Computers, Materials & Continua
DOI:10.32604/cmc.2022.019277
Article

Tech Science Press

Privacy-Enhanced Data Deduplication Computational Intelligence Technique for Secure Healthcare Applications

Jinsu Kim[1], Sungwook Ryu[2] and Namje Park[1,3,*]

[1]Department of Convergence Information Security, Graduate School, Jeju National University, 63294, Korea
[2]Master's Program in Future Strategy, Korea Advanced Institute of Science and Technology, 34141, Korea
[3]Department of Computer Education, Teachers College, Jeju National University, 63294, Korea
[*]Corresponding Author: Namje Park. Email: namjepark@jejunu.ac.kr
Received: 08 April 2021; Accepted: 10 May 2021

Abstract: A significant number of cloud storage environments are already implementing deduplication technology. Due to the nature of the cloud environment, a storage server capable of accommodating large-capacity storage is required. As storage capacity increases, additional storage solutions are required. By leveraging deduplication, you can fundamentally solve the cost problem. However, deduplication poses privacy concerns due to the structure itself. In this paper, we point out the privacy infringement problem and propose a new deduplication technique to solve it. In the proposed technique, since the user's map structure and files are not stored on the server, the file uploader list cannot be obtained through the server's meta-information analysis, so the user's privacy is maintained. In addition, the personal identification number (PIN) can be used to solve the file ownership problem and provides advantages such as safety against insider breaches and sniffing attacks. The proposed mechanism required an additional time of approximately 100 ms to add a ID_{Ref} to distinguish user-file during typical deduplication, and for smaller file sizes, the time required for additional operations is similar to the operation time, but relatively less time as the file's capacity grows.

Keywords: Computational intelligence; cloud; multimedia; data deduplication

1 Introduction

The recent surge in the popularity of cloud services has led to the emergence of a variety of cloud service providers (CSPs). There are many different types of services available on cloud platforms, most of which are storage-based solutions. The cloud storage market will continue to expand [1,2]. In the coming era of the Fourth Industrial Revolution (4IR), data size is expected to grow to astronomical scale, which will ultimately lead to increased costs for storage capacity, a huge burden on operators. Deduplication is therefore one of the key technologies that is critical to cloud environments. Deduplication is a technology that prevents multiple users from storing

the same data (redundancy). The advantage of this technology is that it can reduce data storage to unprecedented levels [3,4].

Currently, deduplication technology is already applied in many cloud storage environments. Due to the nature of cloud environments, large storage servers are required, and additional storage expansion is required as storage capacity increases. However, deduplication techniques can solve these cost problems at source. However, deduplication techniques have structurally privacy problems. For deduplication technology to be applied, the mapping structure of users and files is stored as meta-information, and the list of users who uploaded specific files can be obtained through meta-analysis on the server. In particular, the problem can be very serious if files have sensitivity, such as political tendencies, ideas, certain diseases, and sex life. If the upload user list of the file is available on the cloud server, the risk that the cloud environment will act as a kind of monitoring system cannot be ruled out [5–7].

In this paper, we point out these problems, and propose new deduplication techniques to address them. Since the proposed technique does not store the mapping structure of users and files on the server, meta-information analysis on the server cannot secure a list of users who upload files, protecting users' privacy. In addition, Personal Identification Number (PIN) can be utilized to address ownership issues of files, and it has the advantage of being safe from insider attacks and sniffing attacks.

2 Related Studies

2.1 Data Deduplication Technology

Deduplication is a technology that can store identical data in a single file without duplicating it. Typically, deduplication can be achieved by comparing the files' hash values and is practiced commonly through mapping for users who have access to a certain redundant file(s). Fig. 1 shows an example where deduplication is implemented for real-world applications. If user A and user B have the same file in the application, the cloud server stores the information in a single file with the map structure defined for both users to use at the same time.

During this process, the file removal for user B is completed by removing the mapping information as shown in Fig. 2. When User A later removes the file, it removes all of User A's mapping information and files, resulting in deduplication.

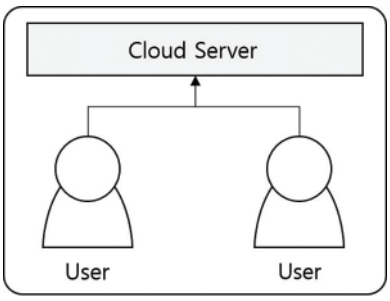

Figure 1: Deduplication technique

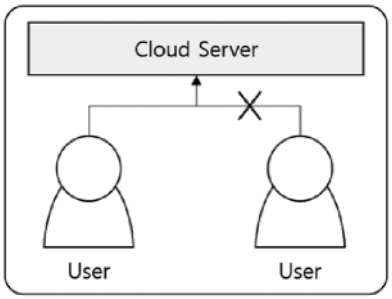

Figure 2: Delete files from deduplication

2.2 Security Technologies for Deduplication

2.2.1 Convergent Encryption

In general, deduplication is applied in a way that has vulnerabilities in terms of security. In other words, when the server alone retains a file(s) belonging to multiple users, the information can be exposed to various security threats due to structural reasons. Hence, the need for appropriate security measures for the deduplication technique.

