This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2987793, IEEE Transactions on Dependable and Secure Computing

Achieving Efficient Secure Deduplication with User-Defined Access Control in Cloud

Xue Yang, Rongxing Lu, *Senior Member, IEEE,* Jun Shao, Xiaohu Tang, *Member, IEEE,* and Ali A. Ghorbani, *Senior Member, IEEE*

Abstract—Cloud storage as one of the most important services of cloud computing significantly facilitates cloud users to outsource their data to the cloud for storage and share them with authorized users. In cloud storage, secure deduplication has been widely investigated as it can eliminate the redundancy over the encrypted data to reduce storage space and communication overhead. Regarding the security and privacy, many existing secure deduplication schemes generally focus on achieving the following properties: data confidentiality, tag consistency, access control and resistance to brute-force attacks. However, as far as we know, none of them can achieve these four requirements at the same time. To overcome this shortcoming, in this paper, we propose an efficient secure deduplication scheme that supports user-defined access control. Specifically, by allowing only the cloud service provider to authorize data access on behalf of data owners, our scheme can maximally eliminate duplicates without violating the security and privacy of cloud users. Detailed security analysis shows that our authorized secure deduplication scheme achieves data confidentiality and tag consistency while resisting brute-force attacks. Furthermore, extensive simulations demonstrate that our scheme outperforms the existing competing schemes, in terms of computational, communication and storage overheads as well as the effectiveness of deduplication.

Index Terms—Secure deduplication, access control, authorized deduplication system, tag consistency, brute-force attacks.

1 Introduction

With the great benefits of cloud computing, an increasing amount of data have been outsourced by data owners to the cloud and shared with authorized users. For example, the Cisco global cloud index shows that the data stored in cloud will nearly reach 1.3 zettabytes by 2021 [1]. As a result, the management of the ever-increasing data becomes a critical challenge for cloud storage services. In fact, the study shows that about 75% of digital data are identical [2], and redundancy in backup and archival storage system is significantly more than 90% [3]. In this situation, *data deduplication* technique [4] has been widely developed in cloud storage because it can significantly reduce storage costs by storing only a single copy of redundant data. Indeed, data deduplication can reduce storage costs by more than 50% in standard file systems and by more than 90% for backup applications, and these savings are transformed into huge financial savings to cloud service providers and users [5].

However, considering security and privacy concerns of outsourced data, users are likely to encrypt data with their keys before outsourcing. As a result, data deduplication would be impeded as an identical data will be encrypted into different ciphertexts. To make data deduplication feasible on encrypted data, convergent encryption and its implementations or variants have been developed in [6]–[8]. However, convergent encryption suffers from brute-force attacks for predictable messages. To overcome this issue, some server-aided encryption schemes [9]–[11] have been presented. Unfortunately, they suffer from the duplicate faking attack [12] that prevents legitimate users from obtaining correct data. In more details, the adversary generates the ciphertext and the corresponding tag from different data m^* and m, respectively. Once the cloud service provider stores the inconsistent ciphertext and tag, subsequent users who upload the tag corresponding to m can only obtain the incorrect data m^*. Although some schemes [13]–[15] try to resist this attack, the tag consistency is verified after downloading the ciphertext by users, which cannot exactly conclude whether the incorrect data is caused by duplicate faking attacks in data upload or is corrupted during data storage. The main reason for this attack is that the ciphertext and the corresponding tag are generated independently, which makes it impossible for the cloud service provider to check the tag consistency. Thus, a solution that computes the tag directly by hashing the ciphertext is presented [12], which obviously supports the tag consistency check by comparing the hash of ciphertext to the received tag. Based on this idea, tag consistency and message authentication have been considered in [16], where the cloud service provider checks tag consistency and users conduct the message authentication after data download.

