Copyright ⓒ 2019 KSII

# Secure and Efficient Storage of Video Data in a CCTV Environment

**Won-Bin Kim and Im-Yeong Lee***
Department of Computer Science and Engineering, Soonchunhyang University
Asan, South Korea, 31538
[e-mail: wbkim29@sch.ac.kr, imylee@sch.ac.kr ]
*Corresponding author: Im-Yeong Lee

*Received October 2, 2018; accepted January 17, 2019; published June 30, 2019*

## Abstract

Closed-circuit television (CCTV) technology continuously captures and stores video streams. Users are typically required by policy to store all the captured video for a certain period. Accordingly, increasing the number of CCTV operation cycles and photographing positions expands the amount of data to be stored. However, expanding the available storage space for video data incurs increased costs. In recent years, this problem has been addressed with cloud storage solutions, which enable multiple users and devices to access and store data simultaneously. However, because of the large amount of data to be stored, a vast storage space is required. Consequently, cloud storage administrators need a way to store data more efficiently. To save storage space, deduplication technology has been proposed to prevent duplicate storage of the same data. However, because cloud storage is hosted on remote servers, data encryption technology must be applied to address data exposure issues. Although deduplication techniques for encrypted data have been studied, there have been various security vulnerabilities. We attempted to solve this problem by addressing various issues such as poison attacks, property forgery, and ownership management while removing the redundant data and handling the data more securely.

***Keywords:*** Cloud storage, data deduplication, data encryption, closed-circuit television (CCTV), ownership management

A preliminary version of this paper was presented at APIC-IST 2018, and was selected by the conference review process.

**http://doi.org/10.3837/tiis.2019.06.026** **ISSN : 1976-7277**

## 1. Introduction

Recently, the use of a closed-circuit television (CCTV) for crime prevention has increased. In this environment, a CCTV captures video streams from a fixed location and stores the captured video on a separate storage device. Therefore, a separate server must be configured to hold the data. CCTV images are increasingly stored using cloud storage.

Cloud storage is a remote server environment that allows data access and storage, from anywhere and at any time, over the network. Cloud storage must be continuously available as multiple users can access, store, and use data simultaneously. Therefore, data storage space must be periodically expanded at a cost to ensure continuous availability of the data store. However, typical cloud storage solutions often store the same data repeatedly, occupying a large amount of space and wasting storage space, because the uploaded data is generated based on a predefined format and is often shared on the Internet. Deduplication is proposed to address this issue

Data deduplication technology is a technique that prevents repeated storage of the same data [1]. To do this, the system determines whether the uploaded data is already stored on the storage device, thereby improving data efficiency. Applying data deduplication to cloud storage that is used by a large number of users can save a large amount of data storage and can significantly benefit data storage space availability.

However, data deduplication can cause data confidentiality and integrity issues when applied to cloud storage for CCTV data. Once these issues are resolved, the CCTV data can be safely and efficiently stored and used with cloud storage. In this study, we used data deduplication technology to efficiently store CCTV images in a cloud storage environment. In addition, we aimed to provide a safe and efficient CCTV operating environment by solving various security issues that may arise during the data deduplication process.

## 2. Related Works

This section describes previous research related to this study.

### 2.1 Data Deduplication

When data is stored in a deduplication solution, the storage must ensure that the same data already exists on the storage system or device, as shown in **Fig. 1**. Therefore, the data deduplication system compares new data with already stored data. To do this, the data source is hashed and then data comparison is performed. Various shapes can be designed and applied in this process [2].

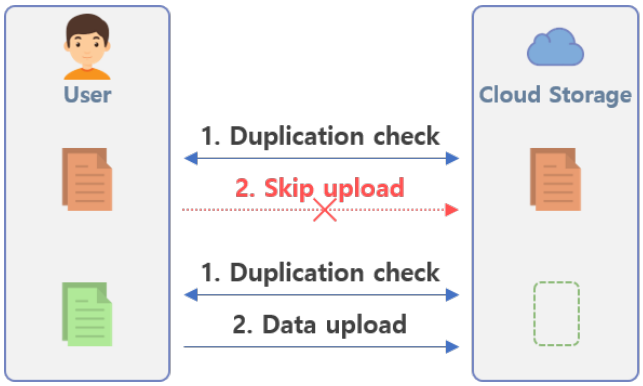


**Fig. 1.** Data deduplication

## 2.2 Data Deduplication Methods

There are several types of processes available for removing redundant data. The first proposed data deduplication system sends all the data to be uploaded to the server, as shown in **Fig. 2**. Then, the server performs a deduplication process. This form is server-side deduplication. However, since all data containing duplicate data is transferred, the amount of duplicate data grows regardless of the redundancy of the transfer data. In addition, multiple users uploading data at the same time can result in bottlenecks. To address this issue, client-side deduplication (CSD) technology was proposed [3, 4].

CSD is a method of checking for redundancy by comparing the list of data items stored with the list of identifiers of the data to be uploaded by the client, as shown in **Fig. 3**. The server checks for duplicate data and sends a non-duplicate list of data to the client. Clients use a list of received data to perform deduplication. Only deduplicated data is transferred to the cloud storage server. As a result, CSD can reduce data transfer traffic and workload on the