The paper by Douceur proposed a convergent encryption method to implement deduplication [1]. This method uses the hash of the original plaintext to obtain encrypted keys. For instance, if the original plaintext is M, its encrypted key can be created by applying the cryptographic hash to M, thus resulting in H(M). Basically, convergent encryption is capable of creating different keys for a single file. Unless the content of the original plaintext is known, it is difficult to figure out or crack the encrypted key and thus the files are managed safely without the need for separate key management. This merit notwithstanding, convergent encryption has a problem in that it basically is vulnerable to brute-force attacks when the data has low or small entropy [4].

2.2.2 DupLESS

Bellare proposed in their research paper a technique to thwart brute-force attacks [5]. In DupLESS, a separate key server is provided to counteract the susceptibility found in the convergent encryption approach. It is a deduplication method on the server's side, where the server in question creates encrypted keys by interacting with the client. However, both convergent encryption and DupLESS have limitations in that the focus is only placed on creating encrypted keys for the deduplicated files.

The security limitations of deduplication techniques cannot be resolved simply by encrypting the files using an appropriate key-management approach. Of note, user privacy protection and deduplication have certain elements in conflict with each other. Chapter 3 further discusses threats of user privacy breaches resulting from using deduplication techniques and other security issues.

3 Privacy Issues with Deduplication Techniques

3.1 Problems with User-to-file Structure

Fundamentally, in order to apply the deduplication technique to cloud storage, as mentioned in Chapter 2, a mapping structure between users and files must be established. However, the user-file mapping structure problem has a serious privacy problem. When looking at the simple mapping structure alone, it may not be easy to understand what kind of privacy problem may

occur. However, if the file is highly sensitive to the user, the user-file mapping structure itself may threaten the user's privacy.

In the case of Korea, According to Paragraph 1 of Article 23 for korea (Restrictions on the Processing of Sensitive Information) of the Personal Information Protection Act, the personal information controller is responsible for information on ideology and belief, union or political party membership or withdrawal, political opinion, health, sexual life, etc., and other information subjects. Sensitive information is defined as information that is likely to significantly infringe on a person's privacy. In addition, according to Paragraph 2 of the same Article, it is stipulated those necessary measures should be taken to ensure the safety of sensitive information.

Strong security measures are required if the upload target file is a file containing sensitive information or if the upload history is exposed in a sensitive situation. In particular, in the case of uploading documents supporting a specific political party, ideological content, specific diseases, and sex-related files, the mapping structure between the user and the file itself may constitute an invasion of privacy when deduplication is processed. In particular, in the deduplication environment, multiple users and a single file are processed in a mapped structure, which means that it is possible to secure a list of users who uploaded a specific file on the server side. It cannot be ruled out that the cloud server even performs a kind of monitoring role.

Whether or not a particular user has uploaded a file is his or her own privacy zone. Even if you are a cloud server administrator, securing all the list of files uploaded by a specific person or, conversely, securing the list of uploaders who uploaded a specific file can be considered an act of actively violating privacy.

3.2 Incompatibility of Deduplication and Securing Confidentiality

Typically, cloud systems provide data confidentiality services based on encryption on the server side. In other words, when a plaintext file is received from the client side, it is common for the server side to prepare for hacking by storing it in storage after encryption processing. In such cases, when the user requests a download, the server side sends a decrypted plaintext file to the user. This means that the server has an encryption key and can be decrypted if necessary. Therefore, if an administrator has a high level of privileges on the cloud server side, the possibility of restoring files based on that key may exist if necessary [6–8].

As a strong security measure to protect users' privacy, the client side can consider uploading file encryption to the server after it has been pre-applied. In this case, the file can be managed safely on the server side, but the deduplication mechanism does not work. As shown in Fig. 3, although User A and User B uploaded the same file, they were encrypted with different keys, effectively recognized by the server as different files, making deduplication impossible. In other words, achieving full confidentiality through pre-encryption on the client side and capacity savings through deduplication at the same time is practically very challenging.

3.3 File Ownership Issue

File ownership and sharing issues are related to security policies from the server's point of view. If user A and user B upload the same file in a cloud environment to which deduplication is applied, the actual saved file is one. In this case, you need to consider who owns the file. Conversely, the user requesting to download a file needs to prove that he or she owns the file.

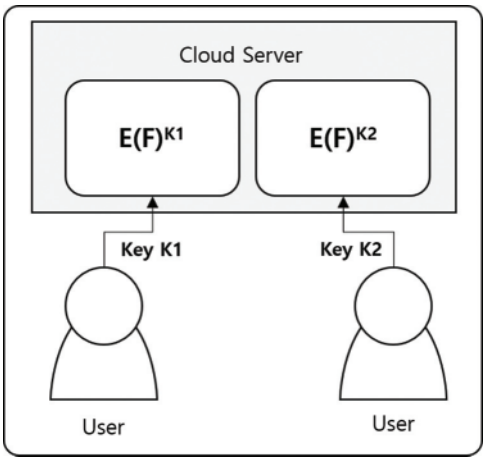

Figure 3: Client-side encryption

This problem can be solved by strictly managing the list of who has previously uploaded a specific file. However, this method has the privacy problem according to the user-file mapping structure mentioned above as it is. When a particular user takes ownership of a particular file, conversely, it means that the file was previously uploaded by that user, which in other words makes it clear to the server who uploaded the particular file.

This means that the list of users who uploaded various sensitive information such as promotional materials supporting a specific party, data related to a specific disease and sex life, and data containing specific ideas and beliefs can be known as it is through meta-analysis on the server side. Therefore, the cloud environment to which the existing deduplication method is applied has a structural privacy problem. Due to this ownership problem, cloud services with deduplication applied to date do not provide a complete privacy protection method from a strict point of view [9–11].