As one of the essential components of cloud computing, access control has been widely used in practical cloud products. Thus, it is an inevitable trend to develop authorized secure deduplication. However, as far as we know, only a few schemes [17]–[19] deal with authorized secure deduplication by introducing an additional authorized server or the hybrid cloud architecture, but they suffer from either duplicate faking attacks or brute-force attacks. As far as we know, none of existing schemes can achieve data confidentiality, tag consistency, access control and resistance to brute-force attacks at the same time. To overcome this challenge, in this paper, we design an efficient secure cross-user deduplication scheme with user-

- *X. Yang and X. Tang are with the Information Security and National Computing Grid Laboratory, Southwest Jiaotong University, Chengdu, China, 610031, and X. Yang is also with the Canadian Institute of Cybersecurity, Faculty of Computer Science, University of New Brunswick, Fredericton, Canada, E3B 5A3 (e-mail: xueyang.swjtu@gmail.com, xhutang@swjtu.edu.cn).*
- *R. Lu and A. Ghorbani are with the Canadian Institute of Cybersecurity, Faculty of Computer Science, University of New Brunswick, Fredericton, Canada, E3B 5A3 (e-mail: rlu1@unb.ca, ghorbani@unb.ca).*
- *J. Shao is with the School of Computer and Information Engineering, Zhejiang Gongshang University, Zhejiang, China, 310018, and also with the Canadian Institute of Cybersecurity, Faculty of Computer Science, University of New Brunswick, Fredericton, Canada, E3B 5A3 (e-mail: chn.junshao@gmail.com).*
- *Corresponding author: R. Lu (e-mail: rlu1@unb.ca).*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2987793, IEEE Transactions on Dependable and Secure Computing

defined access control, which mainly includes the following four contributions:

- Considering that many access settings over the cloud are based on user identity, we set user identity as an example attribute rather than complexity attributes. In order to resist the brute-force attack, our scheme allows users to randomly select the symmetric key instead of directly hashing the data itself. Meanwhile, the tag consistency check can be supported by directly generating the tag from hashing the symmetric ciphertext.
- In order to support access control, our scheme skillfully combines the Composite-order Bilinear Pairing technique [20] and Boneh-Goh-Nissim cryptosystem [21] to design an efficient proxy re-encryption algorithm, which achieves that only the cloud service provider can authorize data access on behalf of the data owner without the privacy breach. Meanwhile, our design can supervise the CSP's behavior since only the CSP owns a private key to complete the re-encryption operation. That is, even if the CSP is not completely trusted, the CSP has to follow each user's authorization set to perform re-encryption operations to maintain good reputation.
- Detailed security analyses demonstrate that our scheme is the first to achieve data confidentiality, resistance to brute-force attacks, tag consistency and access control at the same time. Compared to existing competing schemes, our scheme neither needs to introduce an extra server nor requires users to be constantly online during data upload phase.
- In order to reduce the time complexity of the duplicate search, we introduce the Bloom filter [22] to efficiently check the duplication. Extensive experiments demonstrate the efficiency of our scheme in terms of deduplication effectiveness, computational cost, communication overhead and storage cost.

Note that similar to [19], we set the granularity of access right to user-level in this paper. That is the data owner sets the specific access policy and sends it to the cloud service provider for checking the attributes of other users. If they satisfy with the policy defined by this data owner, the cloud service provider can help them to obtain the secret key of this data owner. For simplifying our presentation, we set user identity as an example attribute, e.g., the data owner U_1 allows other users with $ID = \{U_2, U_3, U_4\}$ to share its data storage. In fact, our scheme can be easily extended to support more fine-grained access right, e.g., file-level. More precisely, the cloud service provider can performs data deduplication only if the attributes of the file F satisfies the policy of the file F^*.

Next, we will introduce the models and design goals in Section 2, before recalling some necessary preliminaries in Section 3. Then, we present our scheme in Section 4, followed by its security analysis and performance evaluation in Sections 5 and 6, respectively. Related work is discussed in Section 7. We conclude this paper in Section 8.

2 Models and Design Goals

In this section, we describe an authorized deduplication model and the corresponding threat model used in this paper and identify our design goals.

2.1 System Model

Similar to many cloud (cloud computing)-related researches [19], [23], our authorized deduplication system comprises three types of entities: a key generation center, a cloud service provider and a number of users $\mathcal{U} = \{U_1, U_2, \ldots, U_w\}$. Figure 1 illustrates the system architecture, and the details of each entity are described as follows.

