IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

SecDedup: Secure Encrypted Data Deduplication with Dynamic Ownership Updating

SHUGUANG ZHANG[1,2,3], HEQUN XIAN[1,2], ZENGPENG LI[1] LIMING WANG[3]

[1]College of Computer Science and Technology, Qingdao University, Qingdao 266071 China
[2]State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences), Beijing 100093 China
[3]Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093 China

Corresponding author: Hequn Xian (e-mail: xianhq@qdu.edu.cn).

This work was supported in part by The National Natural Science Foundation of China (61802214),Project ZR2019MF058 of Shandong Provincial Natural Science Foundation,Open Project of State Key Laboratory of Information Security(2020-MS-09).

ABSTRACT Deduplication eliminates duplicated data copies and reduces storage costs of cloud service providers. However, deduplication of encrypted data is difficult. Current solutions rely heavily on trusted third parties, and do not address the popularity of data, resulting in unsatisfying security and efficiency. A secure encrypted data deduplication scheme based on data popularity is proposed. Check tags are calculated via bilinear mapping to determine whether different encrypted data originate from the same plaintext. Ciphertext policy attribute-based encryption is applied to protect the tags. A secure key delivery scheme is designed to pass the data encryption key from an initial data uploader to subsequent uploaders via the cloud server in an offline manner. The cloud server can perform deduplication without the assistance of any online third party. Security analysis and simulation experiments are provided, proving the practicability and efficiency of the proposed scheme.

INDEX TERMS Deduplication, proxy re-encryption, bilinear mapping, data ownership update.

I. INTRODUCTION

Cloud storage is a classical model of data storage in which digital data are stored in logical pools. In particular, significant growth of data volume promotes a rapid development of cloud storage. Some object storage services like Amazon S3, Oracle Cloud Storage and Microsoft Azure Storage, are all examples of storage that can be hosted and deployed with cloud storage characteristics. During its development, cloud computing including of storage has become a convenient and cost efficient way for companies and mainstream users to outsource data while using remote, shared servers located in the "cloud". However, although cloud computing has all sorts of advantages, there still exist some barriers or weaknesses that casually put a brake on the development of cloud storage [8]. One of the most striking single weakness is massive *duplicated data*. These data are produced as different users upload groups of identical data to the cloud, which significantly raises the storage costs of cloud servers. Cloud service providers eagerly seek efficient ways to address the problem of *duplicated data*.

Notably, *deduplication* is proposed and regarded as an effective solution which dictates that any duplicated data copy should be logically stored for only once and shared by multiple users. To our knowledge, current deduplication techniques can be classified into four categories: client-side deduplication, cloud-side deduplication, block-level deduplication, and file-level deduplication. Different techniques and strategies are often combined depending on the application scenarios, such as security and efficiency factors. Apparently, achieving deduplication on plaintext is trivial. But in reality, most of users tend to store the ciphertext of the private sensitive data instead of storing plaintext directly. In other words, it is generally known that identical plaintext data will be encrypted into different ciphertext if the encryption keys are different. In this case, how to identify duplication from encrypted data has become a challenging problem.

To overcome the above limitation, researchers proposed the Convergent Encryption (CE) [1]. Notably, in CE, the hash value of the data is used as the encryption key, meanwhile, the hash value of encrypted data is used to obtain a check-tag, which is used as a duplication identifier. Unfortunately, CE has been proven incompliance with semantic security re-

quirements, i.e., it is possible for an adversary to get plaintext information by performing brute-force attacks on encrypted data [2]. Thus, a new solution using a trusted third party to achieve the deduplication on encrypted data is proposed [3]. In this situation, the trusted third party has authority to access all the plaintext data, and all users compute duplicate check-tags with the assistance of the trusted third party. However,it is difficult to implement a trusted third party in real-world applications [3]–[7]. Thus, an awkward question was raised naturally,

> *Is it possible to design an efficient deduplication systems without recourse to a trusted third party?*

In order to address this question and to circumvent drawbacks of the previous schemes as mentioned above, we present a secure encrypted data deduplication method named SecDedup. Similar to CE, the duplicate check-tag is computed based on plaintext information. But in order to increase the time complexity of adversary attacks, we use bilinear mapping instead of ordinary hash function to generate redundant tags. In particular, a random parameter is required to construct a bilinear mapping. The parameter is shared by users holding the same data to prevent brute-force attacks.

Further, to our knowledge, Proxy Re-Encryption (PRE) is an important cryptographic primitive which can be used to secure data sharing, e-mail forwarding etc. Various PRE schemes and its optimizations were proposed (e.g., [9]–[11]) after the first candidate PRE scheme proposed by Blaze et al. [12] at EUROCRYP. In this work, we introduce PRE to achieve the functionality of how to share the random parameter. In short, PRE allows a semi-trusted proxy to convert the initial uploader's ciphertext into a ciphertext for subsequent uploaders without exposing the underlying plaintext (i.e., random parameter) [26]. We stress that SecDedup is independent of any online third party, it provides mechanisms for dynamic data ownership update. Furthermore, it is suitable for both file and block oriented storage systems ,in which files and blocks are treated as the basic deduplication object units respectively [3]. Encrypted data deduplication technology allows eliminating redundant encrypted data stored in the cloud environment, such as websites hosted in public clouds, datasets maintained by Internet service providers or comprehensive cloud storage service providers.

Our main contributions are as follows:

- **New duplicate check protocol.** We propose a new duplicate check protocol to uniquely identify each data copy. It is constructed via PRE from bilinear mapping. Further, the cloud server is able to verify whether the data copy is duplicated without knowing any plaintext information.
- **Secure key delivery protocol.** A secure key delivery protocol is constructed, in which the initial uploader of the data is able to deliver the encryption key to subsequent uploaders via the cloud server in an offline manner. Based on this protocol, encrypted data deduplication can be performed securely.
- **Lightweight data ownership update protocol.** We design a lightweight data ownership update protocol. Users cannot access the original data after deleting or modifying them; Data cannot be accessed unless a user testifies his ownership to the cloud server.

The rest of this paper is organized as follows: Section II introduces related works. Section III formalizes the problem and defines the design goals. Section IV contains some preliminaries and the construction of our scheme. Security analysis is provided in Section V, followed by numerical and experimental analysis in Section VI. Finally, Section VII concludes this paper.

II. RELATED WORK

We categorize some related works and describing them in the following subsections according to their characteristics.

A. CONVERGENT ENCRYPTION IN DEDUPLICATION

The easiest way to achieve deduplication is that all users encrypt their data with a global key, then store the cipertexts and the key in the cloud server. However, since the cloud server is honest but curious, data privacy may be compromised. In order to perform secure deduplication, the convergent encryption (CE) was presented [1]. This deterministic encryption scheme allows identical data to be encrypted into the same ciphertext. Nonetheless, CE does not provide semantic security for data with low entropy. Li et al. [15] solved the problem of convergent key management by adopting the RAMP secret sharing scheme, but still did not address the inherent security issues of CE. M.Bellare et al. proposed the message-locked encryption(MLE), with more sophisticated but solid security than CE [2]. Secure client-side deduplication schemes with public auditing capabilities were proposed [18]. They are all based on CE design and used other cryptography methods to further protect data security [19]. Although client-side deduplication can save network bandwidth, it is vulnerable to online brute-force attacks, in which an adversary may enumerate and upload different convergent encrypted data, and observe which data already exist on the cloud server [37].

B. DEDUPLICATION WITH THIRD PARTIES

It is easier to use a trusted online third party to assist the cloud server to perform deduplication on encrypted data. P. Puzio et al. proposed the ClouDedup scheme [4], in which data encryption and decryption process are outsourced to a trusted third party. M. Bellare et al. proposed the DupLESS scheme [3], users with the same data are able to acquire the same encryption key with a trusted third party.Stanek et al. divided cloud data into two categories according to the privacy level [5]. Data of low privacy are protected by CE; data of high privacy are protected by double-layer encryption with the assistance of a third party. Y.L.He et al. proposed an similar scheme which is aware of data popularity [17]. P. Puzio proposed the PerfectDedup scheme [6], using a full hash function to query the privacy level of data through trusted third parties. Similar to the method in Stanek's work,

in PerfectDedup, data of low privacy are protected with CE; data of high privacy are protected with traditional symmetric encryption. Yan's scheme [20] exploited the Elliptic Curves Cryptography (ECC) to generate redundant tags with the assistance of a third party. Hui et al. presented an attribute-based storage system with secure deduplication in a hybrid cloud setting, in which a private cloud is responsible for duplicate detection and a public cloud manages the storage [7]. Although a trusted third party is convenient for deduplication, it is difficult to implement and deploy a completely trusted third party in practical application environments [38]. Additionally, it will significantly increase the communication overhead.