3.4 Insider Attack Vulnerability

In the cloud environment, there is fundamentally a risk of personal information exposure by insiders. Since there have been many cases of data exposure by insiders in the past, insider attacks are actually considered a very large security threat in the cloud environment.

Typical cloud environments provide minimal security through data encryption. However, an internal administrator with the highest level of privileges cannot rule out the assumption that a file decryption key can be collected, or that a file and a user mapping relationship can be collected through metadata analysis, even if the encryption key is unknown. This means that a list of users who have posted a particular file can be obtained, or vice versa, a list of files posted by a particular user can be obtained through metadata analysis [12].

Solving such deduplication and privacy problems simultaneously is practically very difficult. In this paper, we propose ways to solve both deduplication and privacy problems simultaneously.

4 Deduplication Technique to Solve the Proposed Privacy Problem

In this chapter, we propose a new technique to solve the privacy problem in the existing deduplication method.

4.1 Overview of Proposed Deduplication Techniques

Privacy issues in the existing deduplication environment fundamentally arise in the structure in which users and files are mapped. In fact, if the mapping structure between files and users is completely removed, it is impossible to obtain a list of users who have uploaded a corresponding file from a specific file, or vice versa, a list of files uploaded by a specific user. Therefore, in this paper, we propose a method in which users cannot be inferred only through metadata and file structure analysis on the server by fundamentally blocking the mapping relationship between the user-file list and the file [13,14].

The proposed method allows users to upload/download files based on PIN, ID_{Ref}, and F_{has}. Here, the PIN is a value that is known only to the user, and the ID_{Ref} is a value owned by the meta-server and is configured based on the PIN. Furthermore, the F_{has} value is a hash-taken value of the file, and the ID_{Ref} cannot be guessed based on the F_{has}, and vice versa, the F_{has} and PIN cannot be guessed based on the ID_{Ref}. This means that no user-file mapping information can be obtained only through ID_{Ref} and F_{has} held by meta-server and storage servers, and extracting F_{has} based on ID_{Ref} necessarily requires a PIN value that users know. In the proposed method, each element is independently recorded and used in the process of mapping users and files to prevent the other elements from being exposed by one element, thereby solving the privacy problem. However, as processing for each element is required, it requires somewhat more time than the conventional method. Fig. 4 shows the overall appearance of the proposed schema.

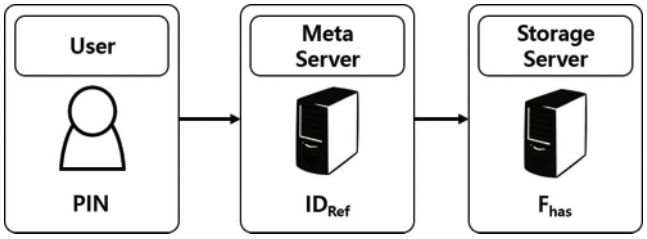

Figure 4: Overview of the proposed scheme

Tab. 1 shows the terms used in the proposal mechanism.

4.2 System Architecture

Fig. 5 shows the configuration of the client and the cloud server. Clients can connect to cloud servers from a variety of devices such as mobile devices, laptops, and PCs. On the other hand, cloud servers have metasurvers and storage servers. Metasurvers and storage servers must be strictly partitioned.

This means that it cannot be configured together on one server, but must be configured on different servers. It is also recommended that meta-server and storage servers be split physically. Here, the administrator responsible for the meta-server and storage server must be configured separately.

4.3 File Duplicate Verification Protocol

File redundancy verification protocol is a step in determining whether a file actually exists on the server. If the same file exists on the server, it is efficient because it does not require a file upload process. Fig. 6 shows the flow of the file redundancy verification protocol.

Table 1: Notation

Abbreviation	Description
ID_U	User ID
ID_{Se}	Session ID
EK_{Pre}	Shared encryption key
F	File upload and download
F_{ha}	Original file's hash value
F_{has}	F_{ha}'s Hash value
F_{Pa}	Full path the destination file
PIN	User ownership identification key
ID_{Ref}	Reference value recorded in the meta server
H( )K	Hash result of specific value
E( )K	Encryption result using key value k
D( )K	Decryption result using key value k

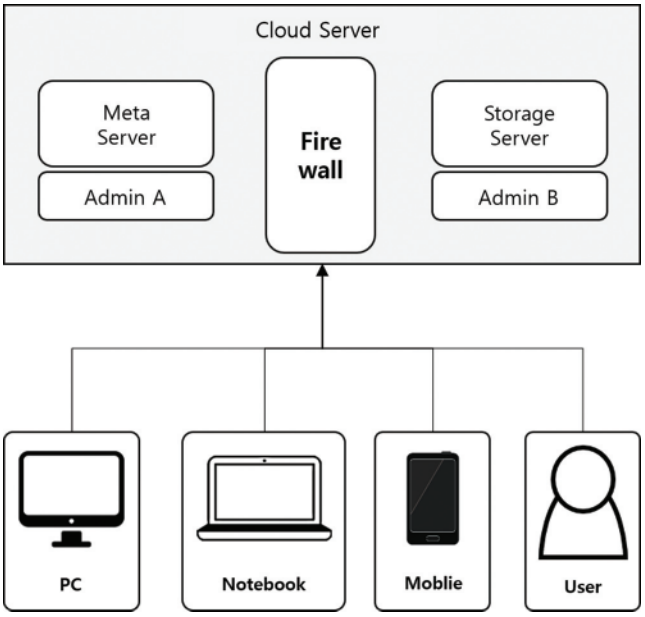

Figure 5: System configuration

① Upon client request, the meta server generates the session ID ID_{Se}.
② Meta-server encrypts the ID_{Se}'s with a pre-shared secret key EK_{Pre} and sends it to the client.
③ The client obtains the ID_{Se}'s by decrypting the ciphertext.
④ The client requests the server to upload the file it wants to record to a specific path.
⑤ Meta server generates the file path requested by the client.
⑥ Metaserver generates routes and informs clients of the results.
⑦ The client hashes the upload destination file to extract the hash value of the file.

