

A Secure and Efficient Data Deduplication Scheme with Dynamic Ownership Management in Cloud Computing

Xuwei Ma, Wenyuan Yang, Yuesheng Zhu, Zhiqiang Bai

School of Electronic and Computer Engineering, Peking University

mxw971201@stu.pku.edu.cn, wyyang@pku.edu.cn, zhuys@pku.edu.cn, baizq@pku.edu.cn

Abstract—Encrypted data deduplication is an important technique for saving storage space and network bandwidth, which has been widely used in cloud storage. Recently, a number of schemes that solve the problem of data deduplication with dynamic ownership management have been proposed. However, these schemes suffer from low efficiency when the dynamic ownership changes a lot. To this end, in this paper, we propose a novel server-side deduplication scheme for encrypted data in a hybrid cloud architecture, where a public cloud (Pub-CSP) manages the storage and a private cloud (Pri-CSP) plays a role as the data owner to perform deduplication and dynamic ownership management. Further, to reduce the communication overhead we use an initial uploader check mechanism to ensure only the first uploader needs to perform encryption, and adopt an access control technique that verifies the validity of the data users before they download data. Our security analysis and performance evaluation demonstrate that our proposed server-side deduplication scheme has better performance in terms of security, effectiveness, and practicability compared with previous schemes. Meanwhile, our method can efficiently resist collusion attacks and duplicate faking attacks.

Index Terms—Data Deduplication, Cloud Computing, Access Control, Storage Management, hybrid cloud

I. INTRODUCTION

Nowadays, with the rapid growth of data volumes, there are urged demands for secure places where private data can be safely stored. Outsourcing the big data to the cloud is an efficient way to solve this problem [1], [2]. Despite all the advantages of cloud computing [3], duplicated data still waste abundant storage space and network bandwidth, and make data management more complicated [4].

The deduplication [5] is a process that identifies the same data by data similarity, which allows the cloud storage provider save the storage by storing only a single copy of the data owned by multiple owners. However, the existing schemes related to deduplication still incur some problems about dynamic ownership management and access control.

First of all, to protect privacy, many users encrypt data before uploading it to the cloud storage. Since the encryption key is randomly generated, the same data encrypted with different keys will produce different ciphertext, which will hinder deduplication. To solve this problem, some deduplication schemes propose that the owners of the same file share the same encryption key [6]–[14]. However, most of them do not consider the dynamic ownership changes that happen fre-

quently in cloud storage service [15]. The cloud users should be revoked from the valid ownership list once they request the cloud storage provider for data deletion/modification.

Second, to address dynamic ownership management, many schemes [4], [15]–[19] were proposed by using either trusted third party or semi-trusted third party to do proxy re-encryption work, such as Authority Party (*AP*) or The Public Cloud Provider (*Pub – CSP*). On the one hand, it could be difficult to implement a trusted third party in practical applications [20]. On the other hand, some schemes can not resist collusion attacks when the third party colludes with unauthorized users.

In this paper, we propose an novel scheme, which aims at efficiently solving the problem of deduplication with frequent cloud user revocation and new cloud user joining in cloud computing. In particular, different from existing data deduplication methods, which employ either trusted/semi-trusted third party to do proxy re-encryption work, our proposed scheme designs a hybrid cloud architecture, which includes a public cloud and further introduces a private cloud. In implementations, the introduced private cloud in our scheme is involved as a data owner and a proxy at the same time to 1) control access to outsourced data by performing re-encryption techniques and 2) manage the dynamic ownership when real data owner is offline or revoke his/her ownership.

Furthermore, we propose to enhance our scheme in terms of efficiency by 1) ensuring that the data owner performs encryption only when he/she is the initial uploader; 2) presenting an access control technique that verifies the validity of the data users before they download data; and 3) requiring the public cloud server can send ciphertext to the cloud user only when the cloud users are in the ownership list. Therefore, the abundant communication cost will be reduced. We evaluate the performance of our scheme through security analysis, comparison with existing work, and implementation-based performance evaluation.

Overall, our contributions are as follows:

- We propose a novel scheme, which introduces a hybrid cloud architecture including a public cloud and a private cloud, to efficiently solve the problem of deduplication with frequent cloud user revocation and new cloud user joining in cloud computing.

- We further propose to improve our scheme in efficiency. The extensive experiments prove the security, effectiveness, and efficiency of our method, which can significantly outperform previous data deduplication methods.

II. RELATED WORK

A. Deduplication without Dynamic Ownership Management

To solve deduplication problem, convergent encryption [21] was proposed, in which a data user got the key of data F by computing its hash code $K = H(F)$. Since data F will be encrypted with K , whoever holds the same data can produce the exact same encrypted data.

A server-aided encryption scheme (DupLESS) for data deduplication was proposed by Bellare et al. [7]. In DupLESS, an independent key server generates the key, which suffers from large computation time in block-level deduplication since it takes long time to generate keys. Liu et al. [22] proposed a client-side encryption that requests the data owner to do ownership check and deduplication, which is impractical. Cui et al. [12] proposed a deduplication scheme by using attribute-based access control technique under hybrid cloud environment.

Recently, a scheme [20] performed duplication check by short hash. Although it can resist offline brute-force, the collision rate will be relatively high because different data may have the same short hash. Later, the advanced scheme [23] used light weight techniques. However, this scheme does not apply to the situation where there is massive duplicated data, and is more suitable for individual users to store data on cloud disks. The drawback of these schemes is not considering the dynamic ownership management among the data users.