- Key generation center (KGC): The KGC is responsible for setting up the whole system at the beginning. Specifically, the KGC generates system parameters, the secret key of the cloud service provider, and public/private key pair of every user. After that, the KGC can keep offline unless a new user joins the system. Note that the KGC only needs to generate the public/private key pair for the newcomer.
- Cloud service provider (CSP): The CSP provides storage services for users. In the authorized deduplication system, without violating access control, the CSP would like to eliminate the redundant data and keep only one copy to reduce the overhead associated with data storage.
- Users: In our authorized deduplication system, users can be either data owners or authorized users. The data owner typically refers to the user who outsources encrypted data to the CSP and shares these outsourced data with authorized users. The authorized user generally refers to the user who is authorized by data owners to access their outsourced data.

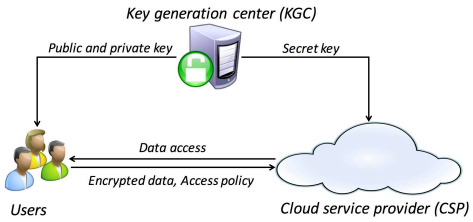


Fig. 1: System model under consideration.

2.2 Threat Model

Similar to most secure deduplication schemes [19], [24], [25], the KGC is assumed to be completely trusted and would not be compromised by any adversary. The CSP is assumed to be honest-but-curious, which means that it honestly follows the underlying scheme, but it may attempt to obtain as much secret information as possible based on its possessions. Users (corrupted by the adversary) are assumed to be malicious that they would try to access unauthorized data.

Similar to [15], [16], two kinds of adversaries are considered in our system: external adversary and internal adversary.

- External adversary: An external adversary is interested in the content of the data outsourced to the CSP, while it hasn't received any secrets from the KGC.
- Internal adversary: An internal adversary can be anyone who received a secret from the KGC and is interested in obtaining the content of unowned or unauthorized data. Specifically, based on some background knowledge of the message space, the CSP could launch offline brute-force attacks to determine which message corresponds to the specific encrypted data, while unauthorized users could launch online brute-force attacks to know whether a user of interest owns certain messages in the message space. Furthermore, any user (corrupted by the adversary) would launch the duplicate faking attack to prevent legitimate authorized users from obtaining the correct data.

In particular, as claimed in [19], [26], [27], we also hold the same assumption that the CSP would not collude with any user due to the

reputation of the CSP and the privacy concerns of users themselves. Obviously, any collusion would worse the reputation of the CSP, which lead to final loss of their business.

2.3 Design Goals

Based on the above threat model, we aim to achieve the following security goals in the proposed scheme.

- Data confidentiality: Any adversary including the CSP or unauthorized users cannot feasibly extract any useful information (i.e., data content) from the outsourced data.
- Resistance to brute-force attacks: Even though brute-force attacks can be launched with the background knowledge of message space, any adversary even corrupting the CSP or unauthorized users cannot obtain which content corresponds to the specific encrypted data or whether a user of interest owns some data.
- Access control: In our authorized deduplication system, the proposed scheme should ensure that outsourced encrypted data can be available not only to the data owner but also to all authorized users selected by this data owner. On the contrary, unauthorized users cannot access the content of encrypted data outsourced by users of interest.
- Tag consistency: After data deduplication, only one copy is stored in the CSP, and if this copy is inconsistent with the related tag, i.e., a malicious user (corrupted by an adversary) launches duplicate faking attacks during data upload, then the subsequent data owner together with authorized users cannot obtain the original data. Thus, the proposed scheme should provide the tag consistency check to ensure the uploaded ciphertext is consistent with the corresponding tag.
- Efficiency: In addition to security requirements, efficiency is also an indispensable metric of data deduplication, especially for practical applications [15], [28]. More specifically, under satisfying access rights, improving the effectiveness of deduplication is our design goal. Besides, achieving the efficiency of computation, communication, and storage is also our design goal.

3 Preliminaries

In this section, we outline the bilinear groups of composite order [20], Boneh-Goh-Nissim cryptosystem [21] and Bloom filter [22], which will serve as the basis of our scheme.