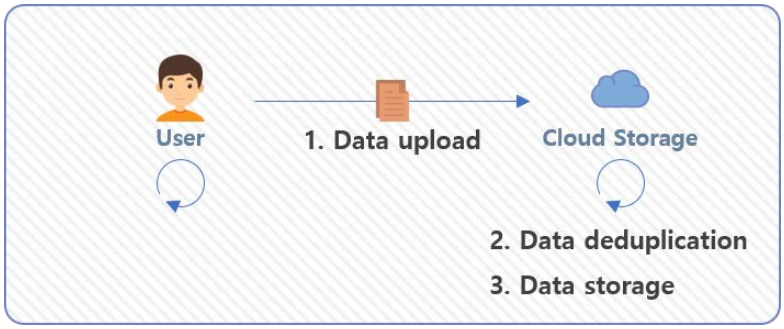


**Fig. 2.** Method of server-side deduplication

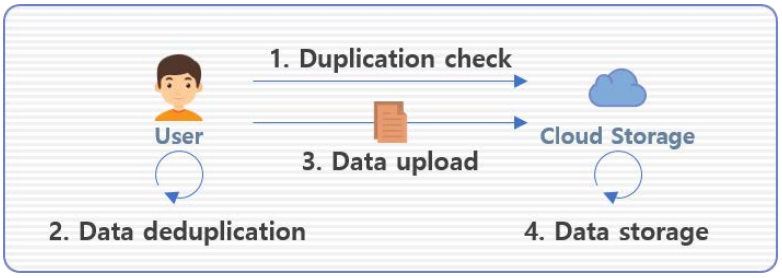


**Fig. 3.** Method of client-side deduplication

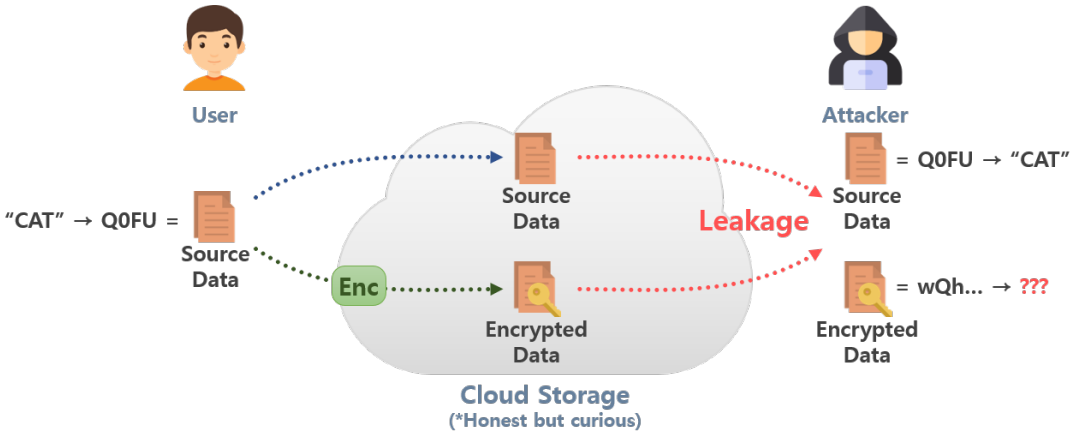


**Fig. 4.** Importance of data encryption

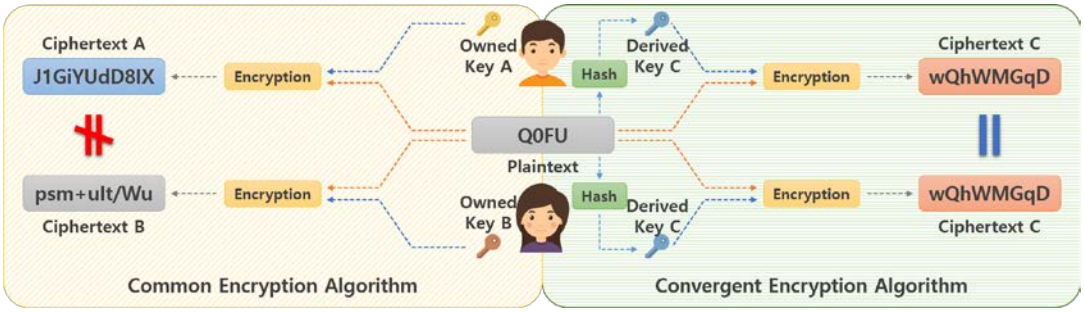


**Fig. 5.** Difference between general encryption and convergent encryption

server side. In this paper, the CSD is used to exploit these advantages [5, 6].

## 2.3 Convergent Encryption

Data deduplication technology compares hash data to determine if data is duplicated. However, cloud storage is hosted on a remote server. So, encryption techniques are required to ensure that the data sources are not prone to data leakage due to security threats from insiders or external cybercriminals, as shown in **Fig. 4**. However, because encryption results depend on the encryption key used for encryption, deduplication of encrypted data cannot be performed, as shown in **Fig. 4**. To solve this issue, convergent encryption (CE) has been proposed, as shown in **Fig. 5** [6, 7, 8].

CE generates hash data from the data source and uses this as the encryption key. Therefore, encrypting data always generates the same encryption key and password text, even if you are different people. Storer has proposed a deduplication technology for secure data using this method [6]. First, the key creation and encryption is done using CE. It then uses the identifier of the encrypted data to determine if the data has been copied and stores the data in accordance with the result. This enables deduplication of the encrypted data. Currently, CE is standard for deduplication. In addition, a variety of technologies have been proposed by applying this CE's idea. However, CE has the threat of dictionary attack and poison attack.

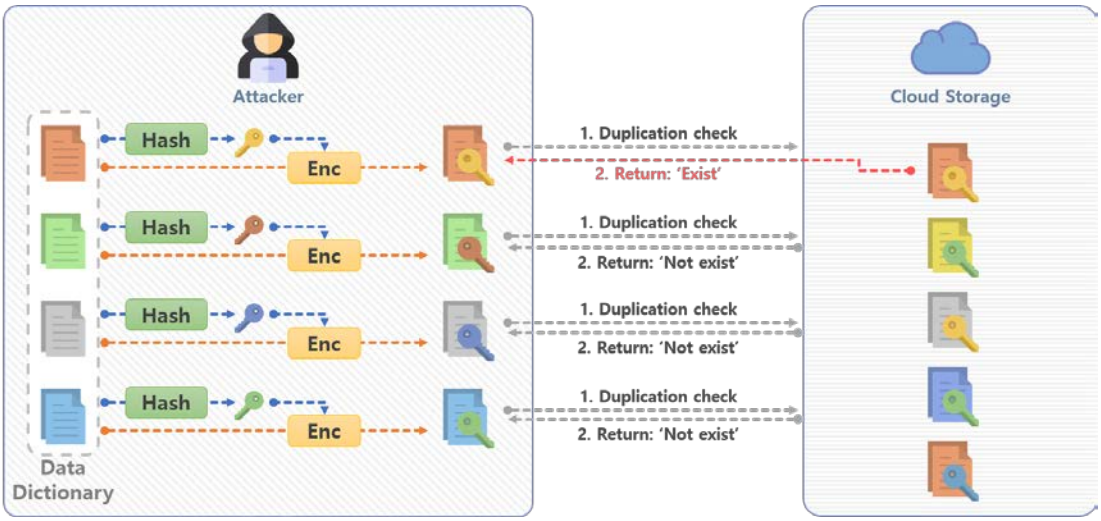


**Fig. 6.** Dictionary attack scenario

## 2.4 Dictionary Attack

Dictionary attacks use existing data to decrypt or obtain information. CE uses a method of hashing data sources to generate encryption keys and encrypt data using hash data as encryption keys. As a result, if an attacker can predict the source of the data, then encrypted data and encryption keys can be found, as shown in **Fig. 6**. As a result, an attacker tries to attack by using a list of data that are expected to be data sources. First, the attacker hashes the source to obtain the expected key of the data. Encryption is performed using the generated key and the expected data source. An attacker can compare a forged ciphertext to a ciphertext stored in that storage to verify that the forged ciphertext matches the ciphertext of the data source. In a predictable data environment, this threat is highly serious [9].

## 2.5 Poison Attack

A poison attack is a threat that prevents stored data from being matched with metadata for browsing data in the storage, as shown in **Fig. 7** [9]. A Poison attack on Data M can cause two types of threats:

- **Data source loss:** Users who acquire ownership through subsequent uploads to storage exposed to Poison attacks cannot recover their original data if they have deleted the data source from the local repository.
- **Damage to malware or modified data:** If data changes are not detected when downloading Poison attack data, data can be corrupted by asset loss or malicious in code due to tampered data that can affect sensitive data.

To protect user data from Poison attacks, users must be able to determine whether stored data and metadata are generated from the same source. The RCE (Randomized Convergent Encryption) used in this study performs data integrity checks by comparing tags with data sources. However, the direct association between tags created by $M$ and cryptogram $C$ is vulnerable to poison attacks because they cannot be found during the data upload phase.