B. Deduplication with Dynamic Ownership Management

To address dynamic ownership management, many schemes were proposed by using either a trusted third party or semi-trusted third party to do proxy re-encryption work, such as AP or Pub-CSP. Wen et al. [16] constructed a convergent key sharing scheme. But this work requires the data user to encrypt/decrypt convergent keys and recover them from secret shares, which is unrealistic since the data user's computation power is limited. A server-side deduplication scheme was proposed by Hur et al. [15] in which the new cloud user joining was not considered since the total number of data users was fixed. Later, an enhanced scheme proposed by Yan et al. [4] provided a heterogeneous data storage management. However, it is not practical for the owner to be always online and send personalized keys to the holders. Beside, once the owner is offline, the access control will be entirely dependent on AP. Premkamal et al. [19] proposed an enhanced scheme with attribute-based access control by using the group key with the help of trusted entities.

Although these schemes address dynamic ownership management, there still are some security flaws. Since most of the schemes used either trusted/semi-trusted third party to do proxy re-encryption work, some data owners do not like to authorize a third party to control their data. Besides, taking

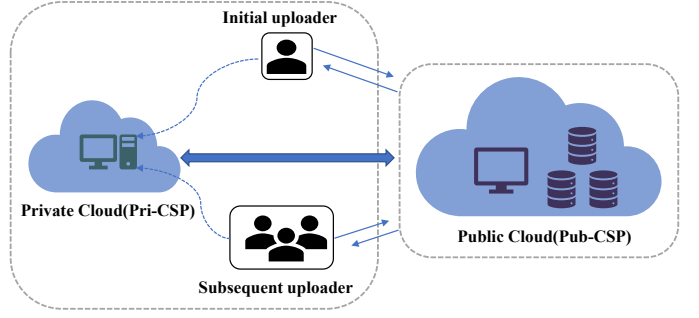


Fig. 1. Architecture of a data deduplication system.

scheme [4] as an example, once the malicious user conspires with the third party (such as a proxy server), it will happen that unauthorized users u who only has access to file A can also have invalid access to file B . This is the problem we intend to solve in this study.

III. SYSTEM MODEL

In this section, we describe the data deduplication system and define the adversary model. Only the file-level deduplication is considered in this paper.

A. Hybrid Architecture for Secure Deduplication

Figure 1 shows the architecture of the data deduplication system, which consists of three entities stated as follows.

- **Data users (DU).** This is an entity that wants to outsource data to Pub-CSP and access data later. In the authorized deduplication system, each user is issued a pair of secret key and public key (sk_u, pk_u) about Proxy Re-Encryption (PRE) and (SK_u, PK_u) for signature. Moreover, if a data user is the first one to upload file F_A , she/he will be defined as the owner $u_{1,A}$; if the file F_A already exist, the user will be defined as the holder $u_{i,A}$.
- **Public Cloud (Pub-CSP).** The entity that provides a data storage service in the public cloud. In this paper, Pub-CSP is assumed to be always online and capable to provide abundant storage.
- **Private Cloud (Pri-CSP).** The entity that provides an execution environment and infrastructure for data users as an interface between DU and Pub-CSP, since the computing resources of DU are limited and Pub-CSP is not fully trusted in practice. Pri-CSP maintains an ownership list for storing data, which is composed of a hash code set for the stored data, identities, and re-encrypted keys belonging to the owners.

In this system, we assume that the infrastructure of Pri-CSP is exclusively used by a single organization consisting of multiple users. It is owned, managed, and operated by the organization itself. Therefore, Pri-CSP can be trusted by all entities. Pub-CSP will strictly follow system design and protocols strictly, while they may still be curious about the raw data of data users. Moreover, We assume that Pri-CSP and DU would never collude with Pub-CSP due to different business interests.

B. Security Requirements

We follow [15] to ensure the following security requirements: Data privacy, Data consistency, Ownership verification, Ownership revocation, Collusion resistance.

Data privacy. Protecting the raw data from the Pub-CSP server and the unauthorized data users.

Data consistency. Ensuring tag consistency against any poison attacks. Any modifications of the ciphertext can be verified by the authorized data users.

Ownership verification. Any unauthorized data users who can not be verified his/her ownerships should be prevented from accessing to the ciphertext or any messages related to decryption stored in the cloud.

Ownership revocation. Any authorized data users who request the cloud storage provider for deletion/modification of the data F should be revoked from the valid ownership list and accessing to the data F anymore.

Collusion resistance. Any unauthorized data users who do not have valid ownership of data should be unable to access the raw data even if they collude with each other or collude with Pub-CSP.

IV. PRELIMINARIES AND DEFINITION

A. Proxy Re-Encryption (PRE)

Proxy re-encryption is initially proposed by Blaze et al. [24] called as atomic proxy functions, in which data users can decrypt ciphertext encrypted by key A with another key B without leaking any message about encryption/decryption key or plaintext. In this article, we have three actors in PRE:

- **Data Owner:** authorizes decryption rights to data user;
- **Data User:** receives decryption rights to access the encrypted data;
- **Proxy:** performs re-encryption functions to allow Data User to access the encrypted data;

Specifically, the Data Owner A encrypts data F by his/her public key, $C_A = En(pk_A, F)$. If the Data User B wants to decrypt the ciphertext C_A , he/she needs to get decryption rights from Data Owner A with his/her public key pk_B . If A agrees, he/she will produce an authorized key by $rk_{A \rightarrow B} = RG(sk_A, pk_B)$ and send it to the proxy.

In our scheme, Pri-CSP plays a role as **Data Owner** and **Proxy** at the same time to reduce client-side computation overhead. Table 1 summarizes the notations used in this paper.

B. Definition

In this section, we define a secure deduplication system for encrypted data with dynamic ownership management. The details of algorithms used in our scheme will be shown as follows.

1) System Setup

- **KeyGen1(u_{id}).** Based on the system parameters, user u generates its own key pair (sk_u, pk_u) about PRE and (SK_u, PK_u) for signature by inputting the unique u_{id} . Meanwhile, Pri-CSP gets (sk_{u_0}, pk_{u_0}) by having $u_{0,id}$ as owner u_0 .