C. DYNAMIC OWNERSHIP UPDATE

The functionality of dynamic data ownership updating is vital to data deduplication systems. It would be a security flaw if some users are able to access the original data after deleting or modifying them. Although the problem is studied [21], the security of the scheme is still unsatisfying since CE is used as the cryptographic primitive. D.Koo et al. proposed a privacy-preserving deduplication scheme of encrypted data with dynamic ownership management in fog computing [22]. In their work, the encrypted data, the ownership of data and the user's decryption key can be updated automatically with zero interaction, which not only reduces communication overhead but also guarantees forward security. However, since public key encryption is used , the computational overhead on the user side is relatively high. The scheme is vulnerable to offline brute-force attacks, in which an adversary can identify whether some data is duplicated simply by sending the hash to the cloud.Other revocation based schemes are proposed which may provide ways for ownership management [16].

D. SUMMARY

In the related works discussed above, most of the schemes rely on a online trusted third party as in Xiong's sheme [23],Wang's scheme [30], et al. Some of the schemes achieved dynamic ownership update, but their using of public key encryption or re-encryption caused significant overhead to the process. In this paper, we propose a secure encrypted data deduplication scheme named SecDedup, which not only achieves deduplicaion of encrypted data without any online third party, but also allows dynamic update of data ownership in a lightweight manner. It is worth noting that we only use public key encryption algorithm to generate signatures and protect encryption keys, the computational overhead of the scheme is relatively low.

III. SYSTEM OVERVIEW AND DESIGN GOALS

A. SYSTEM OVERVIEW

The proposed scheme contains three types of entities as shown in Figure 1, the key distribution center(KDC), the users $U_i(i \in [1, n])$, and the cloud service provider(CSP). Authentication are required for users to access the KDC or the CSP. We assume that data transmission is protected by a secure communication protocol (e.g., SSL/TLS). We do not discuss further details about identification ,authentication and secure communication, as these topics are out of the scope of this work.

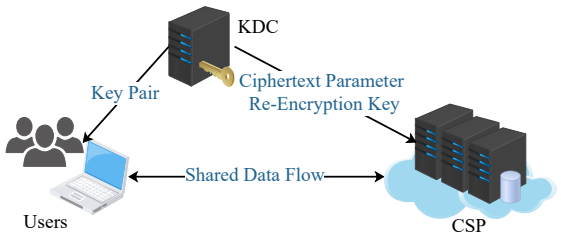

FIGURE 1: System model

In the system setup phase,the KDC provides users with public and private key pairs, it deploys the ciphertext parameters and re-encryption keys into the CSP. It is worth noting that the KDC is no longer needed after the system setup phase, and can go offline.The CSP is responsible for providing storage and sharing services for users.

B. DESIGN GOALS

SecDedup is designed with following objectives.

- **Functionality.** The CSP should perform deduplication on encrypted data without any additional online server, and it should be able to dynamically update data ownership. The scheme should be suitable for both file and block oriented storage systems.
- **Security.** The following security enhancements should be achieved: i). Data and encryption keys should be protected, i.e., only legitimate users can access data in the cloud. ii). Cloud data in the CSP can be accessed only if a user provides a valid access license.
- **Efficiency.** The proposed scheme should be time and storage efficient; additional security enhancement should not cause significant overhead.

C. ADVERSARY MODEL

We assume that the CSP is honest but curious, i.e., it correctly performs the system operations, but will try to explore data content [23]. To some extent, the interests of users and CSP conflict with each other, as users require efficiency while the CSP always seeks ways to reduce storage costs. So, we do not consider the collusion between users and the CSP. Side channel attacks also fall out of the scope of this paper, because they occur primarily in the client-side deduplication scenario. The adversary models are summarized as follows:

1) Malicious CSP

The CSP is capable of performing all interactions with users according to the designed protocols. Since it can access or copy the encrypted data at any time, the check-tags are vulnerable to offline brute-force attacks from the CSP. The attacks are as follows:

a. Guess all possible combinations of data content.

b. Construct check-tags and compare them with the user's check-tag to verify the the guess.

2) Malicious members

We assume that $\mathcal{A} \in \{U_i\}(i \in [1, n])$ is a member of legitimate users and he is able to interact with the KDC and get a valid key pair. In addition, the CSP is always honest in performing all system protocols. It cannot tell whether $\mathcal{A}$ is malicious or not. It is obvious that $\mathcal{A}$ holds more key elements than the CSP does, which may pose serious threat to encrypted data.

The offline brute-force attack and the online brute-force attack are two common attack forms of malicious members. As check-tags can be regarded as unique identifiers of data, in an offline brute-force attack, $\mathcal{A}$ enumerates the data and creates the relevant check-tags. Then he verifies his guess by comparing the check-tags with the original one. In an online brute-force attack, $\mathcal{A}$ sends the enumerated data to the CSP, responses from the CSP are used to determine whether some data already exists.

In addition, the forward attack and the backward attack are also potential attacks of a malicious member:

a. **Forward attack.** $\mathcal{A}$ attempts to access the original data after they are deleted or modified by himself;

b. **Backward attack.** $\mathcal{A}$ tries to access the data before he obtains the ownership.

IV. SECDEDUP SCHEME

A. PRELIMINARIES

1) Hash Function and Short Hash

- A **hash function** transforms an input(also known as pre-image) of any length to a fixed-length output with a hashing algorithm. This transformation is a compression mapping; the space of the hash value is usually much smaller than the input space. Different inputs may yield same output. However, for secure cryptographic hash functions, it is extremely difficult to find a collision, that is, the hash values of different inputs are most likely different. It is computational infeasible to calculate the original data from the hash, but an adversary can perform an offline brute-force attack on the hash. Given enough time and computational power, an adversary can enumerate all possible content of a target data and compare the hash values with the actual hash value to verify the guesses.
- A **short hash** can be obtained by truncating the result of any hash function (e.g., SHA-1, SHA-2, MD5). For example, we can truncate the top 10-20 bits of a hash to get a short hash.It is much more difficult to perform offline brute-force attacks on a short hash. That's because different data may have identical short hash, the collision rate is relatively high.

2) Proxy Re-Encryption(PRE)

A secure PRE scheme usually consists of three types of entities, namely the grantor, the grantees and the proxy. PRE contains the following algorithms:

- $KeyGen(\lambda) \rightarrow (\langle pk, sk\rangle)$. This algorithm is used to generate the user's public and private key pair, which is run by the grantor. Its input is a secure parameter λ, and the output is the user's key pair.
- $Encrypt(pk, m) \rightarrow (C_m)$. This algorithm is run by the grantor to encrypt plaintext m into ciphertext C_m. Its input includes user's public key pk and plaintext m, the output is ciphertext C_m.
- $ReKeyGen(sk_i, sk_j) \rightarrow (rk_{i\rightarrow j})$. This algorithm is used to generate the re-encryption key, which is run by the grantor. Its input includes the grantor's private key sk_i and grantee's private key sk_j, the output is re-encryption key $rk_{i\rightarrow j}$.
- $ReEncrypt(C_m, rk_{i\rightarrow j}) \rightarrow (C_m^{'})$. This algorithm is used to re-encrypt ciphertext C_m into $C_m^{'}$, which is run by the proxy. Its input includes the initial ciphertext C_m and re-encryption key $rk_{i\rightarrow j}$, the output is re-encryption ciphertext $C_m^{'}$.
- $Decrypt(sk_j, C_m^{'}) \rightarrow (m)$. This algorithm is used to decrypt the re-encryption ciphertext $C_m^{'}$, which is run by the grantee. Its input includes the re-encryption ciphertext $C_m^{'}$ and the grantee's private key sk_j, the output is plaintext m.