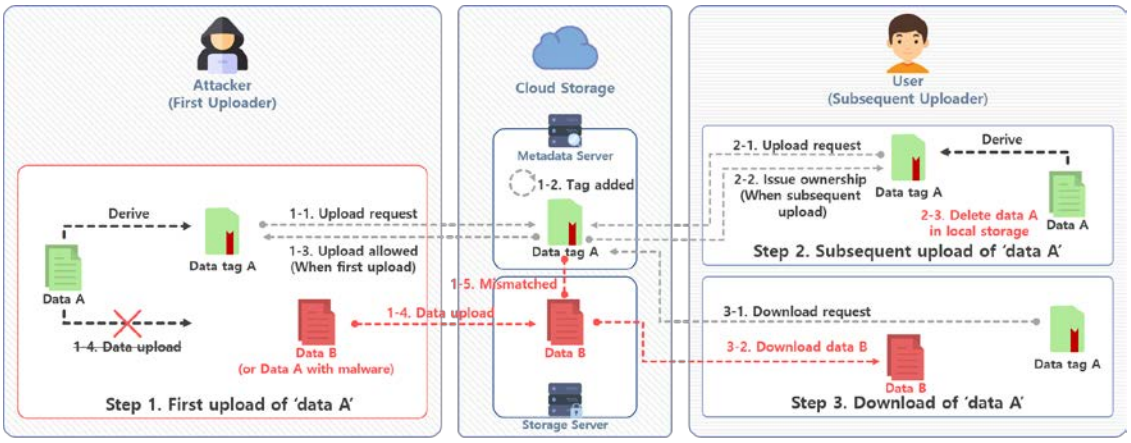


**Fig. 7.** Poison attack scenario

## 2.6 Message-Locked Encryption

Message-locked encryption (MLE) provides 4 types of encryption approaches for the deduplication of encrypted data using techniques proposed by Bellare et al. [6, 10]. In a typical MLE approach, the data source $M$ computes $K \leftarrow H(M)$ to generate an encryption key $K$. Four types of MLEs have been developed: CE without HCE1, hash with CE, hash with tag check (HCE2), and CE with RCE (random integrated encryption). CE and HCE1 methods only perform encryption and decryption. The HCE2 method performs encryption and decryption with tag integrity verification. The RCE method is based on HCE2 and improves key generation entropy by generating random keys on one pass pad during generation of key. In this paper, RCE is used for protocol design.

## 2.7 Deduplication of Video Data

Deduplication of video data requires a separate process from deduplication of common document files. Duplicating video data by dividing the entire file into specific sizes, as in a regular document file, results in a very small percentage of duplicate data. Therefore, in order

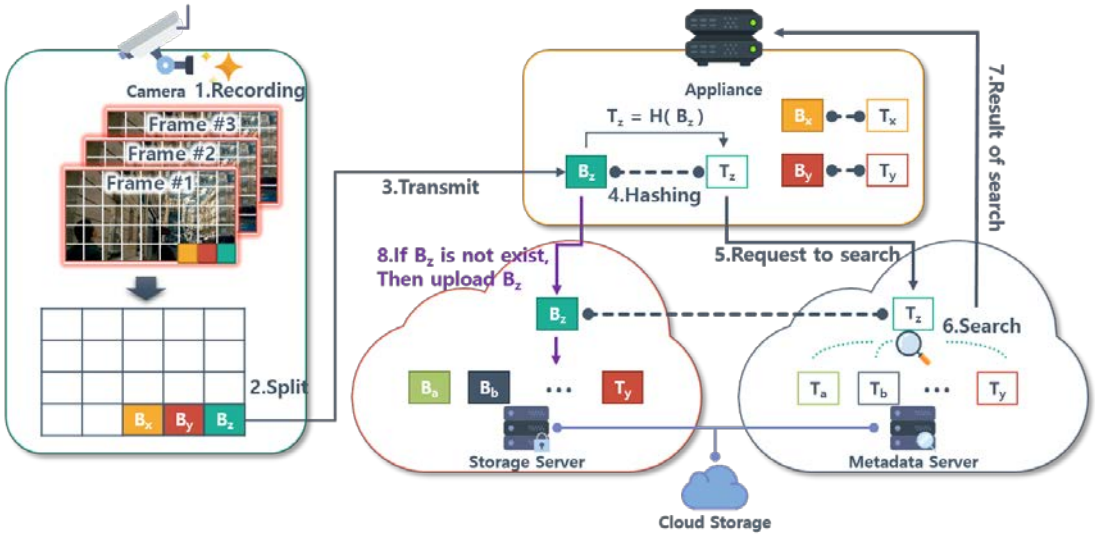


**Fig. 8.** Application of data deduplication technology in a CCTV environment

to deduplicate the video data, it is necessary to understand the structure of the video data and perform deduplication according to the structure of the video data, as shown in **Fig. 8**.

### 2.8 Dynamic Ownership Management

Cloud storage stores a wide variety of data. In addition, each data file has an ownership group. However, it is inconvenient to update all ownerships of an owning group in events that occur frequently, such as adding or discarding data. It is simple to issue ownership to new users. However, changing or disposing of an already issued ownership is a difficult task because it affects other ownerships included in the ownership group. Therefore, managing changes in ownership in real time requires dynamic ownership management technology. In this paper, we propose a re-encryption technique that uses the Merkle tree to manage dynamic ownership [7, 11, 12].

Dynamic ownership management technology proposed by Hur et al. provides anonymous ownership to address the issue of identifying data owners with user ownership information [11, 13, 14]. Typically, the system uses ownership groups to provide ownership anonymity. However, this approach makes it difficult to manage ownership because it must manage new ownerships and update all existing ownerships in the ownership group. To solve this issue, a method of using re-encryption in a proxy server has been proposed. This allows the system to perform ownership renewals more efficiently. However, this method is designed in terms of deduplication on the server side, so the same amount of computation is always required. In this paper, we modified the method of Hur et al. to make it more efficient for client-side deduplication.

### 2.9 Scheme of Hur et al.

In 2016, Hur et al. proposed management of dynamic ownership scheme [11]. This scheme addresses the problem of identifying data owners using your ownership information by providing anonymous ownership. A method of verifying ownership through ownership groups has been developed in a way that provides anonymity of ownership. This method complicates ownership management. However, this is because the system must change ownership of the entire group to issue and renew ownerships. Re-encryption, with dynamic ownership management technology, which improves efficiency, has been used to address this issue. However, this approach is based on a server-side deduplication environment and is constantly affected by data redundancy and always includes the same number of calculations, regardless of whether or not the data is duplicated. Therefore, improved techniques were studied in this paper by using the techniques of Hur et al.

In the scheme of Hur et al., Use the Merkle hash tree (MHT) to dynamically manage data ownership. The leaf node of the MHT locates information of the user's identification and comprises the MHT, as shown in **Fig. 9**. The ownership group $G_i$ represents a list of users who own the data $M_i$. The $KEK(G_i)$ is also configured as shown on the left side of **Fig. 9**, so that the users included in $G_i$ can be included in a minimum number of nodes. Therefore, user1, user3, user4, user7, and user8 belong to group G, and $KEK(G)$ consists of $KEK_1$, $KEK_{34}$, and $KEK_{78}$, as shown in **Fig. 9**. For each user, the path from the root node to the user's own identifier is also provided with the path key (PK). In **Fig. 9**, $PK_{u_1}$(G) is the PK of user1, and it includes $KEK_{root}, KEK_{1234}, KEK_{12}$, and $KEK_1$.