TABLE I
NOTATION.

| Notations | Description |
|------------|---|
| F_{id}/F | The duplicated data F |
| u_{id}/u | The users of cloud service |
| K/K' | The symmetric key of F / Updated key |
| EK/EK' | The encrypted K/K' |
| REK/REK' | The encrypted EK/EK' by PRE |
| sk_{u_0} | The secret key of the Pri-CSP about PRE |
| pk_{u_0} | The public key of the Pri-CSP about PRE |
| sk_u | The secret key of u about PRE |
| pk_u | The public key of u about PRE |
| CT/CT' | The encrypted data F / Renewed ciphertext |
| $H(*)$ | The hash function |
| $HC(F)$ | The hash code set of data F |

2) Data Encryption and Decryption

- **Encrypt(K, F).** For plaintext data F , data owner u_1 encrypts it with the symmetric key K to get ciphertext CT .
- **Decrypt(K, CT).** Data holder u_i decrypts CT with key K and outputs F .

3) Key Control based on PRE Operated by Pri-CSP

- **KeyGen2(Λ)** takes a security parameter as input, and outputs a random symmetric key K .
- **En(pk_{u_0}, K)** takes pk_{u_0} and the symmetric key K as input and outputs an encrypted key EK .
- **RG(sk_{u_0}, pk_{u_i})** outputs re-encryption key $rk_{u_0 \rightarrow u_i}$ by taking sk_{u_0} and pk_{u_i} as input.
- **ReEn($rk_{u_0 \rightarrow u_i}, EK$)** takes input $rk_{u_0 \rightarrow u_i}$ and EK , and outputs $REK_{u_i} = En(pk_{u_i}, K)$ that can be decrypted with sk_{u_i} by $De(sk_{u_i}, REK_{u_i})$.

V. PROPOSED DEDUPLICATION SCHEME

A. Data Deduplication

Figure 2 shows the procedure of initial upload by owner u_1 . Figure 3 shows the procedure of deduplication when subsequent uploaders u_2 uploads the same data with u_1 . We assume that Pri-CSP plays a role as owner u_0 to control deduplication for owner u_1 .

- **Step 1 - Key Generation:** After system parameter generation, each DU asks $KeyGen1(u_{id})$ to generate key pair (sk_u, pk_u) for PRE and (SK_u, PK_u) for signature. Meanwhile, Pri-CSP gets (sk_{u_0}, pk_{u_0}) by $KeyGen1(u_0)$.
- **Step 2 - Duplication Check:** DU u_1 stores data F at Pub-CSP. He/She calculates $H(F)$, signs it with SK_{u_1} and sends data package $dp = \{H(F), sign(H(F), SK_{u_1})\}$ to Pub-CSP. The duplication check will be performed by Pub-CSP to verify if the same data has been stored already after verifying the signature. If the check is positive, go to Step 5. Otherwise, go to Step 3 to request key for encryption.
- **Step 3 - Data Storage:** When Pub-CSP defines u_1 is the first uploader of data F , it generates a unique F_{id} for F and contacts Pri-CSP to get key for encryption.

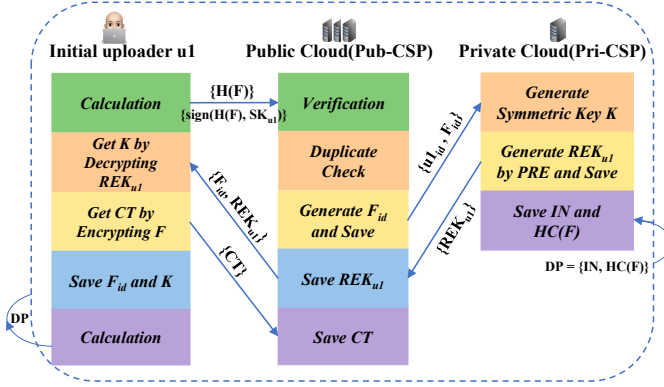


Fig. 2. Procedure of initial upload by owner u_1 .

Pri-CSP generates a random symmetric key K , and encrypts it with pk_{u_0} to get EK , then applies PRE to EK to get cipherkey REK_{u_1} for DU u_1 . Pub-CSP sends (F_{id}, REK_{u_1}) to u_1 after receiving it from Pri-CSP. User u_1 decrypts REK_{u_1} by $De(sk_{u_1}, REK_{u_1})$ with his/her own private key to get K , and then u_1 encrypts data F by $Encrypt(K, F)$. Moreover, user u_1 will randomly select several indexes: $X = \{x_1, x_2, \dots, x_n\}$ that indicate the specific parts of F (e.g., x_1 equals first 0.5% of F ; x_2 equals first 2.5% of F). Then, based on the index X , u_1 calculates the hash code set of data F as $HC(F) = \{HC(F_1), HC(F_2), \dots, HC(F_N)\}$. Then u_1 sends the data package $dp_1 = \{u_{1,id}, REK_{u_1}, CT\}$ to Pub-CSP and sends the data package $dp_2 = \{F_{id}, u_{1,id}, X, HC(F)\}$ to Pri-CSP. Last but not least, Pub-CSP and Pri-CSP both maintain the ownership list separately for each data, what saved in Pub-CSP is P_1 , and what saved in Pri-CSP is P_2 , where X' and $HC'(F)$ is randomly selected from X and $HC(F)$.

$$\begin{aligned} P_1 &= \{F_{id}, CT, H(F), (u_{1,id}, REK_{u_1})\} \\ P_2 &= \{F_{id}, X', HC'(F), EK, (u_{1,id}, rk_{u_0 \rightarrow u_1})\} \end{aligned} \quad (1)$$