3) Bilinear Mapping

$(G_1, \times)$ and $(G_2, \times)$ are two multiplicative cyclic groups of prime order, $e : G_1 \times G_1 \rightarrow G_2$ is a Bilinear Mapping if it has the following attributes [24], [25]:

- $Bilinear$. For all $g_1, g_2 \in G_1$ and $a, b \in Z_p^*$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{a\cdot b}$, p is the order of G_1.
- $Computability$. For all $g_1, g_2 \in G_1$, there are valid algorithms that can calculate $e(g_1, g_2)$.
- $Non\text{-}degeneracy$. Exists $g_1, g_2 \in G_1$, which make $e(g_1, g_2) \neq 1$.

4) Proof of Ownership

The Proof of Ownership (PoW) technique is designed to address a range of security issues that may exist with deduplication systems [28], [29]. When a user is informed by the cloud server that data F has already been uploaded by other users. He needs to go through a secure authentication protocol to gain access to F. In short, PoW is used to verify whether a user really holds the data rather than some meta information(e.g. the hash $H(F)$).

B. SCHEME OVERVIEW

SecDedup is composed of four polynomial time algorithms, as shown in Figure 2.

$$(SystemSet, CheckTagGen, SecureDedup,$$
$$UpdateOwnership).$$

Here is a high-level description of the four algorithms.

- First, the KDC executes $SystemSet$ to initialize the system.
- Then, an uploader runs $CheckTagGen$ with the CSP to generate a duplicate check-tag, with which, the CSP is able to check independently whether the data copy is duplicated.
- After that, the uploader can obtain the encryption key regardless of data redundancy, then employ the key to encrypt the data and upload the encrypted data to the CSP. On the server side, if the data copy is duplicated, the CSP performs deduplication. Otherwise, it stores the data.
- Finally, the system will perform $UpdateOwnership$ when any uploader deletes or modifies some cloud data.

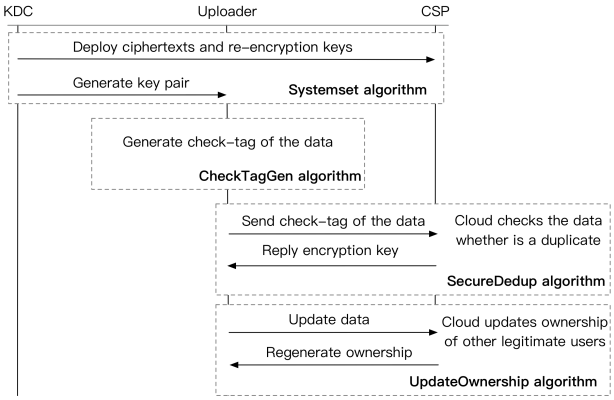

FIGURE 2: The overview of SecDedup

C. ALGORITHM FORMULATION

Symbols and their definitions used in this paper are listed in Table 1.

SecDedup consists of four polynomial time algorithms:

- $SystemSet(\lambda, ID_i, r_i) \rightarrow (\langle pk, sk\rangle, rk_{KDC\rightarrow i}, C_{r_i})$. This algorithm is performed by the KDC. It takes a secure parameter λ, a user's ID_i, a random value r_i as input, and outputs the user's key pair $\langle pk, sk\rangle$, the re-encryption key $rk_{KDC\rightarrow i}$ and the ciphertext of the random value C_{r_i}.
- $CheckTagGen(p, H, x) \rightarrow (\sigma)$. A check-tag generation algorithm run by data uploaders. Its input includes a large prime number p, the hash value of some data H and the ciphertext parameter x. It outputs the check-tag σ.
- $SecureDedup(\sigma, C_F) \rightarrow (0|1)$. A deterministic algorithm run by the CSP to verify whether the encrypted data have been stored. The inputs are the check-tag σ and the encrypted data C_F; it outputs 0 (first copy) or 1 (duplication). The deduplication operation will be performed if this algorithm outputs 1.
- $UpdateOwnership(v_1) \rightarrow (v_1')$. This algorithm is performed by the CSP to update the ownership of data.

TABLE 1: Symbol Definition

Symbol	Definition
ID	user's identification
sk_i/pk_i	user's private/public key in proxy re-encryption
sk_{CSP}/pk_{CSP}	CSP's private/public key in ownership update
sk_{KDC}/pk_{KDC}	KDC's private/public key in proxy re-encryption
$rk_{KDC\rightarrow i}$	re-encryption key
ssk/svk	KDC's signature key in proxy re-encryption
sh_i	short hash value of data m_i
Num	number of same short hashes in CSP
m_i	data to be uploaded
$\mathcal{A}$	a legitimate user
Z_p^*	integer field
G_1, G_2	two multiplicative cyclic group
g,h	generators of G_1
e	bilinear mapping
$E()/D()$	symmetric encryption / decryption algorithm
$Enc()/Dec()$	proxy re-encryption / decryption algorithm
r_i	random parameter generated by KDC
C_{r_i}	the ciphertext of r_i
SH	short hash function
$H/H_1/H_2$	cryptographic hash function
OL	data ownership list
v	access license
σ	duplicate check-tag
κ	data encryption key
C_κ	ciphertext of data encryption key

The input and output are the original ownership v_1 and the new ownership v_1' respectively.

1) $SystemSet$

As shown in Figure 3, in process A, the KDC executes $SystemSet$ to provide key pairs for users. Then it sends the re-encryption key set and ciphertext parameter set into the CSP.In process B, after performing the above operations, the KDC could go offline.

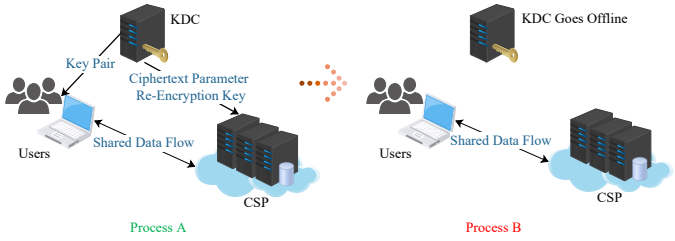

FIGURE 3: SystemSet

Detailed description of $SystemSet$ is as follows:

Step 1. Choose a random $x_i \in Z_p^*$, define $sk_i = x_i, pk_i = g_i^x$. Then send $< pk_i, sk_i >$ to user $U_i (i \in [1, n])$. It is worth noting that every user needs to register with the KDC to obtain the key pair before using the system.

Step 2. Select $x_{KDC} \in Z_p^*$ as the private key sk_{KDC}. Compute re-encryption keys $\{rk_{KDC\rightarrow i}\} \leftarrow \{(sk_{KDC}/sk_i)\ mod\ p\}(i \in [1, n])$, deploy them into the CSP.

Step 3. Set $pk_{KDC} = g^{sk_{KDC}}$. Then, employ pseudo-random number generator to generate parameters $\{r_i\}$ and $\{\kappa_i\}(i \in [1, n])$. After that, encrypt $\{r_i\}$

and $\{\kappa_i\}$ into $\{C_{r_i}\}$ and $\{C_{\kappa_i}\}$ with $pk_{KDC}(i \in [1,n])$; detailed encryption process is given in **Algorithm 1**. Finally, deploy random number ciphertext pool $\{C_{r_i}\}$ and random key ciphertext pool $\{C_{\kappa_i}\}$ into the CSP. Here, $\{C_{r_i}\}$ and $\{C_{\kappa_i}\}$ are used to generate redundant tags and encryption keys for data to be uploaded, respectively.

Algorithm 1 The encryption algorithm in PRE

Input: $\{r_i\}, \{\kappa_i\}(i \in [1,n]), pk_{KDC}$
Output: $\{C_{r_i}\}, \{C_{\kappa_i}\}(i \in [1,n])$
choose a signature key pair as $< svk, ssk >$;
randomly choose $t_i \in Z_p^*$;
select two hash functions $H_1, H_2 : \{0,1\}^* \rightarrow G_1$;
for$(i = 0; i < n; i++)$
  compute $V \Leftarrow (pk_{KDC})^{t_i}$;
  compute $W \Leftarrow e(g, H_1(svk))^{t_i} \cdot r_i$;
  compute $B \Leftarrow e(g, H_1(svk))^{t_i} \cdot \kappa_i$;
  compute $X \Leftarrow H_2(svk)^{t_i}$;
  compute $Y \Leftarrow h^{t_i}$; // h is a generator of G_1
  adopt ssk to compute signature Sig for (W, B, X, Y);
  define $C_{r_i} = (svk, V, W, X, Y, Sig)$;
  define $C_{\kappa_i} = (svk, V, B, X, Y, Sig)$;
output $\{C_{r_i}\}, \{C_{\kappa_i}\}(i \in [1,n])$;

2) *CheckTagGen*

The purpose of this algorithm is for an uploader to generate duplicate check-tags,which can be used to determine whether a file or block is duplicate or not. Uploader U_i sends short hash $sh_i = SH(m_i)$ to the CSP, then computes the check-tag σ_i as follows:

Case 1. If there are Num short hashes in the CSP that are equal to sh_i, $\{sh_j = sh_i\}(j \in [1, Num])$:

the CSP re-encrypts $\{C_{r_j}\}(j \in [1, Num])$ into $\{C'_{r_j}\}(j \in [1, Num])$ with $rk_{KDC \rightarrow i}$, the re-encryption algorithm is given in **Algorithm 2**, where C_{r_j} is associated with sh_j.