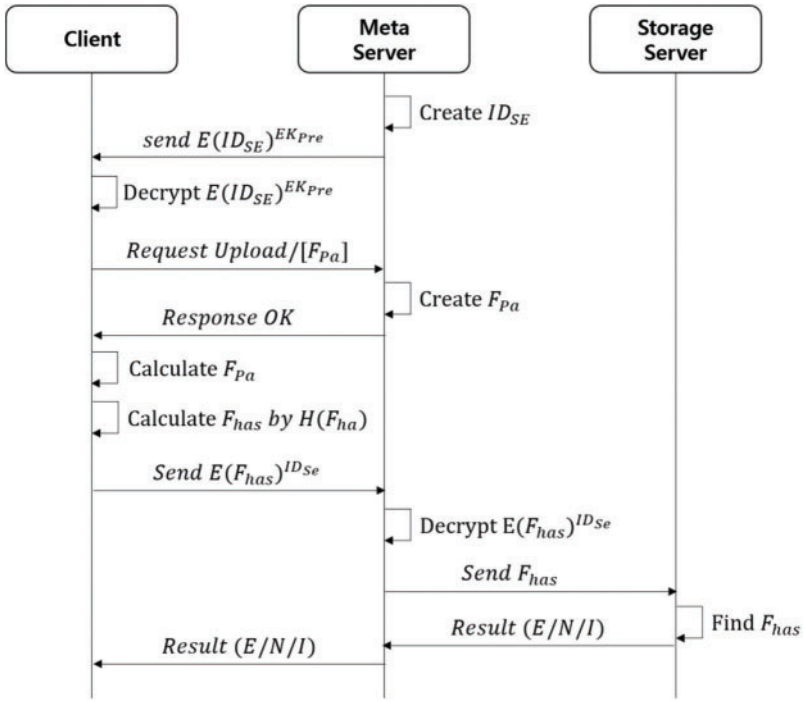

Figure 6: File duplicate check protocol

⑧ The client generates an F_{has} by taking another hash processing on the hash value of the extracted file.
⑨ Clients encrypt and transmit F_{has}.
⑩ The meta-server obtains the F_{has} by decrypting the ciphertext sent from the client.
⑪ The meta-server sends an F_{has} to the storage server to inquire whether the file already exists.
⑫ The storage server checks whether the file is registered or not.
⑬ The result shall be returned by dividing it into one of E, N, and I. where E stands for Exist, which means that the file already exists, and N stands for Not Exist, which means that the file does not exist. Also, I stands for Incomplete, meaning that only a few files have been uploaded.
⑭ Meta-server responds to clients with E/N/I values.

The results of the duplicate file check are as shown in Tab. 2. If you receive E, the same file is already uploaded on the server, so no additional upload is required. Therefore, the client only needs to perform the mapping process without any upload process. However, if an N or I is received, it means that an upload is required on the server, so the client must perform a file upload.

If the result is I, then only partial file uploads are done, so the rest of the uploads except the files uploaded on the server are carried out

4.4 File Upload and Duplicate Mapping Protocol

Fig. 7 shows the flow of file uploads and redundant mapping protocols.

Table 2: Notation

Result	Description
E	If the file already exists on the server
N	If there are no files on the server
I	If the file is partially written to the server