- **Step 4 - Duplicated Data Upload:** Later on, DU u_2 wants to store the same data F at Pub-CSP by sending the data package $dp = \{H(F), sign(H(F), SK_{u_2})\}$.
- **Step 5 - Deduplication:** Pub-CSP performs duplication check as in step 2 after verifying that $u_{2,id}$ is not in the ownership list. When the duplication check is positive, Pub-CSP contacts Pri-CSP for deduplication. Pri-CSP further verifies the ownership of DU u_2 by challenging the hash code set of F before performing deduplication, which ensures the data ownership, since there is a probability that $H(F)$ is eavesdropped or gained by the malicious party. If the ownership verification is positive, Pri-CSP generates the re-encrypted key REK_{u_2} and sends it back to Pub-CSP for saving. Once Pub-CSP receives REK_{u_2} , it sends REK_{u_2} with ciphertext CT to u_2 . When u_2 gets REK_{u_2} , he/she decrypts it with sk_{u_2} to get K . Then u_2 gets plaintext of data F by running $Decrypt(K, CT)$ and checks data consistency, while if

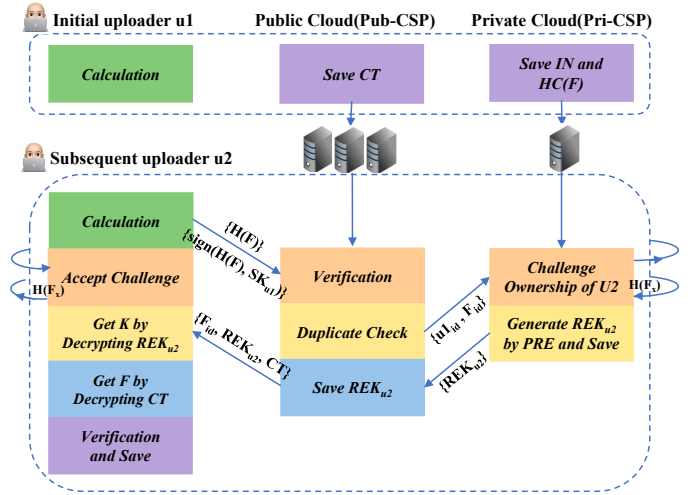


Fig. 3. Procedure of deduplication.

u_2 does not want to check data consistency immediately, he/she can check it by downloading CT and $H(F)$ later. Through data deduplication, both u_1 and u_2 can access the same data F that is stored only once at Pub-CSP.

B. Ownership Revocation

Any authorized data users who request Pub-CSP for the deletion/modification of their data stored in the cloud storage should be removed from the valid ownership list, and be prevented from accessing the previously saved data.

Ownership Revocation of Data Holder. Figure 4 shows the procedure of data update by DU u_2 in the context of data deduplication.

- **Step 1:** User u_2 sends a request of data deletion to Pub-CSP by providing $dp = \{F_{id}, sign(F_{id}, SK_{u_2})\}$.
- **Step 2:** Pub-CSP deletes the storage record of u_2 after verifying the signature. Then Pub-CSP asks owner u_1 /owner u_0 blocking u_2 's future access data F . If owner u_1 is online and willing to do so, go to step 3, otherwise, go to step 4.
- **Step 3:** If owner u_1 is online and willing to do dynamic ownership management, Pri-CSP generates the new symmetric key K' before generating REK'_{u_1} , and sends it back to Pub-CSP. User u_1 obtains the updated key K' by running $De(sk_{u_1}, REK'_{u_1})$ once he receives $dp = \{CT, REK'_{u_1}\}$ from Pub-CSP, and u_1 re-encrypts data F with new key K' to get the updated ciphertext CT' . Then u_1 re-uploads CT' to Pub-CSP.
- **Step 4:** If owner u_1 is offline or asks owner u_0 representing u_1 to do ownership management. Pub-CSP sends (F_{id}, CT) to owner u_0 , and u_0 decrypts CT by running $De(sk_{u_0}, EK)$ first before running $Decrypt(K, CT)$ to get plaintext of data F . The rest is almost same as step 3 that u_0 generates the new symmetric key K' and gets the updated REK'_u for the rest users who still own data

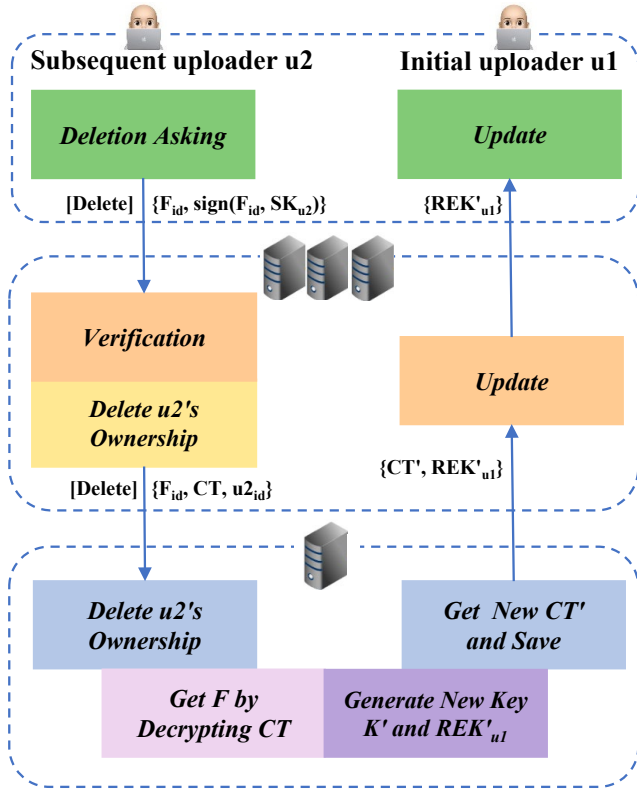


Fig. 4. Procedure of ownership revocation of data holder u_2 when owner u_1 is offline.