Algorithm 2 re-encryption algorithm

Input: $\{C_{r_j}\}(j \in [1, Num]), rk_{KDC \rightarrow i}$
Output: $\{C'_{r_j}\}(j \in [1, Num])$
for$(j = 0; j < Num; j++)$
  set $V' \Leftarrow V^{rk_{KDC \rightarrow i}} = g^{(x_{KDC} \cdot t_i) \cdot (\frac{x_i}{x_{KDC}})} = g^{x_i \cdot t_i}$;
  set $C'_{r_j} \Leftarrow (svk, V', W, X, Y, Sig)$;
output $\{C'_{r_j}\}(j \in [1, Num])$;

As shown in Figure 4, all values linked to sh_j are stored in a table by the CSP, and all associated values can be found by short hash sh_j.

Then, the CSP sends $\{C'_{r_j}\}(j \in [1, Num])$ to U_i. A check-tag set $\{\sigma_i\}(i \in [1, Num])$ is computed by U_i via **Algorithm 3**.

Case 2. If no short hash in the CSP equals sh_i:

The CSP randomly chooses $C_{r_i} \in \{C_{r_j}\}(j \in [1,n])$, then re-encrypts C_{r_i} into C'_{r_i} with $rk_{KDC \rightarrow i}$, sends C'_{r_i} to U_i. The check-tag σ_i is computed by U_i via **Algorithm 4**.

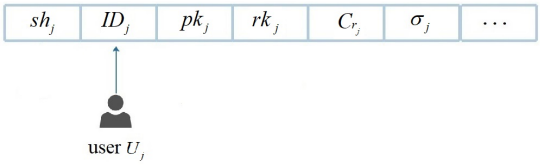

FIGURE 4: Associated value storage form in CheckTagGen

Algorithm 3 duplicate check-tag generation algorithm A

Input: $\{C'_{r_j}s\}(j \in [1, Num]), m_i$
Output: $\{\sigma_i\}(i \in [1, Num])$
for$(j = 0; j < Num; j++)$
  decrypt C'_{r_j} into r_j with sk_i
    $r_j \Leftarrow \frac{W}{e(V', H(svk))^{\frac{1}{sk_i}}} = \frac{e(g, H_1(svk))^{t_i \cdot r_i}}{e(g^{(x_i \cdot t_i)}, H(svk))^{\frac{1}{x_i}}}$;
  set $r_i = r_j$;
  compute $e(g^{sh_i}, H(m_i)^{r_i})$;
  define $\sigma_i \Leftarrow e(g^{sh_i}, H(m_i)^{r_i})$;
output $\{\sigma_i\}(i \in [1, Num])$;

The differences between Case 1 and Case 2 are as follows:

- In Case 1, there are multiple short hashes in the CSP that are the same as the short hashes of the data to be uploaded, which means the data may have been stored by other users. That is, the random parameter ciphertext value used in calculating the check-tags comes from users who hold the same short hash. Therefore, the output in Algorithm 3 is a check-tag set.
- In Case 2, there is no short hash in the CSP that is the same as the short hash of the data to be uploaded, which means the data is new. Therefore, the random parameter ciphertext value used in calculating the check-tags is randomly selected from the random number ciphertext pool by the CSP. That is why Algorithm 4 has only one output value.

3) *SecureDedup*

U_i sends check-tag set to the CSP. The CSP performs a query to determine whether an element in the set already exists.

Case 1. If $\exists \sigma_i \in \{\sigma_i\}(i \in [1, Num])$, which has been stored in the cloud server, then the ciphertext of m_i has been stored. According to the description in the previous subsection , if $sh_i = sh_j, r_i = r_j$. As $\sigma = e(g^s h, H(m)^r) = e(g^s h, H(m))^r$, when $\sigma_i = \sigma_j, m_i = m_j$, which means that m_i has already been stored in the cloud server by U_j. (e.g. $m_i = m_j$,m_j is uploaded by the initial uploader U_j), that is, U_i is a subsequent uploader of $m_i(m_j)$. The following operations are performed:

Step 1. The CSP re-encrypts C_{κ_j} into C'_{κ_j} with $rk_{KDC \rightarrow i}$ (subsequent uploader) via encryption algorithm in Proxy Re-Encryption (**Algorithm 2**), where C_{κ_j} denotes the ciphertext form of $\kappa_j \in Z_p^*$, and it is linked to U_j. $\kappa_j \in Z_p^*$ is the encryption key of m_j.
Step 2. The CSP sends C'_{κ_j} to U_i.

Algorithm 4 duplicate check-tag generation algorithm B

Input: C'_{r_i}, m_i
Output: σ_i
1: decrypt C'_{r_i} into r_i with sk_i;
2: $r_j \Leftarrow \frac{W}{e(V',H(svk))^{\frac{1}{sk_i}}} = \frac{e(g,H_1(svk))^{t_i \cdot r_i}}{e(g^{(x_i \cdot t_i))},H(svk))^{\frac{1}{x_i}}}$;
3: compute $e(g^{sh_i}, H(m_i)^{r_i})$;
4: define $\sigma_i \Leftarrow e(g^{sh_i}, H(m_i)^{r_i})$;
5: **output** σ_i;

Step 3. U_i decrypts C_{κ_j} into κ_j with sk_i, the decryption method is explained in **Algorithm 3** (or **Algorithm 4**).
Step 4. U_i sets $\kappa_i = \kappa_j$.
Step 5. U_i encrypts m_i into C_{m_i} with κ_i, i.e., $C_{m_i} = E(\kappa_i, m_i)$.
Step 6. U_i uploads $(\sigma_i, ID_i, rk_i, C_{\kappa_i}, H_3(C_{m_i}), C_{m_i})$ to the CSP, where $C_{\kappa_i} = C'_{\kappa_j}$.

Since $C_{m_i} = E(\kappa_i, m_i) = E(\kappa_j, m_j) = C_{m_j}$, the CSP deletes C_{m_i}, and adds (ID_i, σ_i) to OL_j (the ownership list of m_j).

As shown in Figure 5, all values linked to σ_i are stored in a table by the CSP, and all associated values can be found by check-tag σ_i. It is worth noting that since C_{m_i} is deleted, the encrypted data linked to σ_i is C_{m_j}.

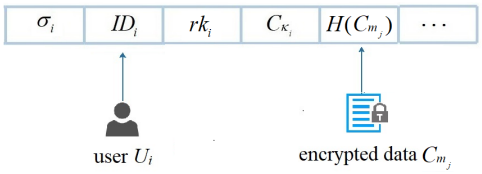

FIGURE 5: Storage structure of associated value

Case 2. If $sh_i (sh_i = sh_t)$ exists in the CSP while $\forall \sigma_i \in \{\sigma_i\} (i \in [1, Num])$ can not be found, then m_i is new, that is, U_i is the initial uploader of m_i. The following operations are performed:

Step 1. The CSP re-encrypts C_{κ_t} into C'_{κ_t} with $rk_{KDC \to i}$, via encryption algorithm in Proxy Re-Encryption, where C_{κ_t} is randomly selected from the random key ciphertext pool.

step 2-6 are the same as in Case 1.

Since m_i is new, the CSP stores $(ID_i, \sigma_i, C_{r_t}, E(\kappa_i, m_i))$, creates an Ownership List OL_i for m_i and adds (ID_i, σ_i) to it.

Case 3. If neither sh_i nor σ_i exists in the CSP, then m_i is new, that is, U_i is the initial uploader of m_i. The CSP performs the same operations as Case 2.