3.1 Bilinear Groups of Composite Order

Given a security parameter κ, a composite bilinear parameter generator $\mathcal{G}en(\kappa)$ outputs a tuple $(N, \mathbb{G}, \mathbb{G}_T, e)$, where $N = pq$, p and q are two κ-bit primes, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic multiplicative groups of composite order N, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map with the following properties:

- Bilinear: $e(x^a, y^b) = e(x, y)^{ab}$ for all $x, y \in \mathbb{G}$, and $a, b \in \mathbb{Z}_N$.
- Non-degeneracy: If g is a generator of $\mathbb{G}$, then $e(g, g)$ is a generator of $\mathbb{G}_T$ with the order N.
- Computability: For all $x, y \in \mathbb{G}$, there exists an efficient algorithm to compute $e(x, y) \in \mathbb{G}_T$.

Some related complexity assumptions are given below. For more comprehensive descriptions, refer to [29], [30].

Definition 1 (Discrete Logarithm (DL) Problem)**.** *The DL problem in $\mathbb{G}$ is stated as follows: given $x \in \mathbb{G}$, compute $a \in \mathbb{Z}_N$ such that $g^a = x$.*

Definition 2 (Divisible Decision Bilinear Diffie-Hellman (DDBDH) Assumption)**.** *The DDBDH assumption is that: given (g, g^a, g^b, W), for $g \in \mathbb{G}$, unknown $a, b \in \mathbb{Z}_N^2$ and $W \in \mathbb{G}_T$, no probabilistic, polynomial-time algorithm $\mathcal{B}$ can determine whether $W = e(g, g)^{a/b}$ or a random element from $\mathbb{G}_T$ with more than a negligible function $\mathtt{negl}(\kappa)$, i.e.,*

$$\begin{aligned}\mathrm{DDBDH} - \mathrm{Adv}_{\mathcal{B}} =& |\Pr[\mathcal{B}(g, g^a, g^b, W) = 1] - \\ & \Pr[\mathcal{B}(g, g^a, g^b, e(g, g)^{a/b}) = 1]| \\ \leq & \mathtt{negl}(\kappa).\end{aligned}$$

Definition 3 (Subgroup Decision (SD) Assumption)**.** *Let $(N = pq, \mathbb{G}, \mathbb{G}_T, e)$ be a tuple produced by $\mathcal{G}en(\kappa)$, where p and q are primes. Let g be a generator of $\mathbb{G}$, then $g_1 = g^q \in \mathbb{G}$ can generate the subgroup $\mathbb{G}_p = \{g_1^0, g_1^1, \ldots, g_1^{p-1}\}$ of order p in $\mathbb{G}$. The SD assumption is that: given the parameters $(N, \mathbb{G}, \mathbb{G}_T, e, y)$, where the element y is randomly drawn from either $\mathbb{G}$ or the subgroup $\mathbb{G}_p$, no probabilistic, polynomial-time algorithm $\mathcal{B}$ can decide whether an element y is in the subgroup $\mathbb{G}_p$ with more than a negligible function $\mathtt{negl}(\kappa)$, i.e.,*

$$\begin{aligned}\mathrm{SD} - \mathrm{Adv}_{\mathcal{B}} =& |\Pr[\mathcal{B}(N, \mathbb{G}, \mathbb{G}_T, e, y) = 1 : y \in \mathbb{G}] - \\ & \Pr[\mathcal{B}(N, \mathbb{G}, \mathbb{G}_T, e, y) = 1 : y \in \mathbb{G}_p]| \\ \leq & \mathtt{negl}(\kappa).\end{aligned}$$

3.2 Boneh-Goh-Nissim Cryptosystem

Boneh-Goh-Nissim (BGN) cryptosystem mainly contains three algorithms: key generation, encryption, and decryption.