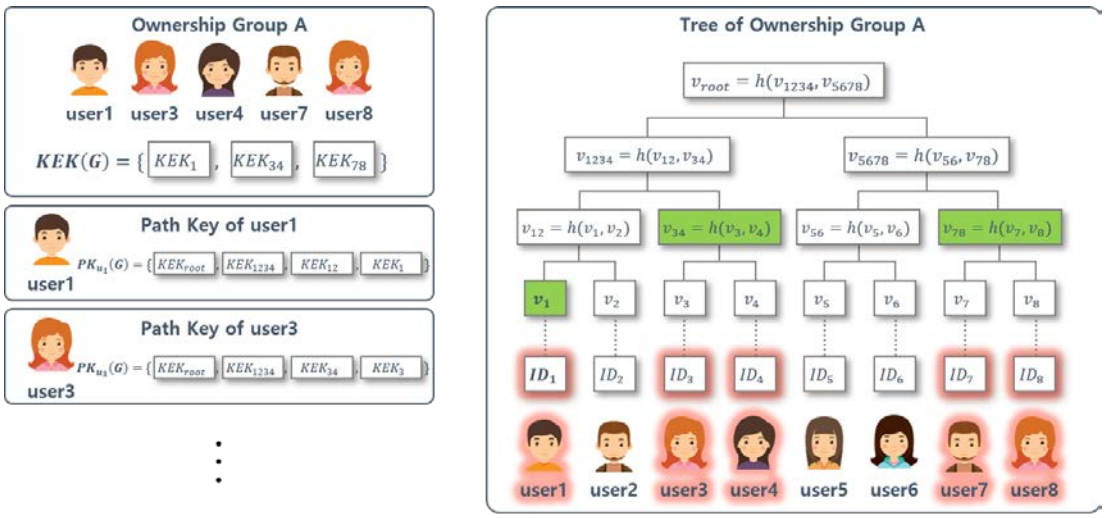


**Fig. 9.** Method of ownership management using re-encryption

The ownership manager creates $KEK(G_i)$, a KEK list of ownership groups $G_i$, and provides $PK_{u_1}(G_i)$ to the user. Next, after creating the group key $GK_i$ of group $G_i$, the data $M_i$ is encrypted to generate a cryptographic statement $C_i = E_{GK_i}(M_i)$. Finally, the group key $GK_i$ performs encryption using the $KEK(G_i)$ to complete the ownership group update of the data $M_i$.

Through this step, the users in the ownership group can obtain the data $M_i$ through the operation (1) using the $PK_{u_1}(G_i)$ held by the members of the ownership group and the $KEK(G_i)$.

$$M_i = D_{PK_{u_1}(G_i) \cap KEK(G_i)}(C_i) \tag{1}$$

## 3. System Requirements

In this section, we examine the system configuration and security requirements of this paper.

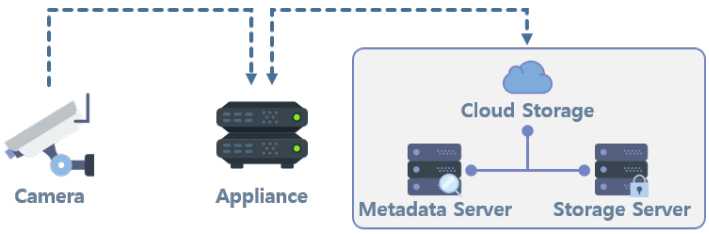


**Fig. 10.** System configuration

### 3.1 System Configuration

In this paper, we optimized the system configuration to deduplicate the CCTV images and store them securely in cloud storage. As shown in **Fig. 10**, the system components consist of cameras, appliances, and cloud storage. Cloud storage consists of a metadata server and a storage server.

### 3.2 Security requirements

The security requirements of this study are confidentiality, integrity, anonymity, efficiency, poison attack resistance and dictionary attack resistance.

- **Confidentiality:** Data stored in the cloud storage must be encrypted in case of data leakage. At the same time, users must also consider encryption that enables deduplication.
- **Integrity:** Data stored in cloud repositories must not be tampered with. When data stored on cloud storage changes without permission, users must be able to detect changes as they download the data.
- **Anonymity:** Ownership of the data should only be used to identify the user's ownership. Ownership information should not be able to specify who the user is.
- **Efficiency:** Deduplication can lead to many operations and data transfers. The benefits of deduplication are offset, so this method must provide computational and data traffic efficiency.
- **Poison attack resistance:** Tags are generated using data sources. You must be able to use this tag to verify the integrity of the original data. Use this method to prevent Poison attacks.
- **Dictionary attack resistance:** An attacker should be unable to dictionary attack. Therefore, an attacker must not be able to generate data or data encryption keys through data source analogy.

## 4. Proposed Scheme

The proposed scheme is based on the concept of dynamic ownership management as in the scheme of Hur et al. The first step in the proposed method is a request for data upload. At the data upload request stage, the subsequent upload method has a different process depending on whether the data is duplicated or not. In the data upload phase, the data upload phase (when a Poison attack occurs) is performed when the uploaded data is confirmed to be exposed to Poison attack. The data upload will be completed with the ownership group renewal phase. In addition, the data download phase is performed when the owning user requests the data to be downloaded, as shown in **Fig. 11**.

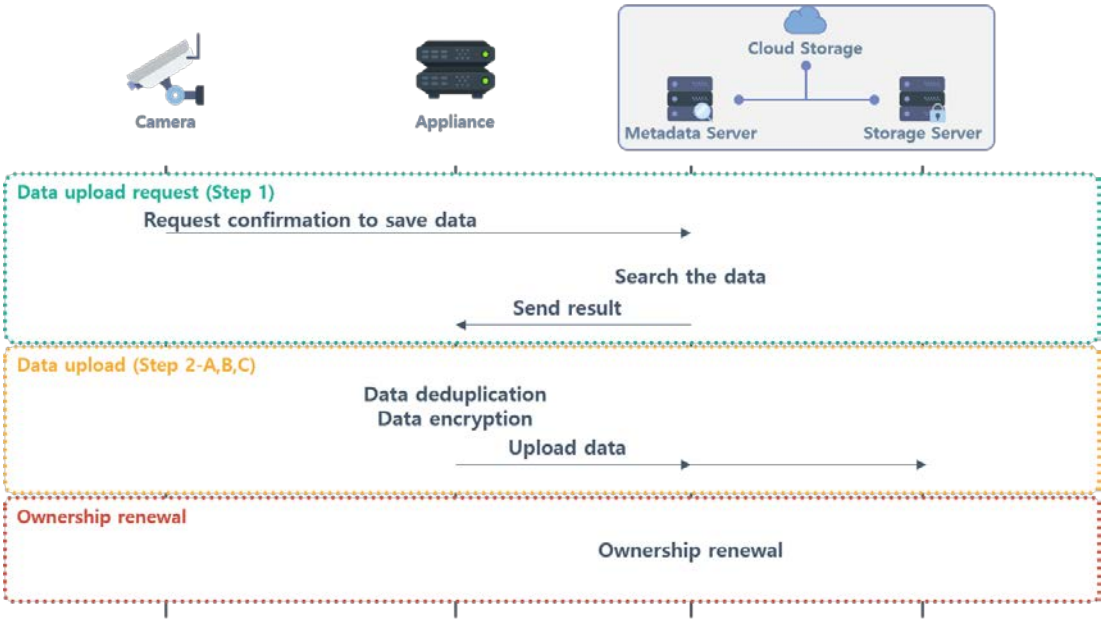


**Fig. 11.** Summary of the proposed scheme

### 4.1 System Parameters

The system parameters used in the proposed scheme are as follows.