4) *UpdateOwnership*

We define data access license (data ownership proof) as v. In order to reduce the user's computational overhead, we use the encrypted duplicate check-tag, such like $\sigma_1^{pk_{CSP}}$, as the access license, where pk_{CSP} is the public key of the CSP. As shown in Figure 6, each short hash sh_i is associated with multiple access licenses. Similarly, each access license v_i is associated with multiple users.Then, we state that only users with v_i can access the data associated with v_i.

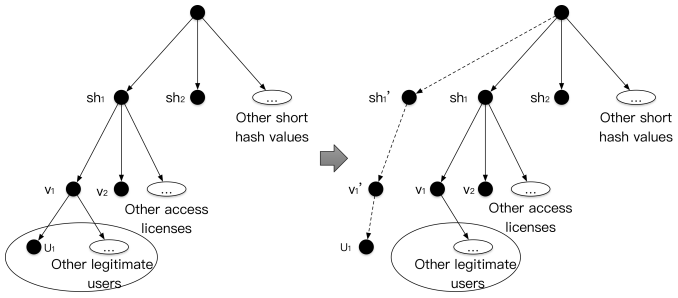

FIGURE 6: Dynamic update of data ownership

The CSP will delete ID_1 from the Ownership List of m_1 when m_1 is modified or deleted by U_1. If U_1 modifies m_1 to m'_1, he needs to run $CheckTagGen$ and $SecureDedup$ again with the CSP to obtain access license v'_1 of m'_1. In addition, the CSP needs to perform the following operations to ensure the security of the system.

step 1. Generate a new key pair $< pk_{CSP_{new}}, sk_{CSP_{new}} >$.
step 2. Compute access license update value $UV = \frac{pk_{CSP_{new}}}{pk_{CSP}}$.
step 3. Send UV to legitimate users other than U_1.

After that, legitimate users may conduct the following calculation to compute the new access license v'_1:

$$\begin{aligned} v'_1 &= v_1^{UV} \\ &= \sigma_1^{UV} \\ &= (e(g^{sh_1}, H(m_1)^{r_1})^{pk_{CSP}})^{UV} \\ &= (e(g^{sh_1}, H(m_1)^{r_1})^{pk_{CSP}})^{\frac{pk_{CSP_{new}}}{pk_{CSP}}} \\ &= e(g^{sh_1}, H(m_1)^{r_1})^{pk_{CSP_{new}}} \end{aligned}$$

Then they can employ the new access licensev'_1 instead of v_1 when trying to access m_1, and the CSP validates whether the equation $(v'_1)^{sk_{new}} = \sigma_1$ holds to determine the validity of the access license:

- If the equation holds, the CSP creates access links for this user.
- Otherwise, the CSP rejects the access request.

V. SECURITY ANALYSIS

In this section, we first combine the adversary models to examine the security of SecDedup. Then, the correctness and uniqueness of the check-tag are analysed.

A. SECURITY OF CHECK-TAGS

In the encrypted data deduplication scenarios,check-tags are most likely misused by an adversary. This is because check-tags are derived from the original data, an adversary can easily perform offline brute-force attacks on them. Therefore, it is a key issue to protect the security of the check-tags.

In convergence encryption(CE) based schemes, the check-tag is usually computed as follows:

$check - tag = H(Enc(H(m), m))$

If the structure or domain of source data $m \in DS$ is predictable, check-tags can be easily enumerated. By comparing the enumerated check-tags with the target, an offline brute-force attack can be carried out. There are schemes relying on a online trusted third party to calculate the check-tags. Secure as they are, it is difficult to implement a fully trusted online third party in real-world applications. SecDedup accomplishes protection against offline brute-force attacks without introducing any online trusted third party. The proof of safety is given below.

Assume that the check-tag of a current uploader U_i is $\sigma_i = e(g^{sh_i}, H(m_i)^{r_i})$. The CSP tends to run offline brute-force attack on σ_i with the following steps:

Step 1. Enumerate the data and construct set $\{m_t\}(t \in [1, z])$, where z is the number of all possible enumeration of the target data.

Step 2. Compute all the elements in set $\{e(g^{sh_t}, H(m_t))\}(t \in [1, z])$.

Step 3. Enumerate random values $r_i \in \{r_i\}(s \in [1, n])$ and obtain set $\{e(g^{sh_t}, H(m_t)^{r_i})\}(i \in [1, n], t \in [1, z])$ by bilinear mapping.

Step 4. Compare each element in $\{e(g^{sh_r}, H(m_r)^{r_i})\}$ with σ_i and try to find a match.

If the CSP finds a match, data privacy is compromised. However, this attack is invalid in SecDedup, the deduction is as follows:

Lemma 1. G_1 is a multiplicative cyclic group with order p. Given g, g^a, $h \in G_1$, where $a \in Z_p^*$, it is difficult to compute $h^a \in G_1$ (CDH problem) [31].

Lemma 2. From $CheckTagGen$, the CSP cannot decrypt C_{r_j}, i.e., it cannot get x_j in any case.

Proposition 1. The CSP cannot run any offline brute-force attacks on σ_i.

Proof. Without losing generality. To prove the security of the system, we assume that the information entropy of data m_i is extremely low, that is, m_i can be guessed(e.g., a document containing only a 6-digit password).

It is indicated by $SystemSet$ that $r_i(i \in [1, n])$ is randomly generated by the KDC. Therefore, it is computationally infeasible to enumerate r_i . Besides, even if the CSP has correct $\{r_i\}$, it still cannot get useful information. This is because, according to **Lemma 1** and **Lemma 2**, even if the CSP has $e(g, H(m_i))$ and $e(g^{sh_i}, H(m_i)^{r_i})$, it still cannot compute $\{e(g^{sh_i}, H(m_i)^{r_i})\}$. Therefore, the CSP cannot perform step 3 and step 4 in the attack described above; it cannot run any offline brute-force attacks on σ_i.

B. SECURITY OF ENCRYPTED DATA

Both malicious users and the malicious CSP can pose security threats to the encrypted data.

Case 1. If the adversary is a malicious user $\mathcal{A}$. His attack is composed of the following steps:

Step 1. Enumerate data set $\{m_t\}(t \in [1, z])$ and random value set $\{r_i\}(i \in [1, n])$w,here z is the number of all possible enumeration of the target data.

Step 2. Compute check-tag set $\{\sigma_{t_i} = e(g^{sh_t}, H(m_t))^{r_i}\}(t \in [1, z], i \in [1, n])$.

Step 3. Upload each σ_{t_i} to the CSP to obtain the encryption key κ_{t_i}.

Step 4. Encrypt each $m_t(t \in [1, z])$ into $C_{m_{t_i}}(t \in [1, z], i \in [1, n])$ with κ_{t_i}.

Step 5. Send each $C_{m_{t_i}}$ to the CSP to observe whether deduplication occurs.

Proposition 2. $\mathcal{A}$ cannot conduct online brute-force attack on encrypted data.

Proof. First, since we adopt cloud-side deduplication in $SecureDedup$, $\mathcal{A}$ cannot observe whether some data already exist by side channel attacks.

Besides, according to $SecureDedup$, $i)$ if some enumerated guess $C_{m_{t_i}}$ is correct, which means that there exists some m_j equal to $C_{m_{t_i}}$ in the CSP, the CSP returns $C'_{\kappa_j} = E(pk_{\mathcal{A}}, \kappa_j)$ to $\mathcal{A}$, and $ii)$ if there is no duplicated data while there exists some short hash match $sh_q = sh_t$, the CSP returns $C'_{\kappa_q} = E(pk_{\mathcal{A}}, \kappa_q)$ to $\mathcal{A}$.

$\mathcal{A}$ cannot tell the difference between the two cases. C'_{κ_j} and C'_{κ_q} are generated by the KDC via the same pseudo-random number generator. Hence $\mathcal{A}$ cannot take online brute-force attack on encrypted data.

Note: If there is no short hash match in the CSP, $\mathcal{A}$ is able to know that the current guess m_{t_i} is incorrect. Even so, $\mathcal{A}$ cannot get any useful information from it. Moreover, since short hash function has a high collision rate, this case is unlikely to occur after a large amount of data are stored in the CSP.

Case 2. Assume that the adversary is the malicious CSP.

Proposition 3. The CSP cannot conduct any attack on the encrypted data.