F , and re-encrypts data F with K' before re-uploading the updated ciphertext CT' to Pub-CSP.

- **Step 5:** Update the ownership lists saved on the cloud. To make it easier to understand, we assume that there are only two owners u_1 and u_2 sharing the same data F . The p_1 and p_2 are updated as P'_1 and P'_2 separately.

$$\begin{aligned}
 P_1 &= \{F_{id}, CT, H(F), (u_{1,id}, REK_{u_1}), (u_{2,id}, REK_{u_2})\} \\
 P'_1 &= \{F_{id}, CT', H(F), (u_{1,id}, REK'_{u_1})\} \\
 P_2 &= \{F_{id}, X', HC'(F), EK, (u_{1,id}, rk_{u_0 \rightarrow u_1}), (u_{2,id}, rk_{u_0 \rightarrow u_2})\} \\
 P'_2 &= \{F_{id}, X', HC'(F), EK', (u_{1,id}, rk_{u_0 \rightarrow u_1})\}
 \end{aligned}$$

Ownership Revocation of Data Owner. The data owner's ownership revocation basically is the same as the data holder's except for some details shown as follows.

- **Step 1:** User u_1 sends a request of data deletion to Pub-CSP by providing $dp = \{F_{id}, \text{sign}(F_{id}, SK_{u_1})\}$.
- **Step 2:** Pub-CSP deletes the storage record of u_1 after asking owner u_1 to pick one from the holders to be the new owner u'_1 , or u_1 can just pick owner u_0 as the new owner u'_1 to manage the dynamic ownership afterwards. Since we have the new owner u'_1 now, the remaining steps are the exactly same as **Ownership Revocation of Data Holder**.

TABLE II
COMPARISON OF SECURE DEDUPLICATION SCHEMES.

| Scheme | CE | RCE | DedupDUM | Our Scheme |
|------------------------------|----|-----|----------|------------|
| Encrypted data deduplication | ✓ | ✓ | ✓ | ✓ |
| Tag consistency | × | ✓ | ✓ | ✓ |
| Access control | × | × | ✓ | ✓ |
| Dynamic ownership management | × | × | ✓ | ✓ |
| Possession proof | × | × | × | ✓ |

VI. PERFORMANCE EVALUATION

A. Comparison with Existing Work

Table 2 is a comparison among four data deduplication schemes, the convergent encryption(CE) [21], the randomized convergent encryption (RCE) [25], the data deduplication with dynamic user management [18] (DedupDUM) and our scheme. On the basis of encrypted data deduplication, tag consistency, access control, dynamic ownership management, and possession proof.

All the schemes guarantee data confidentiality and privacy by saving encrypted data. However, scheme CE is vulnerable to the tag consistency attack. While other schemes can guarantee data integrity by enabling DU to check the tag consistency of the received data. Scheme DedupDUM solves the dynamic ownership management problem by using the group key generated by DU's public key, which supports ownership revocation and new user joining. However, they do not verify that the holders hold the entire original file instead of only having a tag, the fake ciphertext and ID, and they also do not consider collusion attacks caused by the dishonest cloud server and attackers.

Different from the previous schemes, our scheme achieves dynamic ownership management by maintaining the ownership list for each data F at Pub-CSP and Pri-CSP separately. Since the re-encrypted key REK_{u_i} is generated with pk_{u_i} , our scheme supports the cloud user revocation and new user joining. Moreover, deduplication is performed after checking that DU 's access to file F is unauthorized and he/she does own the whole file. Therefore, the communication overhead can be greatly reduced.

B. Efficiency Analysis

The comparison on the basis of communication overhead among four schemes is shown in **Table 3**. C_C denotes the size of the encrypted data, C_{ID} denotes the size of a cloud user's u_{id} , C_H denotes the size of a hash code, C_{HC} denotes the size of hash code set of data F , C_K denotes the size of a key, C_p denotes the size of a public key.

For the first upload of data F , scheme CE, RCE, and DedupDUM have the same upload message sizes. In our scheme, it increases the size of the hash code set $HC(F)$ used for DU 's ownership verification before deduplication. However, for the subsequent upload of F , our scheme only uploads $H(F)$ before ownership is verified or access is checked while the other schemes need to re-upload all messages every time as shown in Table 3.

TABLE III
COMMUNICATION OVERHEAD.

| Scheme | For initial uploader | | | | For subsequent uploader |
|------------|----------------------------------|-----------------------|-----------------------|-------------|----------------------------------|
| | Upload message size | Download message size | Rekeying message size | Key size | Upload message size |
| CE | $C_C + C_H + C_{ID}$ | C_C | — | C_K | $C_C + C_H + C_{ID}$ |
| RCE | $C_C + C_K + C_H + C_{ID}$ | $C_C + C_K + C_H$ | — | C_K | $C_C + C_K + C_H + C_{ID}$ |
| DedupDUM | $C_C + C_K + C_H + C_{ID} + C_P$ | $C_C + C_K + C_H$ | C_P | $C_K + C_P$ | $C_C + C_K + C_H + C_{ID} + C_P$ |
| Our scheme | $C_C + C_H + C_{HC} + C_{ID}$ | $C_C + C_K + C_H$ | C_K | C_K | $C_H + C_{ID}$ |

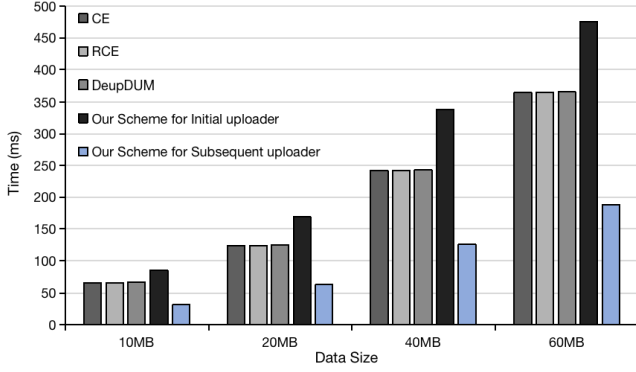


Fig. 5. Computation time for upload.