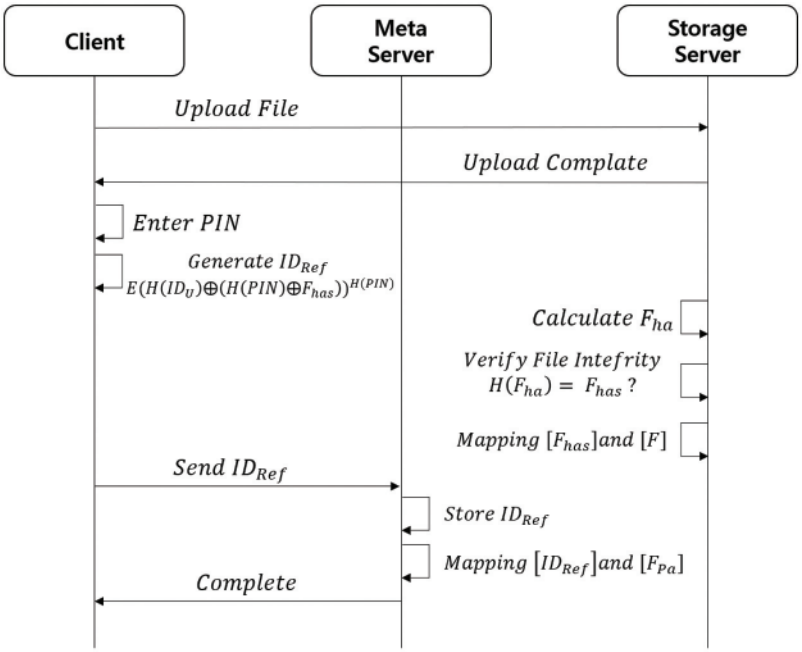

Figure 7: File upload and duplicate mapping protocol

① Steps ① and ② are only applicable if the results of the duplicate file check are N and I. In other words, if the result of a duplicate file is E, there is no need for a separate upload process. However, if the values of N and I are received, the actual file upload is performed to the storage server.
② Inform the client that the upload is complete.
③ The user checks the upload completion and enters the PIN through the client.
④ The client generates a RefID based on a PIN. ID_{Ref} are generated by the following expressions:

$$RefID = E(H(UserID) \oplus (H(PIN) \oplus FHV))^{H(PIN)} \tag{1}$$

⑤ Storage servers extract hash values for verifying file integrity.
⑥ The storage server verifies the integrity of the received file by comparing it with the hash value of the file and the F_{has} value.
⑦ Map the F_{has} and the uploaded file and save it.
⑧ The client sends the ID_{Ref} generated by Step ④ to the meta-server.
⑨ The meta-server stores the ID_{Ref} received by the client.

⑩ Meta server maps the ID_{Ref} and F_{Pa} values for deduplication of files.
⑪ The meta-server notifies the client that the upload is complete.

4.5 Deduplication File Download Protocol

Fig. 8 shows the flow of the deduplication file download protocol.

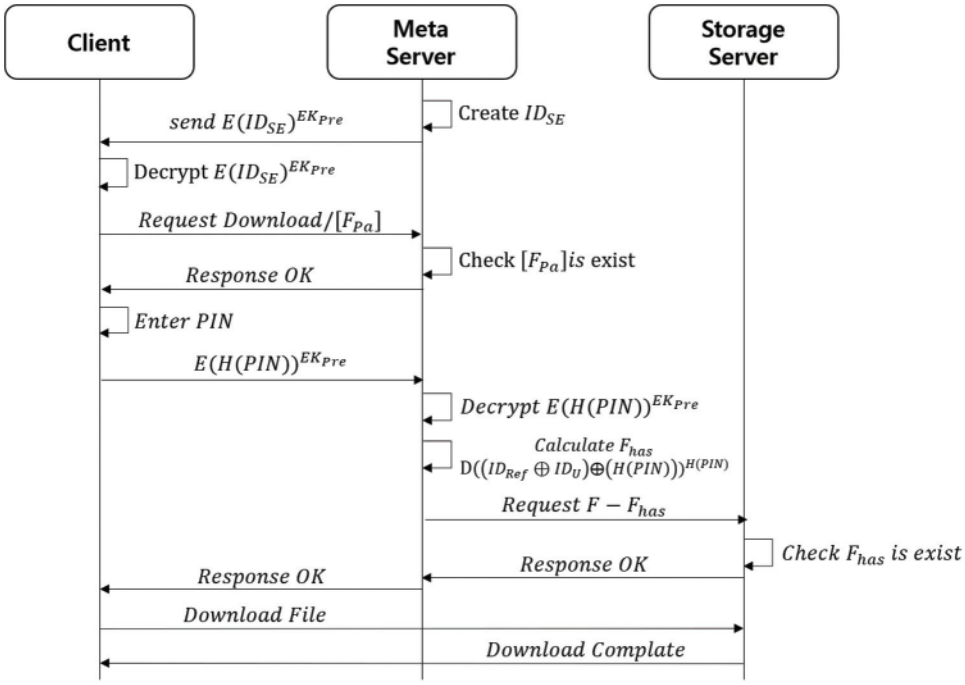

Figure 8: File download protocol

① Upon client request, the meta server generates the session ID ID_{Se}.
② The meta server encrypts the generated ID_{Se}'s with the private key EK_{Pre} and sends it to the client server.
③ The client receives a passphrase from the meta-server and obtains the ID_{Se}'s through decryption.
④ The client uses the ID_{Se} to request a file download to the meta-server.
⑤ The meta-server first checks if the file exists in the requested path. If it does not exist, terminate it.
⑥ The meta-server informs the client that the file exists in that path.
⑦ The client receives a PIN from the user.
⑧ The client encrypts the hash value of the PIN entered by the user with the ID_{Se} key and sends it to the meta-server.
⑨ The meta-server decrypts the ciphertext to obtain the hash value of the PIN, H(PIN).
⑩ The meta-server extracts the F_{has} from the following expressions based on the ID_{Ref} and H(PIN) values.

$$D((RefID \oplus UserID) \oplus (H(PIN)))^{H(PIN)} \tag{2}$$

⑪ Meta servers deliver F_{has} to storage servers.

⑫ The storage server determines whether a specific file mapped to the F_{has} exists.
⑬ The storage server responds to the meta-server that there is a file mapped to the corresponding F_{has}.
⑭ The meta-server informs the client that it is ready for download.
⑬ The client requests files from the storage server.
⑬ The client transfers the file from the storage server to complete it.

5 Security and Efficiency Analysis

5.1 Privacy Protection Issue

The existing deduplication mechanism ensures the security of files by implementing the file-encryption technique, but lays the user-to-file mapping structure bare. Such vulnerability renders metainformation analysis feasible, potentially leading to serious user-privacy breaches.