Analysis. Since data have been encrypted with users' encryption keys, the CSP cannot decrypt the data without the keys. Therefore, the CSP has to run offline brute-force attack on the encrypted data if he wants to get some useful information. According to **Proposition 1**, however, the CSP cannot conduct offline brute-force attack on encrypted data.

C. ANTI-COUNTERFEITING OF ACCESS LICENSE

Proposition 4. The probability that the malicious user $\mathcal{A}$ can forge the access license is negligible, i.e., $i)$ the probability that $\mathcal{A}$ can calculate v_i, the access license of m_i, is negligible if he does not have m_i ($\mathcal{A}$ cannot forge v_i). $ii)$ after $\mathcal{A}$ modifies or deletes m_i, the probability that he can access the original data is negligible ($\mathcal{A}$ cannot forge v'_i).

Proof. Since the initial value of access license v_i is the check-tag of m_i, which $\mathcal{A}$ does not have. It can be deduced from **Proposition 2.** that $i)$ is correct. Besides, according to $UpdateOwnership$, the CSP will send the access license update value to legitimate users (except for $\mathcal{A}$) if m_i is modified or deleted by $\mathcal{A}$. That is, $\mathcal{A}$ cannot access m_i anymore because he cannot update v_i to v'_i. Therefore, $ii)$ is correct.

D. CORRECTNESS OF THE CHECK-TAGS

Assume that the initial uploader U_j of data m_j has stored the check-tag $e(g^{sh_j}, H(m_j)^{r_j})$ in the CSP, and $e(g^{sh_i}, H(m_i)^{r_i})$ designates the check-tag calculated by a current uploader U_i.

Lemma 3. The secure hash function $H()$ has a deterministic result, i.e., if $m_i = m_j$, then $H(m_i) = H(m_j)$ [32].

Proposition 5. If $m_i = m_j$, the probability of $e(g^{sh_j}, H(m_j)^{r_j}) \neq e(g^{sh_i}, H(m_i)^{r_i})$ is negligible, i.e.,

$$Prob[\sigma_j \neq \sigma_i | m_i = m_j] < \varepsilon$$

Where ε indicates a negligible value.

Proof. Without losing generality.

a. According to $CheckTagGen$ and $SecureDedup$, if $sh_i = sh_j$, then $r_i = r_j$.

b. The following equation is obtained with the properties of bilinear mapping:

$$e(g^{sh_i}, H(m_i)^{r_i}) = e(g, H(m_i))^{sh_i \cdot r_i} \quad (1)$$

$$e(g^{sh_j}, H(m_j)^{r_j}) = e(g, H(m_j))^{sh_j \cdot r_j} \quad (2)$$

c. From **Lemma 3**, if $m_i = m_j$, then $H(m_i) = H(m_j)$.

d. According to a, b and c, the following equation can be obtained:

$$e(g^{sh_j}, H(m_j)^{r_j}) = e(g^{sh_i}, H(m_i)^{r_i}) \quad (3)$$

The correctness of **Proposition 5** is proved.

E. THE UNIQUENESS OF CHECK-TAG

Proposition 6. If $e(g^{sh_j}, H(m_j)^{r_j}) = e(g^{sh_i}, H(m_i)^{r_i})$, the probability of $m_i \neq m_j$ is negligible, i.e.,

$$Prob[m_i \neq m_j | \sigma_j = \sigma_i] < \varepsilon$$

Proof. Without losing generality.

a. According to $CheckTagGen$ and $SecureDedup$, if $sh_i = sh_j$, then $r_i = r_j$.

b. The following equation is obtained by the properties of bilinear mapping:

$$e(g^{sh_i}, H(m_i)^{r_i}) = e(g, H(m_i))^{sh_i \cdot r_i} \quad (4)$$

$$e(g^{sh_j}, H(m_j)^{r_j}) = e(g, H(m_j))^{sh_j \cdot r_j} \quad (5)$$

Also, we can draw the following conclusions. If the values in formula (5) equal those in formula (6), and $sh_i = sh_j \wedge r_i = r_j$, then $H(m_i) = H(m_j)$, i.e., $m_i = m_j$. The correctness of **Proposition 6** is proved.

F. SECURITY OF PROXY RE-ENCRYPTION

The security of proxy re-encryption(PRE) is not the focus of our research, however, PRE meets the security requirement of our scheme.

If the Decisional Bilinear Diffie-Hellman (DBDH) assumption holds in $(\mathbb{G}, \mathbb{G}_T)$, where $\mathbb{G}$, $\mathbb{G}_T$ represents two groups generated by some generaters g and y, and there is a bilinear map $\mathbb{G}_T \leftarrow \mathbb{G} \times \mathbb{G}$, then the PRE scheme meets Chosen-Ciphertext Attack security for $\mathbb{G}_T$ of messages in the standard model. The detailed proof of the DBDH security assumption was provided in reference [36], we do not elaborate on it in our work.

G. SECURITY COMPARISON WITH OTHER SCHEMES

We compare our scheme with others on the following two criterions. 1. Whether the encryption key is derived from the plaintext of the data or not. 2. Whether the system have to rely on a online trust third party. For the CE and MLE scheme, the encryption keys are all derived from the plaintext of the data being protected. In our scheme, the encryption key is neither derived from the plaintext nor associated with any plaintext information of the data. This feature makes our scheme more secure than CE and MLE. There are other secure deduplication schemes such as DupLess, Cloudedup and PerfectDedup, in which the encryption keys also have no direct connection with the plaintext. However, as far as the trusted third party is concerned, those schemes all need an online trusted third party to function properly. In comparison, our scheme not only satisfy the first criterion, but also the second one, which means that it has a obvious superiority over existing schemes.

VI. PERFORMANCE EVALUATION

In the experiments, the system is implemented using C ++ with GMP [33], PBC [34] and OPENSSL [35] libraries. The experiments are run on a Linux server with 2.7 GHz CPU, 8 GB memory and 1 TB hard disk. The bandwidth is limited to 1 Mbps to simulate real-world scenarios. The elliptic curve base field is set to 512 bits, and the size of the element in the field is 160 bits. Approximately 15,600 files are stored in the server. The experiments consist of three parts:

1. Randomly select and upload file F_1 (e.g., a 65MB file), record the time span for each phase of SecDedup and compare them with other schemes.

2. Upload files of different sizes:

a. Measure the time spent by our scheme and Convergent Encryption(CE) scheme in the encryption algorithm (including encryption key obtaining and symmetric encryption).

b. Measure the time spent by our scheme and scheme [9] in updating data ownership.

3. Randomly select and upload file F_2 (e.g., a 10MB file), evaluate the total time overhead of SecDedup and compare it with other schemes. All the three parts of the experiments are repeated 10 times and the average result is acquired.

In addition, we analyze communication overhead of SecDedup.

A. THE TIME SPAN FOR EACH PHASE

Experimental results are shown in Figure 7. The user-side only needs to perform lightweight operations such as file segmentation, check-tag generation and symmetric encryption. Intensive computing tasks are outsourced to the server, including check-tag verification and key delivery.

Whether a check-tag exists in the CSP is the key to identify duplication. The time span to generate and verify the tag in different schemes is shown in Figure 8. Experiment results show that our scheme has distinct advantage.

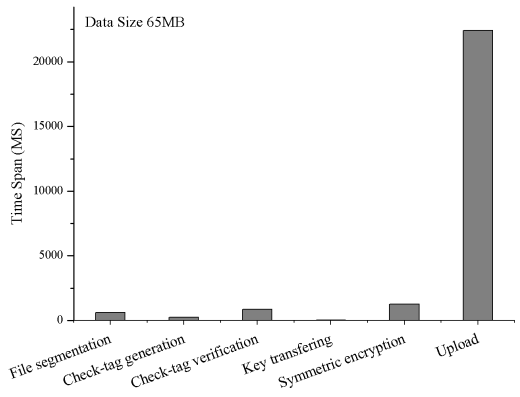

FIGURE 7: The time span for each phase

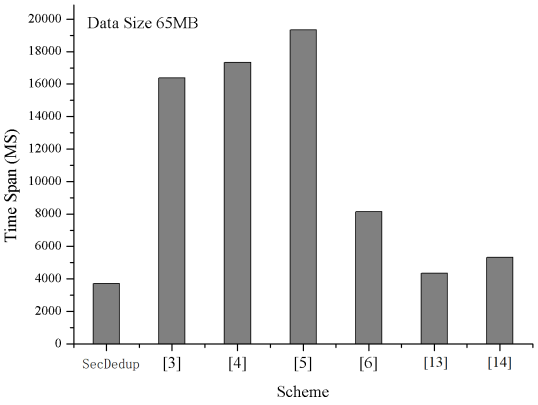

FIGURE 8: comparison of time overhead for check-tag generation and varification

B. TIME SPAN OF DATA ENCRYPTION AND OWNERSHIP UPDATING

In order to meet the requirements of semantic security, the secure key delivery algorithm is necessary, which inevitably increases the system overhead and communication costs. To assess its impact on the overall performance, we use data of different sizes and measure the time span of the encryption operation in SecDedup and Convergent Encryption(CE) respectively (denoted by $t(s)$ and $t(c)$). We define D-value $dv = t(s) - t(c)$ as the difference, and define specific value $sv = \frac{dv}{t(c)} = \frac{t(s)-t(c)}{t(c)}$ to indicate the relative difference with respect to the total time span. As shown in Figure 9, our scheme spends more time on encryption operation than CE does. However, the difference is minor, and it is limited within a range regardless of the data size. As data size increases, the difference turns negligible if compared with the total time span.