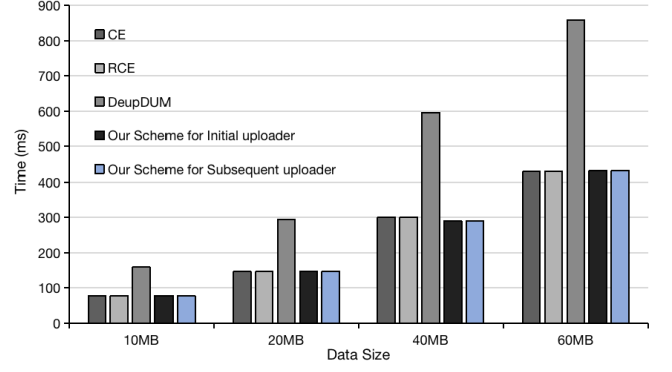


Fig. 6. Computation time for download.

Concerning the rekeying message size, the DedupDUM and our scheme increase the size of the re-encryption key while scheme CE and scheme RCE do not consider key updating. However, even though the group key is used to manage ownership revocation, the encryption key K determined by data F still never changes in DedupDUM once it is settled, which is unsecured since the withdrawn owners can collude with Pri-CSP for getting ciphertext before rekeying. Our scheme updated K as long as there is ownership revocation, which is more secure and computation overhead is accepted since it can be performed by Pri-CSP instead of DU.

C. Performance Evaluation

In this section, we compare our scheme with previous works. We follow common practice [18], [21], [25] for fair comparison, each cryptographic operation is implemented by using the umbral library ver. 0.3.0 and Crypto library ver. 1.4.1. We perform data encryption and decryption algorithm with AES where the key is 128-bit. The data size ranges from 10MB to 60MB. The testing environment is Intel(R) Core(TM) i5-7300HQ CPU 3.1 GHz 16.0GB RAM.

1) Computation time for upload:

We tested the efficiency of the file uploading process under different schemes, and scheme DedupDUM requires the same computations as scheme CE and scheme RCE. As shown in **Figure 5**, there is a slight increase in our scheme compared with others since the process in our scheme includes calculating hash code and hash code set of data F , signing and verifying the signature, decrypting re-encrypted key, and encrypting data F with AES. Moreover, We can see that our scheme has great advantages in the duplicated file uploading process. Since

our scheme only uploads $H(F)$ before ownership is verified or access is checked while the other schemes need to re-upload all message every time. The abundant communication cost can be saved in our scheme.

2) Computation time for download:

Compared with other schemes, we apply the ownership verification mechanism to verify whether DU has access to whole data or not by challenging DU with random $H(F_x)$ in the hash code set, which can save abundant communication cost when DU does not have ownership. The comparison of computation time for downloading ciphertext among four schemes is depicted in **Figure 6**.

3) Computation time for encryption and decryption:

In the encryption stage, the DedupDUM and our scheme resolve the dynamic ownership management problem by performing re-encryption. While our schemes take a shorter time than DedupDUM since Pub-CSP needs to decrypt and re-encrypt the ciphertext for each subsequent uploader in DedupDUM, which will result in high computation complexity as the number of holders grows. The detailed encryption and decryption time for different data sizes (ranging from 10MB to 60MB) is shown in **Figure 7**.

4) Computation time for deduplication:

In this process, the total number of DU is set up as 25, and each of them have a different 2MB-sized file, which means the total size of data saved in the cloud is 50MB when the duplicate ratio(DR) is 0. Specifically, when DR is 20%, which means 5 DUs own the same file, and the rest 20 DUs own different files. From **figure 8**, it shows how computation time changes during each process with DR grows from 0 to 100%. It can be observed that such a process is very efficient,

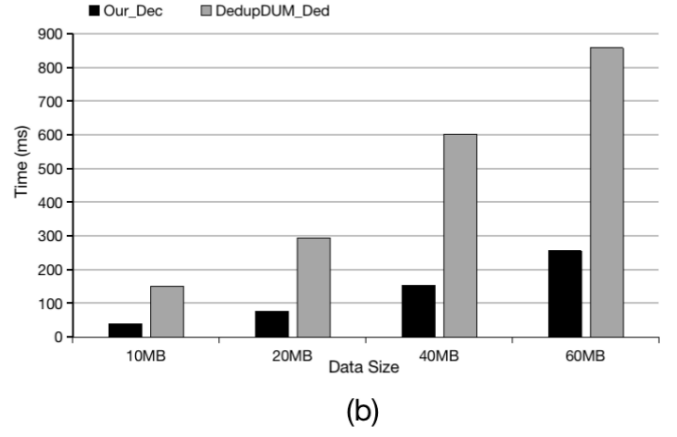
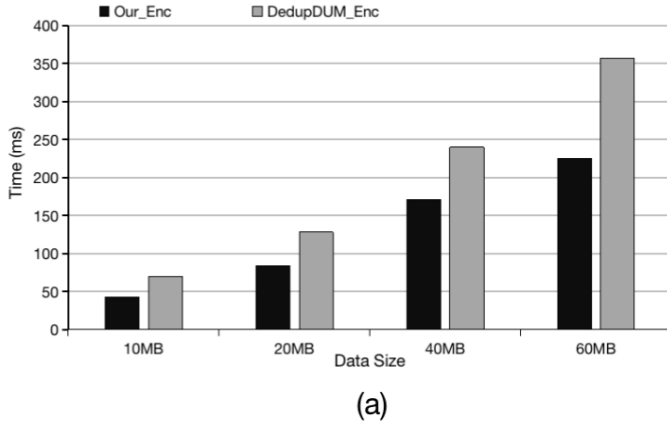


Fig. 7. Computation time for (a) encryption and (b) decryption.

taking less than 0.373 seconds for data upload, and 0.251 for deduplication when the ratio is 100%. Therefore the proposed deduplication scheme can greatly reduce data uploading time.