- $*$: participant (CAM: CCTV camera; AS: appliance server; MS: metadata server; SS: storage server; u: user)
- i: Index of $chunk$
- j: Index of user
- $M$: List of $chunk_i$; source of image data
- $\text{chunk}_i$: $i$-th block piece of $M$
- $K_i$: Key derived from $chunk_i$
- $t_i$: Tag of $chunk_i$
- $L_i$: Encryption key of $\text{chunk}_i$
- $C_i^1$: Encrypted $\text{chunk}_i$
- $C_i^2$: Encoded key
- $C_i^3$: Verification data
- $G_i$: Ownership group of $\text{chunk}_i$
- $GK_i$: Group key of $G_i$
- KEK: Key encryption key
- $\text{KEK}(G_i)$: $KEK$ list of $G_i$
- $PK_{u_j}(G_i)$: Path key list of user j

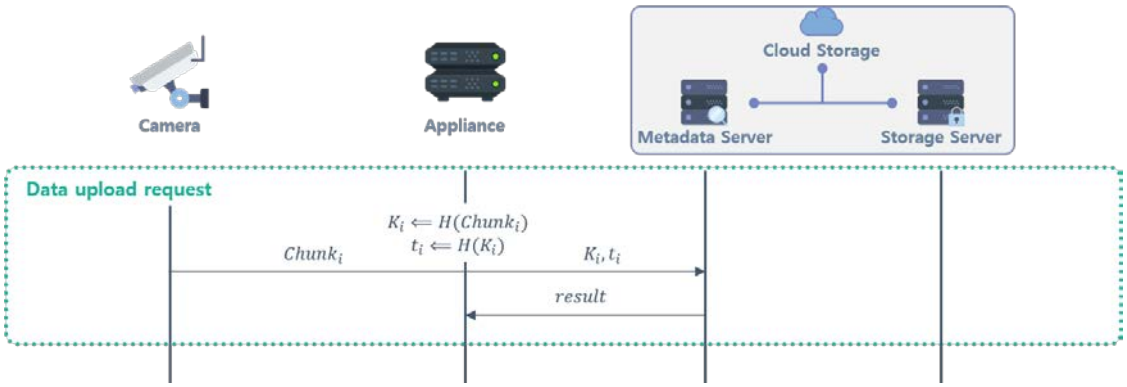


**Fig. 12.** Data upload request step

### 4.2 Data Upload Request Phase

At this stage, users use the data identifier for uploading data, which requests confirmation that the same data exists on the cloud storage server, as shown in **Fig. 12**. The appliance server uses a hash algorithm to generate $K_i$ and $t_i$ data to upload. The generated $t_i$ is sent to the metadata server and is used to request a storage confirmation:

$$M = \{chunk_0, chunk_1, \dots, chunk_i\} \tag{2}$$

$$K_i \leftarrow H(chunk_i) \tag{3}$$

$$t_i \leftarrow H(K_i) \tag{4}$$

Subsequently, the appliance transmits the generated tag $t_i$ to the metadata server. The metadata server explores whether the same data as the tag $t_i$ sent by the appliance exists on the cloud storage server, and notifies the appliance of the results.

### 4.3 Data Upload Phase (First Upload)

This step is performed when the data is not present in the data upload request phase, as shown in **Fig. 13**.

The metadata server notifies the user of the result that the data requested does not exist on that server. Users who receive a response from the metadata server create $L_i$, $C_i^1$, $C_i^2$, and $C_i^3$:

$$L_i \xleftarrow{\$} \{0,1\}^{\lambda(K)} \tag{5}$$

$$C_i^1 \leftarrow E_{L_i}(chunk_i) \tag{6}$$

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

$$C_i^3 \leftarrow H(chink_i \parallel L_i) \tag{8}$$

The appliance sends the generated $t_i$, $C_i^2$, $C_i^3$, and $ID_u$ to the metadata server and sends $C_i^1$ to the storage server. The renewal of the ownership group is then carried out.

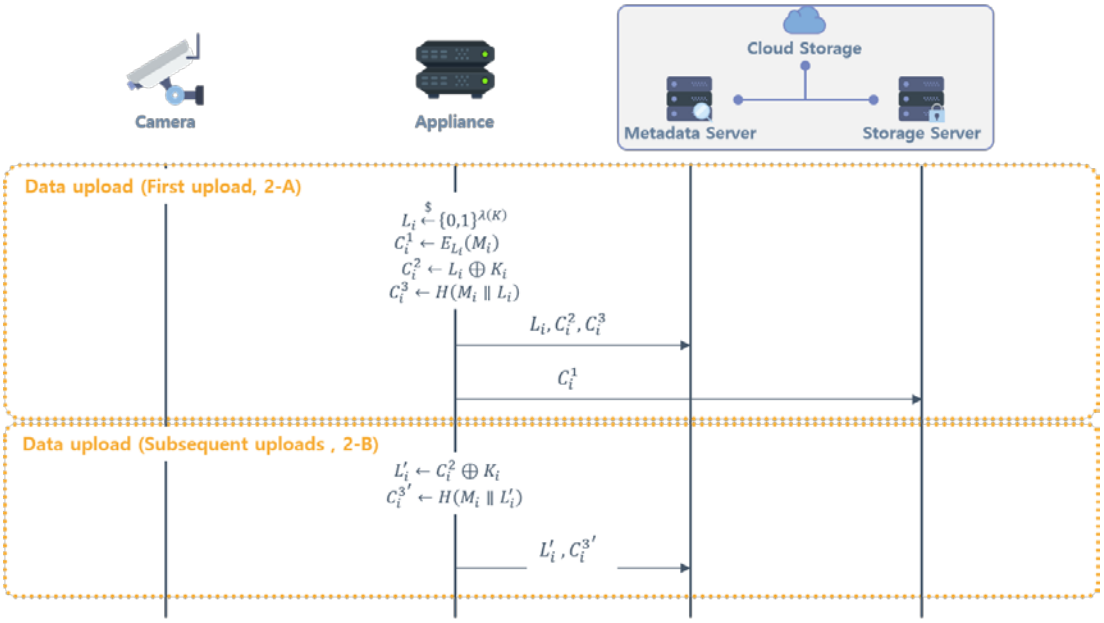


**Fig. 13.** Data upload step (first and subsequent upload)

### 4.4 Data Upload Phase (Subsequent Upload)

This step occurs if the data requested for upload already exists on the cloud storage server, as shown in **Fig. 13**. The metadata server determines that the data requested by the user already exists and notifies the user of the results. The appliance uses (9) to calculate the data $C_i^2$ obtained from the metadata server to obtain $L_i$. Then, it creates $C_i^{3'}$ using the $L_i$ obtained from (10).

$$L_i' \leftarrow C_i^2 \oplus K_i \tag{9}$$

$$C_i^{3'} \leftarrow H(chunk_i \| L_i') \tag{10}$$

The user sends $t_i, C_i^3, ID_u$ to the metadata server. In addition, ownership group renewals are performed on the server.

### 4.5 Data Upload Phase (When a Poison Attack Occurs)

This step is performed when you determine that the data you want to upload has already been exposed to Poison attacks, as shown in **Fig. 14**. The metadata server verifies that the appliance has the data that has been requested for uploading and sends the result and $C_i^2$. The appliance calculates data $C_i^2$ from the metadata server, as shown in (7), to obtain $L_i$. Then, it creates $C_i^{3'}$ using the acquired $L_i$, as shown in (8). The appliance sends the generated $\mathrm{t_i}$, $\mathrm{C_i^3}$, and $\mathrm{ID_u}$ to the metadata server. The metadata server identifies that the data uploaded by the appliance $C_i^{3'}$ is different from the $C_i^3$ previously held by the server, indicating that the data stored on the existing server was exposed to Poison attacks. The metadata server requests the appliance to upload data. The renewal of the ownership group is then executed. The appliance that receives an upload request from the metadata server generates $\mathrm{L_i''}$, $\mathrm{C_i^{1''}}$, $\mathrm{C_i^{2''}}$, and $\mathrm{C_i^{3''}}$.