The technique proposed by this paper separates the user and file information completely. ID_{Ref} is stored on the meta server and the F_{has} value on the storage server.

ID_{Ref} refers to the value that is subjected to PIN-based XOR operation and encryption. The so encrypted value cannot be decoded even by the administrator. For decoding the H(PIN) value is needed, which, however, is not stored on the server [15–18].

Using ID_{Ref}, therefore, the F_{has} cannot be computed and conversely, ID_{Ref} cannot be tracked based on F_{has}. If the file in question is to be downloaded using all of the information needed, the PIN information known to the user must be available. Of note, ID_{Ref} is encrypted using the H(PIN) value as the key and thus makes the analysis more difficult. That is, without knowing the PIN value, the information on the server cannot be used to reconstruct the mapping relationship between user information and files, thereby ensuring that users' privacy is secure.

5.2 Resolution of Ownership Issue

The conventional deduplication technology typically handles the file ownership issue using the user mapping structure of files. However, the existing deduplication method has a problem that it can be configured in a manner that may invade privacy. In the deduplication system proposed herein, the PIN information serves as a piece of evidence that substantiates a particular file-ownership of users. When an authorized individual attempts to download the file, the F_{has} value cannot be computed normally from ID_{Ref} because he/she does not know the PIN; hence file downloading infeasible in the usual way. Contrastingly, any authorized user can compute the F_{has} value from ID_{Ref} based on the PIN information that was used for the upload and accordingly executes the download. In the latter case, computing the file F_{has} from ID_{Ref} can be done only by the individual(s) who knows the PIN, and thus the user concerned can prove his/her ownership of the file. In other words, the confidential information is PIN, and the file ownership issue can be resolved while user privacy is being ensured when one implements the characteristic that confidential information is not stored on the server.

5.3 Insider Attacks

The existing deduplication technique keeps the file-to-user mapping structure as metainformation. In that mechanism, the administrator with viewing rights can collect the metainformation, which makes the system extremely susceptible to insider attacks. In other words, the cloud server administrator/manager can easily access the information including the uploader lists of certain files and conversely the file lists of certain users.

In the proposed approach, even if an insider attack has exposed the entirety of the meta server and storage server, the hacker still cannot obtain the file lists of users or the uploader lists of certain files. The server-stored information is ID_{Ref} and F_{has}, and these two sets of values are not related to each other. The ID_{Ref} information, in particular, is encrypted in H(PIN) keys, hence it makes meta-analysis even more challenging. Even if the data were exposed by an insider-originated brute-force attack, the relationships between files and users cannot be established without knowing the PIN, hence the insider-attack problem can be solved.

5.4 Sniffing Attacks

In general, sniffing attacks are aimed at intercepting packets that occur in the process of communication between a server and a client. The proposed technique encrypts the comparison parameters of the client and server in the protocol execution process before sending out the data. This step guarantees security even if a hacker launches sniffing attacks. In particular, since the decryption process of the encrypted comparison parameter is performed at the receiving client or server, the possibility of eavesdropping by a hacker can be reduced. Of note, the technique utilizes ID_{Se}, i.e., a single-session ID randomly generated by the meta server, as the key to encrypt the parameters that are transmitted during the protocol application. Therefore, even if hackers perform replaying following their sniffing attacks, they cannot execute the normal up/download protocol for the file unless knowing the EK_{Pre} previously shared between the server and client [19–23].

5.5 Integrity Considerations

The proposed deduplication technique employs yet another verification process on the server's side so as to ensure the integrity of files. In this mechanism, the F_{has} value refers to the rehashed value of the file hash value, and the server can verify the integrity of the uploaded file by comparing the two hash values of the uploaded file and the F_{has} value to find a match. In other words, the F_{has} value is characterized by its trait that it is utilized for substantiating file ownership during download protocol execution and at the same time can play the role of integrity verifier for the uploaded file. Additionally, a notable merit is that the alteration status of storage server-stored files can be monitored.

5.6 Efficiency Considerations

Deduplication techniques typically generate file-hash values based on Merkle trees. Generating hash values for files virtually takes up the largest amount of time during the process of deduplication and thus this study compared the time required for generating Merkle trees during file redundancy check against the combined time spent on computing the ID_{Ref} in the proposed algorithms. Tab. 3 below presents the environments for performance measurement.

Fig. 9 and Tab. 4 show the results of measuring the performance of the proposed mechanism.

In the proposed technique, enhanced security functions are incorporated into the conventional deduplication method. As such, the actual processing time in the new mechanism requires some additional time to be spent on security implementations (e.g., ID_{Ref} operation) compared to the conventional deduplication approach. This time requirement notwithstanding, the new technique shows, in practical terms, no significant drop in performance against the conventional Merkle tree-based deduplication method, and yet offers merits that include greater privacy protection and security. Hence, the proposed mechanism should likely replace the current deduplication approach as a way to strengthen its security performance.