VII. SECURITY ANALYSIS

In this section, our scheme security is analyze on the basis of data privacy, data consistency, data ownership verification, ownership revocation, and collusion resistance.

A. Data Privacy

In terms of data privacy, the raw data should be prevented from Pub-CSP (honest but curious) and unauthorized data users. Therefore, there are normally two kinds of attacks, separately from Pub-CSP and invalid data users. Firstly, as an attack from Pub-CSP is concerned, what is saved on Pub-CSP is the authorized DU's re-encrypted key that is encrypted through PRE by Pri-CSP and only can be decrypted by DU's private key, since Pri-CSP and authorized users will not collude with Pub-CSP considering their profits, it is impossible for the Pub-CSP to get plaintext by cipher key. Second, supposing an unauthorized user u_2 requests data F with $(F_{id}, u_{1,id})$ (u_1 is valid, and using $u_{2,id}$ will simply not pass the ownership check), Pub-CSP searches the ownership list $P2 = \{F_{id}, CT, H(F), (u_{1,id}, REK_{u_1})\}$, and return $\{F_{id}, CT, REK_{u_1}\}$ to user u_2 based on $(F_{id}, u_{1,id})$. Since REK_{u_1} can only be decrypted by the private key of user u_1 , it is computationally infeasible for user u_2 to obtain plaintext of F by decrypting CT with REK_{u_1} . Therefore, data privacy against the honest-but-curious Pub-CSP and unauthorized users is guaranteed.

B. Data Consistency

In the data deduplication scheme, data integrity may be threatened by a poison attack on tag consistency, which could be identified during the decryption process by the data holders. Assuming that attacker u_2 owns the same data F_A with authorized user u_3 , u_2 uses $F_B \neq F_A$ to generate a fake ciphertext CT_B , then uploads $(H(F_A), CT_B)$ to Pub-CSP to pretend to be CT_A . When the authorized user

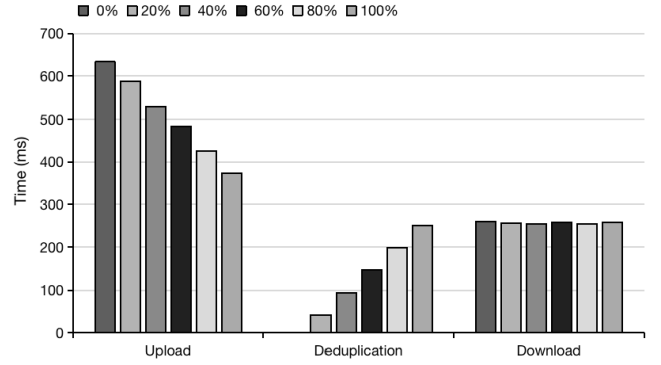


Fig. 8. Computation time for each process with a duplicate ratio grows.

u_3 wants to upload F_A , u_3 sends $H(F_A)$ to Pub-CSP to check duplication. Since $H(F_A)$ exists, Pub-CSP asks Pri-CSP performing deduplication. Pub-CSP sends (CT_B, REK_{u_3}) back to user u_3 after deduplication, and u_3 checks whether $H(\text{Decrypt}(\text{Dec}(sk_{u_3}, REK_{u_3}), CT_B)) = H(F_A)$ holds or not, if this is not consistent then u_3 drops the messages and reports this to Pub-CSP. Therefore, our scheme guarantees the data integrity.

C. Data Ownership Verification

In our scheme, the data ownership is verified by challenging DU with random $H(F_x)$ in hash code set (randomly select specific parts of the data, e.g., the hash code of 10.5-14.3% of F). F_x is randomly selected and function $H(*)$ is non-invertible, therefore it is impossible to calculate $H(F_x)$ without the original plaintext.

D. Ownership Revocation

The ownership withdrawn data user should be restricted to access data F . Our scheme guarantees ownership revocation by using Pri-CSP playing a role as owner u_0 . Whenever the data owner u_1 withdraws his/her ownership or some other holders may request to delete or modify their data, the owner

u_0 deletes the ownership information of the requestor from the ownership list, and u_0 (if u_1 is offline or revokes his/her ownership or asking u_0 update for him/her) re-encrypts the plaintext with the new symmetric key before re-uploading it to Pub-CSP, and updates the re-encrypted keys of the rest users. Hence, the withdrawn data owner will not be able to pass the access check and decrypt the latest ciphertext with the non-updated cipher-key.

E. Collusion Resistance

Since Pri-CSP is fully trusted, we further discuss the collusion attacks launched by dishonest Pub-CSP and attackers. First, if unauthorized user u_1 colludes with dishonest Pub-CSP for getting plaintext of data F , Pub-CSP will ask Pri-CSP performing deduplication for u_1 with faking information. Pri-CSP will verify u_1 's ownership of data F before sharing the re-encrypted key REK_{u_1} of F with u_1 . Since u_1 does not have plaintext, he/she can not pass the ownership check to get cipher-key even though Pub-CSP sends u_1 the ciphertext dishonestly. Second, since each cipher-key can only be decrypted by the corresponding user, the unauthorized users are not able to decrypt them even if they collude with each other. Therefore, our scheme guarantees collusion resistance.