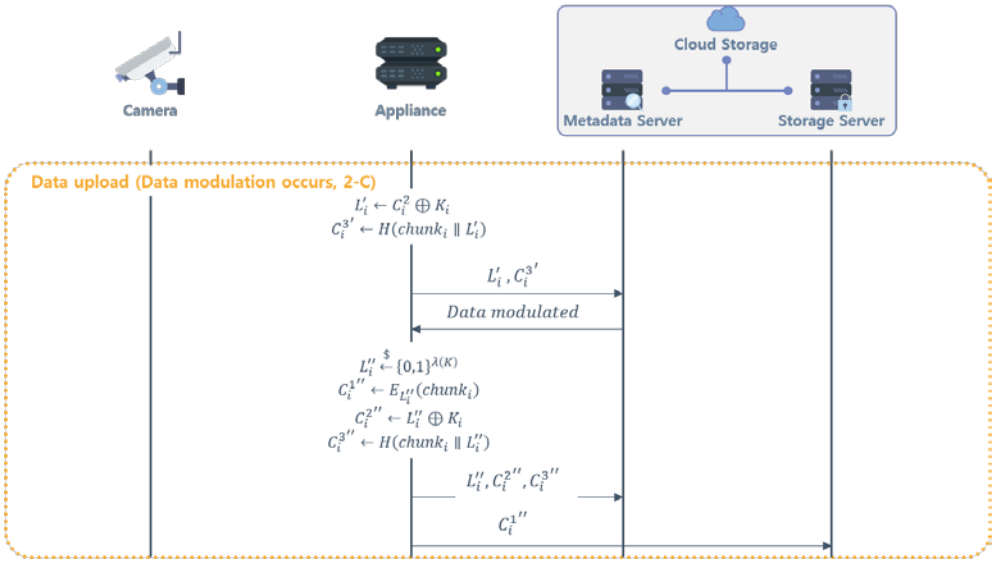


**Fig. 14.** Data upload step (when a poison attack occurs)

$$L_i'' \xleftarrow{\$} \{0,1\}^{\lambda(K)} \tag{11}$$

$$C_i^{1''} \leftarrow E_{L_i''}(chunk_i) \tag{12}$$

$$C_i^{2''} \leftarrow L_i'' \oplus K_i \tag{13}$$

$$C_i^{3''} \leftarrow H(chunk_i \| L_i'') \tag{14}$$

The user sends the generated $t_i$, $C_i^{2''}$, $C_i^{3''}$, and $ID_u$ to the metadata server and sends $C_i^{1''}$ to the storage server. Then, the ownership group is renewed.

### 4.6 Ownership Group Renewal Phase

This step is performed according to the results after the data upload phase is performed, and the ownership group contains the user whose ownership has been issued. At this phase, the metadata server encrypts the group key $GK_i$ using the added $KEK$. **Fig. 15, 16** show how the user tree and ownership group are constructed.

The owner group $G_i$ of the data $M_i$ has a key $KEK(G_i)$ to encrypt the group key $GK_i$. The $KEK(G_i)$ term is a list of the $KEK$s on the minimum node that can include all user nodes in $G_i$. For example, when $G_A = \{u_1, u_3, u_4, u_7, u_8\}$, $KEK(G_A) = \{KEK_1, KEK_{34}, KEK_{78}\}$ is derived. The process of updating a group is as follows.

After deduplication and uploads, the metadata server adds users to the ownership group $G_i$ of data $chunk_i$. Then, $KEK(G_i)$ is updated. Here, the added $KEK$ is assumed to be $KEK_2$.

The $KEK(G_i)$ consists of a $KEK$ that combines the added and the existing $KEK$ to cover all users in the $G_i$ group with a minimum of $KEK$.

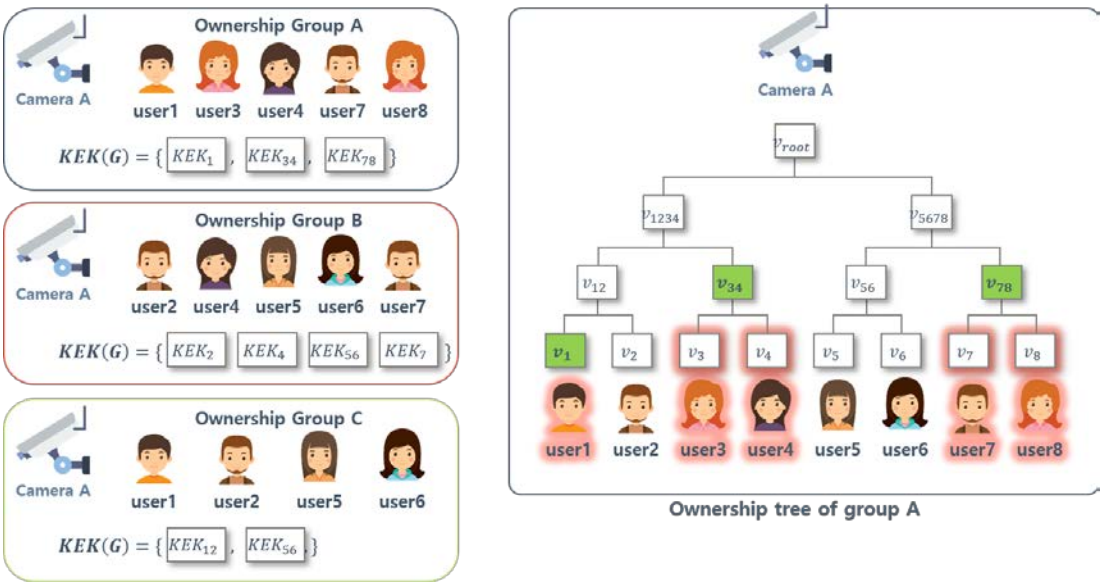


**Fig. 15.** Example of ownership tree in group A

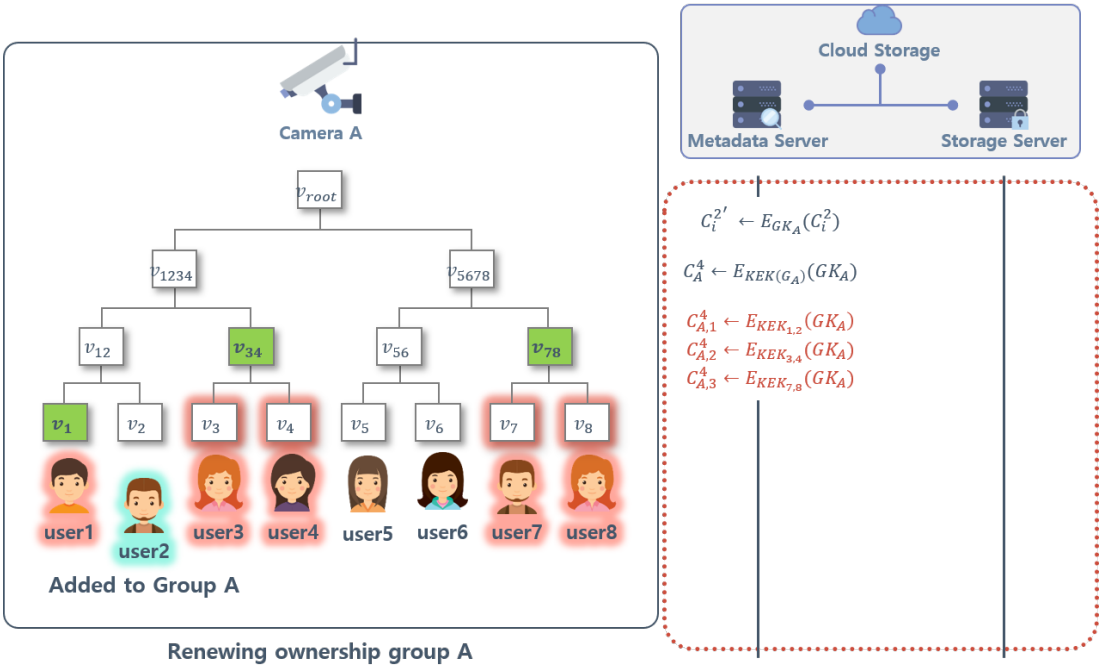


**Fig. 16.** Renewing ownership group

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

### 4.7 Data Download Phase

This step is performed when a user who owns ownership of the data $chunk_i$ requests a download of $chunk_i$, which is shown in **Fig. 17**.