Table 3: Test environment

O/S	Windows 7
CPU	Intel i5-3470 3.2 GHz
RAM	4 GB
Language	C++
Hash Algorithm	SHA-1

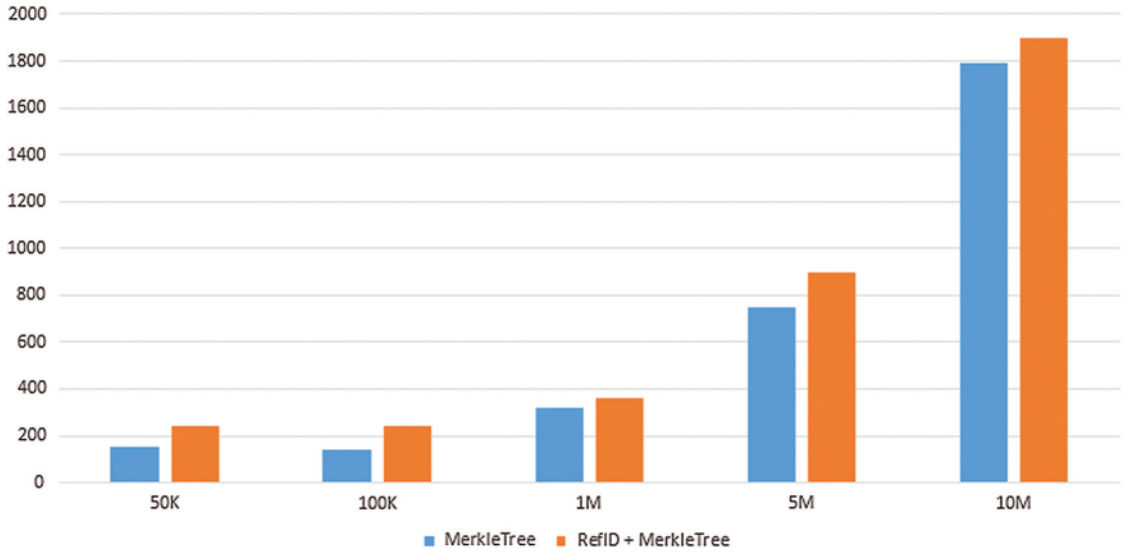

Figure 9: Performance measure result

Table 4: Comparison of performance measurement results

Method	50K	100K	1M	5M	10M
Merkle Tree	152	141	315	747	1786
Merkle Tree + ID_{Ref}	239	243	357	899	1907

6 Conclusion

As the network infrastructure evolves, the utilization of cloud platforms is increasing. In particular, as cloud services are not only used in limited fields but are used throughout society, the market for providing services is also becoming huge. Cloud services will expand from 4IR to convergence with a variety of technologies in the future, as they do not put much pressure on users compared to convenience. However, the use of cloud services by various users means that astronomical data is collected at the same time. Increasing data leads to higher costs required to provide services, for which redundant data removal technology can be a great solution [24–26]. In the construction of cloud storage systems, deduplication technology is positioned as an essential element technology. As a technology that can dramatically reduce storage capacity, deduplication technology has the advantage of direct cost reduction and will continue to serve as a core technology for the cloud in the future. However, the deduplication technology structurally has a

problem of infringing on the user's privacy, and the existing deduplication security technology has focused only on the file encryption itself [27–29].

Existing deduplication applications are experiencing privacy problems due to user-file mapping structures. Thus, tracking to users can be prevented by breaking the user-file mapping structure. The mechanism proposed in this paper uses three main elements: the user's unique secret key PIN, ID_{Ref} referenced based on the PIN stored on the meta-server, and F_{has}, hash values for the hash values of the file. In this paper, we pointed out the privacy problem according to the user-file mapping structure, and proposed a new technique to supplement it. The proposed method stores only the ID_{Ref} and F_{has} values in the meta server and the storage server, respectively, and has the characteristic that the two values do not have any connection to each other, so privacy can be safely protected and the meta information exposure caused by insider attacks can also be secured. In addition, by utilizing the user's PIN value, the file ownership and privacy issues were solved at the same time.

To describe the proposed method, Chapter 2 of the paper reviewed previous studies relating to the conventional deduplication technology. Chapter 3 indicated the breach of privacy resulting from implementing the current technology as well as other possible security problems. In Chapter 4, a new deduplication approach was proposed, followed by security performance analysis for the new method in Chapter 5.

Deduplication technology is an essential technology in building cloud storage, and deduplication technology is currently applied to many cloud services. In order to provide secure cloud service in the future era of the 4th industrial era, research on privacy issues that may occur in deduplication technology will be continuously required.

Funding Statement: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1I1A3A01062789) (received by N. Park).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] 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. Int. Conf. on Distributed Computing Systems*, Vienna, Austria, pp. 1–14, 2002.

[2] N. Park and N. Kang, "Mutual authentication scheme in secure internet of things technology for comfortable lifestyle," *Sensors*, vol. 16, no. 1, pp. 20, 2016.

[3] D. Lee and N. Park, "Geocasting-based synchronization of Almanac on the maritime cloud for distributed smart surveillance," *Journal of Supercomputing*, vol. 73, no. 3, pp. 1103–1118, 2017.

[4] K. Park, J. E. Eom, J. Park and D. H. Lee, "Secure and efficient client-side deduplication for cloud storage," *Journal of the Korea Institute of Information Security and Cryptology*, vol. 25, no. 1, pp. 83–94, 2015.

[5] B. Mihir, S. Keelveedhi and T. Ristenpart, "DupLESS: Server-aided encryption for deduplicated storage," in *Proc. USENIX*, Berkeley, USA, pp. 179–194, 2013.

[6] D. Lee and N. Park, "Teaching book and tools of elementary network security learning using gamification mechanism," *Journal of the Korea Institute of Information Security and Cryptology*, vol. 26, no. 3, pp. 787–797, 2016.