VIII. CONCLUSION

In this paper, we proposed a secure and practical scheme that managed the encrypted data with deduplication, based on ownership challenge under a hybrid cloud architecture, where Pub-CSP manages the storage and Pri-CSP plays a role as owner u_0 and proxy at the same time to perform deduplication and dynamic ownership management. Further, our scheme proves that the owner holds the real data alone pass the data ownership, and encrypted data can be securely accessed because only authorized data holders can obtain the symmetric keys used for data decryption. Security analysis, comparison with existing work, and implementation-based performance evaluation show that our scheme is secure and efficient, and resists collusion attacks and duplicate faking attacks.

ACKNOWLEDGMENT

This work was supported in part by the National Innovation 2030 Major S&T Project of China under Grant 2020AAA0104203, and in part by the Nature Science Foundation of China under Grant 62006007. We thank all the anonymous reviewers for their constructive comments and suggestions. The corresponding author of this paper is Yuesheng Zhu.

REFERENCES

- [1] Ibrahim, Abaker, Targio, Hashem, Ibrar, Yaqoob, Nor, Badrul, Anuar, and Salimah, "The rise of "big data" on cloud computing: Review and open research issues," *Information Systems*, vol. 47, no. Jan., pp. 98–115, 2015.
- [2] F. M. Awaysheh, M. N. Aladwan, S. Alawadi, J. C. Cabaleiro, and T. F. Pena, "Security by design for big data frameworks over cloud computing," *IEEE Transactions on Engineering Management*, vol. PP, no. 99, 2021.
- [3] Duan and Qiang, "Cloud service performance evaluation: status, challenges, and opportunities-a survey from the system modeling perspective," *Digital Communications & Networks*, pp. 101–111, 2016.
- [4] Z. Yan, L. Zhang, W. Ding, and Q. Zheng, "Heterogeneous data storage management with deduplication in cloud computing," *IEEE Transactions on Big Data*, pp. 1–1, 2017.
- [5] S. Quinlan and S. Dorward, "Venti: A new approach to archival storage," *proc.usenix conf.on file & storage tech*, 2002.
- [6] G. R. Blakley and C. Meadows, "Security of ramp schemes," in *Advances in Cryptology, Crypto 84, Santa Barbara, California, Usa, August*, 1984.
- [7] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: Server-aided encryption for deduplicated storage," in *Usenix Conference on Security*, 2013.
- [8] X. Jin, L. Wei, M. Yu, N. Yu, and J. Sun, "Anonymous deduplication of encrypted data with proof of ownership in cloud storage," *IEEE/CIC International Conference on Communications in China*, 2013.
- [9] J. Li, X. Chen, M. Li, J. Li, P. P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, 2014.
- [10] J. Li, Y. K. Li, X. Chen, P. Lee, and W. Lou, "A hybrid cloud approach for secure authorized deduplication," *Parallel & Distributed Systems IEEE Transactions on*, vol. 26, no. 5, pp. 1206–1216, 2015.
- [11] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Interactive message-locked encryption and secure deduplication," in *Springer, Berlin, Heidelberg*, 2015.
- [12] H. Cui, R. H. Deng, Y. Li, and G. Wu, "Attribute-based storage supporting secure deduplication of encrypted data in cloud," *IEEE Transactions on Big Data*, pp. 1–1, 2017.
- [13] W. Shen, Y. Su, and R. Hao, "Lightweight cloud storage auditing with deduplication supporting strong privacy protection," *IEEE Access*, vol. 8, pp. 44 359–44 372, 2020.
- [14] N. Almrezeq, M. Humayun, A. El-Aziz, and N. Z. Jhanjhi, "An enhanced approach to improve the security and performance for deduplication," *Turkish Journal of Computer and Mathematics Education (TURCO-MAT)*, vol. 12, no. 6, pp. 2866–2882, 2021.
- [15] Hur, Junbeom, Koo, Dongyoung, Shin, Youngjoo, Kang, and Kyungtae., "Secure data deduplication with dynamic ownership management in cloud storage," *IEEE Transactions on Knowledge & Data Engineering*, vol. 28, no. 11, pp. 3113–3125, 2016.
- [16] W. Mi, K. Ota, L. He, J. Lei, C. Gu, and S. Zhou, "Secure data deduplication with reliable key management for dynamic updates in cps," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 137–147, 2016.
- [17] J. Shen, X. Deng, and Z. Xu, "Multi-security-level cloud storage system based on improved proxy re-encryption," *EURASIP J. Wirel. Commun. Netw.*, vol. 2019, p. 277, 2019. [Online]. Available: <https://doi.org/10.1186/s13638-019-1614-y>
- [18] H. Yuan, X. Chen, T. Jiang, X. Zhang, Z. Yan, and Y. Xiang, "Dedupdum: Secure and scalable data deduplication with dynamic user management," *Information Sciences*, vol. 456, pp. 159–173, 2018.
- [19] P. K. Premkamal, S. K. Pasupuleti, A. K. Singh, and A. Pja, "Enhanced attribute based access control with secure deduplication for big data storage in cloud," *Peer-to-Peer Networking and Applications*, no. 3, 2021.
- [20] S. Zhang, H. Xian, Z. Li, and L. Wang, "Secdedup: Secure encrypted data deduplication with dynamic ownership updating," *IEEE Access*, vol. 8, pp. 186 323–186 334, 2020.
- [21] 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 *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, 2002.
- [22] J. Liu, N. Asokan, and B. Pinkas, "Secure deduplication of encrypted data without additional independent servers," in *CCS. ACM*, 2015, pp. 874–885.
- [23] S. Srisakthi and G. A. Ansari, "Pcsp: A protected cloud storage provider employing light weight techniques," *Information Security Journal A Global Perspective*, no. 4, pp. 1–12, 2021.
- [24] M. Blaze, "Divertible protocols and atomic proxy cryptography," *EUROCRYPT 1998*, 1998.
- [25] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2013.