The user sends the identifiers $t_i$ and validation data $C_i^3$ of the data to be downloaded to the meta data server.

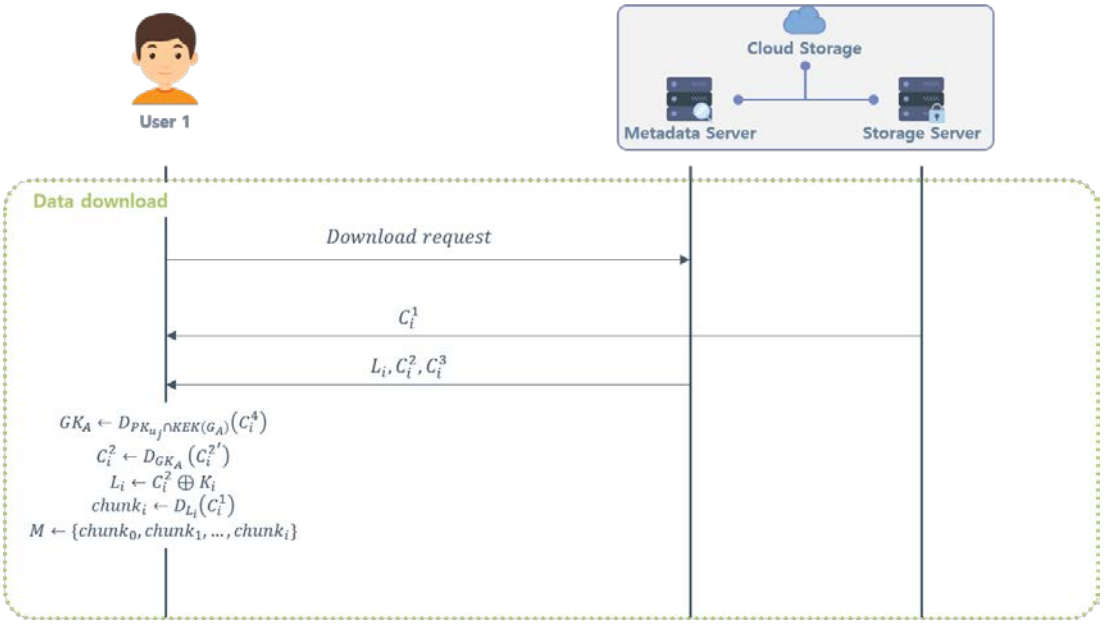


**Fig. 17.** Data download phase

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

The user obtains $chunk_i$ using $C_i^1, C_i^{2'}, C_i^4$, and the user's own $PK_{u_j}$ and $k_i$.

$$GK_A \leftarrow D_{PK_{u_j} \cap KEK(G_A)}(C_i^4) \tag{15}$$

$$C_i^2 \leftarrow D_{GK_A}(C_i^{2'}) \tag{16}$$

$$L_i \leftarrow C_i^2 \oplus K \tag{17}$$

$$chunk_i \leftarrow D_{L_i}(C_i^1) \tag{18}$$

The user compares $C_i^3$ and $C_i^{3'}$. This enables verification of data integrity.

$$C_i^3 \,?= H(chunk_i \parallel L_i) \tag{19}$$

## 5. Analysis of Proposed Scheme

In this paper, we propose a more efficient and secure method for deduplication of the encrypted data based on the dynamic ownership concept as in the scheme of Hur et al. [11]. Therefore, it displays characteristics similar to that of Hur et al., but differs in the following points.

**Table 1.** Comparison of proposed scheme

| | | | | Hur, et al. | Kim, et al. | Proposed Scheme |
|---|---|---|---|---|---|---|
| **Deduplication site** | | | | Server Side | Client Side | Client Side |
| **Dynamic ownership management** | | | | ○ | × | ○ |
| **Confidentiality** | | | | ○ | ○ | ○ |
| **Integrity** | | | | △ | ○ | ○ |
| **Anonymity** | | | | ○ | × | ○ |
| **Poison attack resistance** | | | | × | ○ | ○ |
| **Dictionary attack resistance** | | | | × | ○ | ○ |
| **Compu-tation** | **Upload** | **First upload** | **Uploader** | 2H + 1SE + 1⊕ | 3H + 1SE | 3H + 1SE + 1⊕ |
| | | **Subsequent upload** | **Uploader** | 2H + 1SE + 1⊕ | 3H + 1SE | 3H + 1⊕ |
| | | **Poison attack occurred** | **Uploader** | 2H + 1SE + 1⊕ | 3H + 1SE | 4H + 1SE + 1⊕ |
| | **Download** | | **Uploader** | 1H + 3SE + 1⊕ | 1H + 1SE | 1H + 3SE + 1⊕ |
| | **Ownership group renewal** | | **Server** | 1H + 3SE + 1⊕ | 1H + 1SE | 1H + 3SE + 1⊕ |

○: Offer; ×: Not offer; △: Partial offer;
**H:** Hash algorithm; **SE:** Symmetric key encryption; ⊕ : XOR Operation;

- **Confidentiality:** Cloud storage is always exposed to cyber attacks. Therefore, data encryption is required to keep data safe in the cloud. However, the deduplication process and the encryption process are the opposite. CE has been used to resolve these issues. CE is a technology that hashes data source chunks to obtain password key $K$, as shown in (20). As a result, user who own the same source always generate the same encryption key and ciphertext. This allows the system to deduplicate encrypted data using CE. The proposed technique uses CE-based MLE RCE mode to perform data encryption.

$$K \leftarrow H(chunk) \tag{20}$$

- **Integrity:** Once the data has been downloaded, the user can determine whether the downloaded data has been converted by checking the RCE tag. This process allows the user to perform hash operations using an encryption key $L_i$ that can obtain and verify data $chunk_i$, creating $C_i^3$, as shown in (19).

- **Anonymity:** This method is proposed by Hur et al. When a user requests a download of data, cloud storage sends the encrypted group key to the user. Users with legal person ownership can decrypt the key and verify that they have legal person ownership, while remaining anonymous. As a result, the user takes ownership of the data, and downloads and decodes the data through processes (13)-(16).

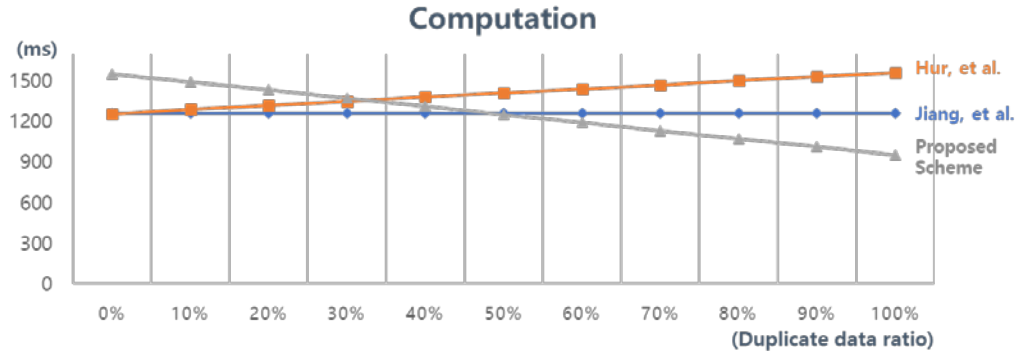


**Fig. 18.** Data download phase

- **Efficiency:** The proposed scheme is inversely proportional to the ratio of data redundancy and the amount of computation, as shown in Table 1. The proposed scheme is performed with fewer computations per block compared to the Hur et al's and Kim et al's Scheme. It performs the upload in a different way when the first upload and poison attack occur [9, 11]. However, the proposed technique omits the 3H + 1SE operation if duplicate data is present. As the percentage of redundant data increases, you can reduce the total number of jobs. The proposed approach uses client-side deduplication. This method lets you upload and deduplicate data while reducing the computational and transfer overhead compared to the server-side deduplication method when uploading duplicate data, as shown in **Fig. 18**.