The CSP is able to dynamically update data ownership, which improves the security of the system. The scheme of

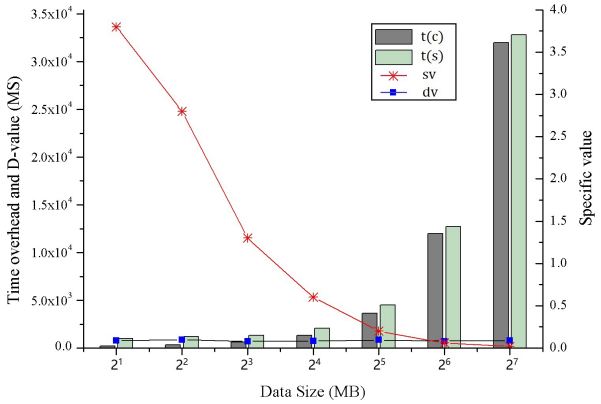

FIGURE 9: The time span of the two encryption algorithms

J.Hur et al. [21] provided a method to address this issue, but it needs to re-encrypt the data and update the encryption key, which significantly undermines efficiency. Figure 10 shows the time spans for updating data ownership in the scheme of J.Hur et al. and our scheme. According to the experiment results, the computational costs of our scheme are lower, and the difference between the two is more obvious as data size increases.

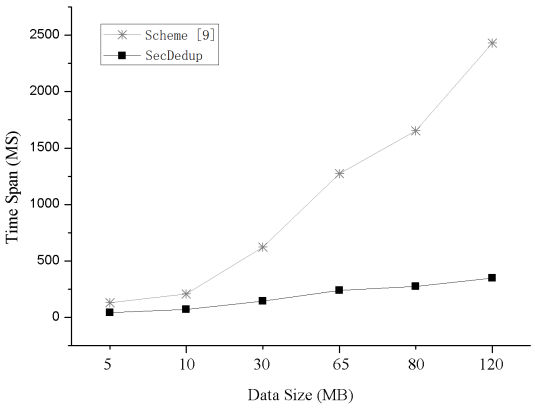

FIGURE 10: Time span of the two schemes in updating data ownership

C. TOTAL TIME OVERHEAD

As opposed to existing schemes [3]–[6], the CSP does not need to interact with a third party in our scheme, hence the communication overhead in our scheme is lower, which makes it more efficient. Results are shown in Figure 11.

D. COMMUNICATION OVERHEAD

In SecDedup, the communication overhead is mainly caused by $CheckTagGen$, $SecureDedup$ and $UpdateOwnership$.

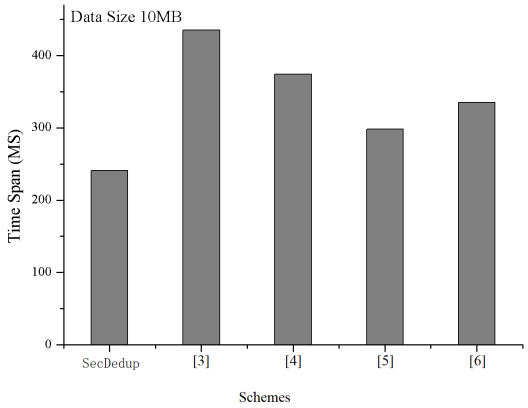

FIGURE 11: Total time span of each scheme

1) Communication Overhead in $CheckTagGen$

In $CheckTagGen$, communication overhead lies in two stages:

1. A User sends a short hash sh to the CSP.

2. The CSP replies ciphertext parameters $C_{r_i}(i \in [1, K])$ to the user.

sh is obtained by truncating the output of a secure hash function , so the length $|sh|$ is fixed. The length of the output of proxy re-encryption is also fixed. The length of one ciphertext parameter is $K \cdot \sum_{n=1}^{6} L_n$ bits, where L_1 and L_2 represent the length of elements(pubic key) in ECC; L_3 represents the length of elements in integer field Z_p^*; L_4 represents the length of the result of hash function H_2; L_5 represents the length of the generator element of G_1; L_6 represents the length of ECC based signature. Therefore, the total communication overhead of $CheckTagGen$ is $(|sh| + K \cdot \sum_{n=1}^{6} L_n)$ bits.

2) Communication Overhead in $SecureDedup$

In $SecureDedup$, communication overhead lies in two stages:

1. A user sends K check-tags $\{\sigma_i\}(i \in [1, K])$ to the CSP.

2. The CSP replies an encrypted key C_{κ_i} to the user.

Therefore, the total communication overhead of $SecureDedup$ is $(K + 1) \cdot L_3$ bits. L_3 represents the length of encrypted element in Z_p^*.

3) Communication Overhead in $UpdateOwnership$

In $UpdateOwnership$, the ownership update value UV consumes $n \cdot L_1$ bits, where n represents the number of users who need to update UV.

4) Comparison

We compare our scheme with other existing solutions using the three metrics described above.

Similar to our scheme, BdDedup [20] employs ECC cipher algorithm to generate redundancy tags, and adopts proxy re-encryption cryptographic primitives to perform encrypted data deduplication. As shown in Table 2, BdDedup is more efficient than SecDedup in the first two metrics, but the redundancy tag generation does not satisfy semantic security. When the entropy of the data is low, the calculation results are predictable.

In terms of ownership updates, DomDedup [21] applies the Merkel tree to build the key management system for users to transform data and update data ownership. Although the communication overhead is low during the update process, it requires a series of complicated calculations such as re-encryption, resulting in high computational overhead. To the best of our knowledge, only FogDedup [22] minimizes communication overhead while ensuring high efficiency. However, it can only be achieved with the assistance of a fog storage system. Unlike the above schemes, SecDedup uses bilinear mapping to construct the ownership update algorithm, which achieves the same level of communication overhead as FogDedup without introducing high computational overhead.

TABLE 2: Comparison of communication overhead

Schemes	$CheckTagGen$	$SecureDedup$	$UpdateOwnership$
SecDedup	$\|sh\| + K \cdot \sum_{n=1}^{6} L_n$	$(K+1) \cdot L_3$	$n \cdot L_1$
BdDedup	$L_1 + L_3 + L_4$	$4 \cdot L_3$	$--$
FogDedup	L_4	$--$	$n \cdot L_1$
DomDedup	L_4	$--$	$(n-m)log\frac{n}{n-m} \cdot L_3$

VII. CONCLUSION

In this paper, we propose a deduplication scheme for encrypted data, named SecDedup. It does not relay on any online trusted third party, and it allows dynamic ownership updating. We use cryptographic primitives such as proxy re-encryption and bilinear mapping instead of convergent encryption. Compared with previous schemes, the security of SecDedup is significantly enhanced. In our design, the CSP can work as an intermediary, which focuses on three core functionalities, in particular, **1)** updating the ownership list when a user updates or deletes his data, **2)** assisting an initial uploader to validate subsequent uploaders, and **3)** delivering data encryption keys in an offline manner. The optimized CSP in SecDedup can easily and effectively perform deduplication on encrypted data. Meanwhile, it is enforced that a user is allowed to access the data only if he can provide a valid access license. Simulation experiments show that our scheme is applicable and efficient.

REFERENCES

[1] X.Yang, R. Lu, J. Shao, et al. "Achieving efficient and privacy-preserving multi-domain big data deduplication in cloud",IEEE Transactions on Services Computing,, 2018.