[7] N. Park, J. Kwak, S. Kim, D. Won and H. Kim, "WIPI mobile platform with secure service for mobile RFID network environment," *Advanced Web and Network Technologies, and Applications, LNCS*, vol. 3842, pp. 741–748, 2006.
[8] H. Kim, C. Park, D. Hong and C. Seo, "Encrypted data deduplication using key issuing server," *Korea Information Science Society*, vol. 43, no. 2, pp. 143–151, 2016.
[9] C. Park, D. Hong, C. Seo and K. Y. Chang, "Privacy preserving source-based deduplication in cloud storage," *Korea Institute of Information Security and Cryptology*, vol. 25, no. 1, pp. 123–132, 2015.
[10] N. Park and M. Kim, "Implementation of load management application system using smart grid privacy policy in energy management service environment," *Cluster Computing*, vol. 17, no. 3, pp. 653–664, 2014.
[11] J. Li, X. Chen, M. Li, J. Li, P. P. Lee *et al.,* "Secure deduplication with efficient and reliable convergent key management," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1615–1625, 2014.
[12] N. Park and H. C. Bang, "Mobile middleware platform for secure vessel traffic system in IoT service environment," *Security and Communication Networks*, vol. 9, no. 6, pp. 500–512, 2016.
[13] N. Park, H. Hu and Q. Jin, "Security and privacy mechanisms for sensor middleware and application in internet of things (IoT)," *International Journal of Distributed Sensor Networks*, vol. 12, no. 1, pp. 2965438, 2016.
[14] C. Park, D. Hong and C. Seo, "A secure and practical encrypted data de-duplication with proof of ownership in cloud storage," *Korea Information Science Society*, vol. 43, no. 10, pp. 1165–1172, 2016.
[15] N. Park, "UHF/HF dual-band integrated mobile RFID/NFC linkage method for mobile device-based business application," *Journal of the Korean Institute of Communication Sciences*, vol. 38, no. 10, pp. 841–851, 2013.
[16] N. Park, "Design and implementation of mobile VTS middleware for efficient IVEF service," *Journal of Korean Institute of Communications and Information Sciences*, vol. 39C, no. 6, pp. 466–475, 2014.
[17] P. Puzio, R. Molva, M. Onen and S. Loureiro, "ClouDedup: Secure deduplication with encrypted data for cloud storage," in *2013 IEEE 5th Int. Conf. on Cloud Computing Technology and Science*, Bristol, UK, vol. 1, pp. 363–370, 2013.
[18] N. Park, J. Park and H. Kim, "Inter-authentication and session key sharing procedure for secure M2M/IoT environment," *International Information Institute (Tokyo) Information*, vol. 18, no. 1, pp. 261–266, 2015.
[19] D. Lee and N. Park, "A secure almanac synchronization method for open IoT maritime cloud environment," *Journal of Korean Institute of Information Technology*, vol. 15, no. 2, pp. 79–90, 2017.
[20] Y. J. Yoo, S. J. Kim and Y. W. Ko, "Cloud file synchronization scheme using bidirectional data deduplication," *Journal of Korean Institute of Information Technology*, vol. 12, no. 1, pp. 103–110, 2014.
[21] B. K. Kim, Y. W. Ko and K. M. Lee, "Performance enhancement for data deduplication server using bloom filter," *Journal of Korean Institute of Information Technology*, vol. 12, no. 4, pp. 129–136, 2014.
[22] D. Lee and N. Park, "Electronic identity information hiding methods using a secret sharing scheme in multimedia-centric internet of things environment," *Personal and Ubiquitous Computing*, vol. 22, no. 1, pp. 3–10, 2018.
[23] D. Lee, N. Park, G. Kim and S. Jin, "De-identification of metering data for smart grid personal security in intelligent CCTV-based P2P cloud computing environment," *Peer-to-Peer Networking and Applications*, vol. 11, no. 6, pp. 1299–1308, 2018.
[24] N. Park, "The core competencies of SEL-based innovative creativity education," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 19, pp. 837–849, 2018.
[25] N. Park, "STEAM education program: Training program for financial engineering career," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 19, pp. 819–835, 2018.
[26] M. Masud, G. S. Gaba, S. Alqahtani, G. Muhammad, B. B. Gupta *et al.,* "A lightweight and robust secure key establishment protocol for internet of medical things in COVID-19 patients care," *IEEE Internet of Things Journal*, pp. 1–11, 2020. https://doi.org/10.1109/JIOT.2020.3047662.

[27] A. Sedik, M. Hammad, F. E. A. El-Samie, B. B. Gupta and A. A. A. El-Latif, "Efficient deep learning approach for augmented detection of Coronavirus disease," *Neural Computing and Applications*, pp. 1–18, 2021. https://doi.org/10.1109.s00521-020-05410-8.

[28] K. Goswami and B. G. Kim, "A design of fast high-efficiency video coding scheme based on markov chain monte carlo model and bayesian classifier," *IEEE Transactions on Industrial Electronics (IEEE)*, vol. 65, no. 11, pp. 8861–8871, 2018.

[29] J. H. Lee, G. S. Hong, Y. W. Lee, C. K. Kim, N. Park *et al.,* "Design of efficient key video frame protection scheme for multimedia Internet of Things (IoT) in converged 5G network," *Mobile Networks and Applications (Springer)*, vol. 24, no. 1, pp. 208–220, 2019.