- **Poison attack resistance:** In the Hur et al. scheme, if a Poison attack occurs, it is not possible to recover data loss from the Poison attack. Thus, the proposed technique detects poison attacks to prevent data loss during subsequent upload stages. If a poison attack occurs in a Hur et al scheme, the system can determine it, if the attack occurred during the download phase. However, the proposed technology detects poison attacks during subsequent uploads to prevent data loss because the data lost due to poison attacks cannot be recovered, as shown in **Table 1** [9].

- **Dictionary attack resistance:** CE is vulnerable to dictionary attacks. Generally, CE imports encryption keys from a data source, which enables an attacker to obtain the encryption key by guessing the data source. Therefore, data encryption key acquisition through data speculation must not be possible to ensure resistance to dictionary attacks. The proposed technique uses the RCE mode of CE-based MLE to perform data encryption. In the RCE mode, the keys obtained by hashing a data source are not used as data encryption keys, but as a key $K$ to encrypt data encryption key $L$. This method is shown in (21) - (23). Thus, even if an attacker speculates about a data source, the system can resist a dictionary attack because the attacker cannot obtain a data encryption key and a key to decrypt the data encryption key.

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

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

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

## 6. Conclusion

This study was conducted to store CCTV images more securely and efficiently in cloud storage environments. Cloud storage wastes space because it stores the same data in duplicate. Deduplication, which saves storage space by preventing duplicate storage of the same data, has been proposed as a method to address this issue. As a result, the uploaded data remains the same compared to the data stored on existing storage. However, data encryption should not be used to identify data sources because cloud storage is always at risk of a data leakage. As a result, cloud storage must encrypt and archive data with deduplication. CE has been proposed to perform data encryption and deduplication simultaneously. Because CE is a technology that hashes data sources to generate encryption keys, the same cryptograms are always generated when encrypting the same data source. As a result, the encrypted data can be deduplicated. CE, however, is at risk of preemptive attacks and client-side deduplication is at risk of poison attack. We have solved this problem using MCE's RCE mode. MLE is a CE-based encryption technology that helps prevent data source guesswork. The server can also detect poison attacks using inspection data generated in RCE mode. In addition, user ownership information is updated when deduplication is performed. Because cloud storage is uploaded with a variety of data and is used by many users, ownership information is updated frequently. This can make it difficult to disclose the identity of users when managing ownership and using such ownership information. To solve these problems, we applied ownership management technology using Merkle Tree based on techniques proposed by Hur et al. With this approach, data is safe from poison attacks, dictionary attacks, proof of ownership, ownership management, and anonymity The system can also apply client-side deduplication to reduce the number of calculations and traffic. The data redundancy rate also increases. As a result, the proposed technique is effective against a variety of security threats and the calculation decreases as the data redundancy increases.

## Acknowledgement

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 Soonchunhyang University Research Fund and.the Soonchunhyang University Research Fund.

## References

[1] K. W. Kim, Y. H. Joo and Y. I. Eom, “Deduplication Technologies over Encrypted Data,” in *Proc. of Symposium of the Korean Institute of Communications and Information Sciences*, vol. 33, no. 1, pp. 68–77, Feb. 2018. Article (CrossRef Link).

[2] N. Kaaniche and M. Laurent, “A Secure Client Side Deduplication Scheme in Cloud Storage Environments,” in *Proc. of 2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1-7, 2014. Article (CrossRef Link).

[3] P. Puzio, R. Molva, M. Önen and S. Loureiro, “PerfectDedup: Secure Data Deduplication,” *Lecture Notes in Computer Science, Springer International Publishing*, vol. 9481, pp. 150–166, 2016. Article (CrossRef Link).

[4] P. Puzio, R. Molva, M. Onen and S. Loureiro, "ClouDedup: Secure Deduplication with Encrypted Data for Cloud Storage," in *Proc. of 2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, 2013. Article (CrossRef Link).
[5] M. W. Storer, K. Greenan, D. D. E. Long and E. L. Miller, "Secure data deduplication," in *Proc. of the 4th ACM international workshop on Storage security and survivability - StorageSS '08*, pp. 1-10, 2008. Article (CrossRef Link).
[6] M. Bellare, S. Keelveedhi and T. Ristenpart, "Message-Locked Encryption and Secure Deduplication," in *Proc. of Advances in Cryptology – EUROCRYPT 2013, Springer Berlin Heidelberg*, pp. 296–312, 2013. Article (CrossRef Link).
[7] S. Halevi, D. Harnik, B. Pinkas and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *Proc. of the 18th ACM conference on Computer and communications security - CCS '11*, pp. 491-500, 2011. Article (CrossRef Link).
[8] J. R. Douceur, A. Adya, W. J. Bolosky, P. Simon and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system" in *Proc. of 22nd International Conference on Distributed Computing Systems*.
[9] K. Kim, T. -Y. Youn, N. -S. Jho and K. -Y. Chang, "Client-Side Deduplication to Enhance Security and Reduce Communication Costs," *ETRI Journal*, vol. 39, no. 1, pp. 116–123, Feb. 2017. Article (CrossRef Link).
[10] M. Bellare, S. Keelveedhi, T. Ristenpart, "DupLESS: Server-Aided Encryption for Deduplicated Storage," *IACR Cryptology ePrint Archive*, 429, 2013.
[11] J. Hur, D. Koo, Y. Shin and K. Kang, "Secure Data Deduplication with Dynamic Ownership Management in Cloud Storage," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 11, pp. 3113–3125, Nov. 2016. Article (CrossRef Link).
[12] M. Naor and O. Reingold, "Number-theoretic constructions of efficient pseudo-random functions," *Journal of the ACM*, vol. 51, no. 2, pp. 231–262, Mar. 2004. Article (CrossRef Link).
[13] D. Chaum, "Blind Signatures for Untraceable Payments," *Advances in Cryptology, Springer US*, pp. 199–203, 1983. Article (CrossRef Link).
[14] M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko, "The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme," *Journal of Cryptology*, vol. 16, no. 3, pp. 185–215, Jun. 2003. Article (CrossRef Link).

**Won-Bin Kim** received the M.S.degrees in Depart of Computer Science Engineering from Soonchunhyang University, Korea, in 2015, respectively. He is now a Ph.D. candidate in Department of Computer Science and Engineering from Soonchunhyang University, Korea. His research interests include Cloud Storage Security, Cryptography, Data Deduplication, Data Sharing, etc.

**Im-Yeong Lee** is corresponding author. He received the B.S. degrees in Department of Electronic Engineering from Hongik University, Korea, in 1981 and the M.S. and Ph.D. degrees in Department of Communication Engineering from Osaka University, Japan, in 1986 and 1989, respectively. From 1989 to 1994, he had been a senior researcher at ETRI (Electronics and Telecommunications Research Institute), Korea. Now he is a professor in Department of Computer Software Engineering from Soonchunhyang University, Korea. His research interests include Cryptography, Information theory, Computer & Network security.