subsequent upload phase to prevent data loss. In Hur et al.'s scheme, when a poison attack occurs, its occurrence can be checked during the download phase. However, data lost from a poison attack cannot be recovered. Therefore, the proposed scheme detects a poison attack during the subsequent upload phase to prevent a data loss [8].

- **Resistance to dictionary attack:** CE is vulnerable to dictionary attacks. Basically, because CE has a structure for obtaining an encryption key from a data source, the encryption key can also be obtained if the data source can be guessed. Therefore, to achieve resistance to a dictionary attack, data encryption key acquisition through data guessing should be made impossible. For this purpose, the proposed scheme conducts data encryption using RCE mode of MLE based on

CE. In RCE mode, the key obtained by hashing the data source is not used as a data encryption key, but is used as a key “K” for encrypting the data encryption key “L.” The formula for this is shown in (8),(21–23). Therefore, even if a data source is guessed, it is possible to resist a dictionary attack, which is an attack achieved through data guessing, because it does not obtain the data encryption key directly, but acquires a key capable of decrypting the data encryption key.

$$L \overset{\$}{\leftarrow} \{0,1\}^{\lambda(K)} \tag{21}$$

$$C^1 \leftarrow E_L(M) \tag{22}$$

$$C^2 \leftarrow L \oplus K \tag{23}$$

6. Conclusion

This study was conducted to improve the efficiency and security of data deduplication. Cloud storage wastes space owing to the redundant storage of the same data. Data deduplication has been proposed to solve this problem. Data deduplication is a technology that saves storage space by preventing the same data from being stored. Therefore, the uploaded data are compared with the data stored in the existing storage to make sure the data are the same. However, cloud storage always incurs a threat of data leakage, and thus data encryption should not be used to identify the data source. Therefore, cloud storage requires the data to be encrypted and archived through deduplication. However, CE was proposed to deal with the conflicting characteristics of data encryption and deduplication technologies. Because CE is a technology for generating an encryption key by hashing the data source, the same ciphertext is always generated when encrypting the same data source. It therefore becomes possible to deduplicate the encrypted data. However, CE poses a threat of a dictionary attack, and client-side deduplication poses a threat to a poison attack. Therefore, we use RCE mode of MLE to solve this problem. MLE is a CE-based encryption technology, and can prevent data-source guessing. In addition, the verification data generated in RCE mode can be used to identify the occurrence of a poison attack. In addition, when data deduplication is conducted, the ownership information of the user is updated. Because cloud storage is an environment used by multiple users, such ownership information is frequently updated. Therefore, ownership management difficulties may occur, and the identity of the user may be exposed when using such ownership information. To solve this problem, we applied ownership management technology using a binary tree based on Hur et al.’s scheme. Through this, this paper is safe from poison attack, dictionary attack, proof of ownership, ownership management and ownership anonymity. In addition, by applying a client-side deduplication environment, the number of computations and amount of communication can be reduced as the data redundancy ratio increases. As a result, the proposed scheme is effective against various security threats, and reduces the amount of computational traffic as the data redundancy ratio increases.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2016R1D1A1B03935917) and Barun ICT Research Center at Yonsei University.

References

[1] Kim, K. W., Joo, Y. H., Eom, Y. I. “Technical trends for cloud storage data deduplication”, Proceedings of Symposium of the Korean Institute of Communications and Information Sciences, 2012, pp. 228–229. DOI: 10.22648/ETRI.2018.J.330107

[2] Kim, W. B., Lee, I. Y., Ryou, J. C. "Improving dynamic ownership scheme for data deduplication." Computer Applications and Information Processing Technology (CAIPT), 2017 4th International Conference on. IEEE, 2017. DOI: 10.1109/CAIPT.2017.8320671

[3] Storer, M. W., Greenan, K., Long, D. D., Miller, E. L “Secure data deduplication”, Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, ACM, 2008, pp.1–10. DOI: 10.1145/1456469.1456471

[4] Douceur, J. R., Adya, A., Bolosky, W. J., Simon, P., Theimer, M. "Reclaiming space from duplicate files in a serverless distributed file system." Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on. IEEE, 2002. pp. 617-624. DOI: 10.1109/ICDCS.2002.1022312

[5] Bellare, M., Keelveedhi, S., Ristenpart, T. "DupLESS: Server-Aided Encryption for Deduplicated Storage." IACR Cryptology ePrint Archive 2013 (2013): 429.

[6] Kaaniche, N., Laurent, M. "A secure client side deduplication scheme in cloud storage environments." NTMS 2014: 6th International Conference on New Technologies, Mobility and Security. 2014. pp. 1-7. DOI: 10.1109/NTMS.2014.6814002

[7] Bellare, M., Keelveedhi, S., Ristenpart, T. "Message-locked encryption and secure deduplication." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2013, pp. 296-312. DOI: 10.1007/978-3-642-38348-9_18

[8] Halevi, S., Harnik, D., Pinkas, B., Shulman-Peleg, A. “Proofs of ownership in remote storage systems”, Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011, pp. 491-500. DOI: 10.1145/2046707.2046765

[9] Hur, J., Koo, D., Shin, Y., Kang, K. “Secure Data Deduplication with Dynamic Ownership Management in Cloud Storage”, IEEE Transactions on Knowledge and Data Engineering, 28(11), 2016, pp. 3113-3125. DOI: 10.1109/TKDE.2016.2580139

[10] Kim, K., Youn, T. Y., Jho, N. S., Chang, K. Y. "Client-Side Deduplication to Enhance Security and Reduce Communication Costs." ETRI Journal 39(1), 2017, pp. 116-123. DOI: 10.4218/etrij.17.0116.0039