[2] M.Bellare, S.Keelveedhi, and T.Ristenpart, "Message-Locked Encryption and Secure Deduplication", Advances in Cryptology Eurocrypt 2013, Springer Berlin Heidelberg, 2013, pp.296-312.

[3] M.Bellare, S.Keelveedhi, and T.Ristenpart, "DupLESS: server-aided encryption for deduplicated storage", In Proceedings of the 22nd Usenix conference on Security, Usenix Association, 2013, pp.179-194.

[4] P.Puzio, R.Molva and S.Loureiro, "Cloudedup: Secure deduplication with encrypted data for cloud storage", In IEEE International Conference on Cloud Computing Technology and Science, 2013, pp.363-370.

[5] J.Stanek, A.Sorniotti, E.Androulaki, et al, "A secure data deduplication scheme for cloud storage", Ibm Corporation, 2014, PP.99-118.
[6] P.Puzio, R.Molva and M.Önen, "PerfectDedup: Secure Data Deduplication", International Workshop on Data Privacy Management, Springer International Publishing, 2015, PP.150-166.
[7] C.Hui, H.D.Robert and L.Yingjiu, et al,"Attribute-Based Storage Supporting Secure Deduplication of Encrypted Data in Cloud", IEEE Transactions on Big Data, 2016.
[8] X.R.Ge, J.Yu, H.L Zhang, C.Y.Hu, Z.P.Li, Z.Qin, and R.Hao,"Towards Achieving Keyword Search over Dynamic Encrypted Cloud Data with Symmetric-Key based Verification",IEEE Transactions on Dependable and Secure Computing, 2019.
[9] B.Libert, and D.Vergnaud, "Unidirectional Chosen-Ciphertext Secure Proxy Re-Encryption",IEEE Transactions on Information Theory, 2011, pp. 1786-1802.
[10] R.Bellafqira, G.Coatrieux and D.Bouslimi,"Proxy Re-Encryption Based on Homomorphic Encryption", Computer Security Applications Conference ACM, 2017, pp. 154-161.
[11] Z.Li, C.Ma, and D.Wang,"Achieving Multi-Hop PRE via Branching Program", IEEE Transactions on Cloud Computing, 2018, doi=10.1109/TCC.2017.2764082.
[12] M.Blaze, G.Bleumer and M.Strauss,"Divertible protocols and atomic proxy cryptography", in Proc. EUROCRYPT 1998, pp.127-144.
[13] Y. Zhou, D. Feng, Y. Hua, et al.,"A similarity-aware encrypted deduplication scheme with flexible access control in the cloud", Future Generation Computer Systems, 2018, 84, pp. 177-189.
[14] B. T. Reddy, T. Rao, "Filter Based Data Deduplication in Cloud Storage using Dynamic Perfect Hash Functions", International Journal of Simulation Systems, Science and Technology, 2018.
[15] J.Li, X.Chen and M.Li, et al, "Secure deduplication with efficient and reliable convergent key management", IEEE transactions on parallel and distributed systems, 2014, 25(6): pp.1615-1625.
[16] Y.Zhang, J.Yu, R.Hao, C.Wang and K.Ren,"Enabling Efficient User Revocation in Identity-based Cloud Storage Auditing for Shared Big Data", IEEE Transactions on Dependable and Secure Computing, 2018. DOI: 10.1109/TDSC.2018.2829880
[17] Y.L He, H.Q. Xian, et al, "Secure Encrypted Data Deduplication Based on Data Popularity",Mobile Networks and Applications,2020. DOI: 10.1007/s11036-019-01504-3.
[18] H.Y.Hou, J.Yu and R.Hao, "Cloud storage auditing with deduplication supporting different security levels according to data popularity", Journal of Network and Computer Applications, 2019, 134:pp. 26-39.
[19] J.Yu and H.Q.Wang, Strong Key-Exposure Resilient Auditing for Secure Cloud Storage, IEEE Transactions on Information Forensics and Security, 2017, 12(8): pp.1931-1940.
[20] Z.Yan , W.Ding and X.Yu, et al, "Deduplication on Encrypted Big Data in Cloud", IEEE Transactions on Big Data, 2016, 2(2):PP.138-150.
[21] J.Hur, D.Koo and Y.Shin, "Secure Data Deduplication with Dynamic Ownership Management in Cloud Storage", IEEE Transactions on knowledge and data engineering, 2016, 28(11): pp.3113-3125.
[22] D.Koo, and J.Hur, "Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing", Future Generation Computer Systems, 2018, 7(8): pp. 739-752.
[23] J. Xiong, Y. Zhang, S. Tang, et al., "Secure Encrypted Data With Authorized Deduplication in Cloud", IEEE Access, 2019, 7, pp. 75090-75104.
[24] D.Boneh, C.Gentry and D.Lynn, et al, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps", Lecture Notes in Computer Science 2003, pp.416-432.
[25] R.T. Hou, H.Q. Xian, "A Graded Reversible Watermarking Scheme for Relational Data", Mobile Networks and Applications, 2019, https://doi.org/10.1007/s11036-019-01491-5
[26] R.RCanetti and R.Hohenberger, "Chosen-ciphertext secure proxy re-encryption", Proceedings of the 14th ACM conference on Computer and communications security, ACM, 2007, pp.185-194.
[27] Y. Fan, X. Lin, W. Liang, et al., "A secure privacy preserving deduplication scheme for cloud computing", Future Generation Computer Systems, 2019.
[28] R.B.Sirsat, and N.R.Talhar, "Deduplication in cloud storage on the basis of proof of ownership", International Conference on Computing Communication Control and Automation IEEE, 2017, pp.1-5.
[29] L.G.Manzano, J.M.D.Fuentes and K.K.R.Choo, "ase-PoW: A Proof of Ownership Mechanism for Cloud Deduplication in Hierarchical Environments", Security and Privacy in Communication Networks, 2017.
[30] L. Wang, B. Wang, W. Song, et al., "A key-sharing based secure deduplication scheme in cloud storage", Information Sciences, 2019, pp. 48-60.
[31] J.C.Herzog, "The Diffie-Hellman Key-Agreement Scheme in the Strand-Space Model", Computer Security Foundations Workshop, 2003, pp.234-247.
[32] I.B.Damgard, "A Design Principle for Hash Functions", Conference on the Theory and Application of Cryptology Springer, 1989, pp.416-427.
[33] M.Loukides, A.Oram, "Programming with GNU software", O'Reilly and Associates, 1997.
[34] Y.Li, G.Q.Xu, H.Q.Xian, et al., "Novel Android Malware Detection Method Based on Multi-dimensional Hybrid Features Extraction and Analysis", Intelligent Automation And Soft Computing, 2019,25(3): p-p.637ÍC647,DOI: 10.31209/2019.100000118
[35] X.Ting, "Research and Improved Implementation of AES Algorithm in OpenSSL", Microcomputer Information, 2009, 25(12): pp.83-85.
[36] R.Canetti, S.Hohenberger, "Chosen-ciphertext secure proxy re-encryption", Acm Conference on Computer and Communications Security, 2007.
[37] X.Yang, R.Lu, J.Shao, et al. "Achieving Efficient Secure Deduplication with User-Defined Access Control in Cloud",IEEE Transactions on Dependable and Secure Computing, 2020, PP(99):1-1.
[38] H.Yuan, X.Chen, J.Wang, et al. "Blockchain-based Public Auditing and Secure Deduplication with Fair Arbitration", Information Sciences, 2020.

SHUGUANG ZHANG is currently a Master degree candidate in college of Computer Science and Technology, Qingdao University, China. His research interests include cloud security and big data security.

HEQUN XIAN received Ph.D degree in the Institute of Software, Chinese Academy of Sciences in 2009. He was a visiting scholar with college of information science and technology, the Pennsylvania State University. His research interests include cryptography, cloud computing security, and network security.

ZENGPENG LI received his PhD degree from Harbin Engineering University. He is currently a Lecturer of Qingdao University. He has worked on lattice based cryptography, password based cryptography, and cryptographic protocol.

LIMING WANG received Ph.D degree in the Institute of Software, Chinese Academy of Sciences in 2003. He is currently a Professor of Institute of Information Engineering, Chinese Academy of Sciences. His research interests include cryptography, cloud computing security, and network security.