- Key generation: Given a security parameter κ, run $\mathcal{G}en(\kappa)$ to get a tuple $(N = pq, \mathbb{G}, \mathbb{G}_T, e)$. Randomly choose two generators $g, x \in \mathbb{G}$ and set $y = x^q$. The public key is $pk = (N, \mathbb{G}, \mathbb{G}_T, e, g, y)$ and the corresponding private key is $sk = p$.
- Encryption: Given a message $m \in \{0, 1, \ldots, W\}$, where $W \ll q$, choose a random number $r \in \mathbb{Z}_N$, and compute the ciphertext as $C = g^m y^r \in \mathbb{G}$.
- Decryption: Given the ciphertext C and private key p, compute $C^p = (g^m y^r)^p = (g^p)^m$. Let $g' = g^p$, then $C^p = g'^m$. To obtain m, it suffices to compute the discrete logarithm of g'^m. Actually, since the message space is very small, say $W \ll q$, it is possible to compute the discrete logarithm by utilizing Pollard's lambda method with the expected time $O(\sqrt{W})$ [31].

Based on the above descriptions, the BGN decryption contains two steps: 1) eliminate the random number with the private key p, and 2) compute the discrete logarithm. However, when the message space is large, computing the discrete logarithm becomes a hard problem (see Definition 1), which means that the entity owning the private key can performs the first step to remove the effects of the random number, but cannot complete the second step to obtain the plaintext. This property exactly meets the requirements of our scheme that only the CSP with the private key can authorize data access on behalf of the data owner without threatening the privacy of users. Thus, we adopt the BGN cryptosystem in this paper, where the detailed introduction and analysis will be shown in Sections 4.2.3 and 5.1, respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2987793, IEEE Transactions on Dependable and Secure Computing

3.3 Bloom Filter

The Bloom filter (BF) is a space-efficient data structure for representing a set and testing whether an element is definitely not or possibly in this set. Specifically, a BF is initialized by $InitBF(\delta)$ to generate an array of δ bits, where all bits are set to 0 (see Fig. 2). The BF mainly contains two operations: element addition and membership query. In order to add an element or query whether an element is in the set, the BF chooses f independent hash functions $\{h_1, h_2, \ldots, h_f\}$, each of which uniformly maps the element to one of δ array positions, i.e., $h_i : \{0,1\}^* \rightarrow \{1, 2, \ldots, \delta\}$ for $i = 1, 2, \ldots, f$.

- Element addition $AddBF(x)$: To add an element x in a set, f array positions in bit array are computed as $\{h_1(x), \ldots, h_f(x)\}$. Then, set the $h_i(x)$-th bit in the array to 1. Note that a bit location can be set to 1 multiple times, but only the first change has an effect.
- Membership query $QueryBF(y)$: To query whether an element y is included in the set, check the value of the $h_i(y)$-th bit in the array for $i = 1, 2, \ldots, f$. The result of $QueryBF(y)$ is either 1 or 0. If any of the bits at f positions is 0, then return 0, which means that the element y is definitely not in the set. If all are 1, then return 1, which means that either the element y is in the set, or the bits have by chance been set to 1 during the addition of other elements, resulting in a false positive [32].

Particularly, if n elements have been added into the BF, and each element is mapped to the f positions with equal probability, the false positive probability can be calculated by

$$\mathcal{P} = \left(1 - (1 - 1/\delta)^{fn}\right)^f \approx \left(1 - e^{-fn/\delta}\right)^f, \qquad (1)$$

which can be minimized when $f = \frac{\delta}{n} \ln 2$. Figure 2 provides an example of BF.

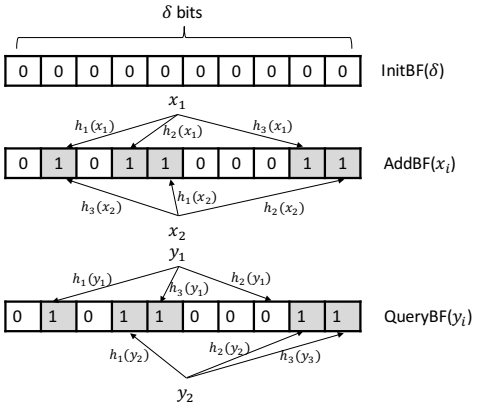


Fig. 2: An example of a BF initialized by $InitBF(10)$ and using 3 hash functions $\{h_1, h_2, h_3\}$.

4 Proposed Scheme

In this section, we present our scheme, which mainly includes three phases: system initialization, data upload and data download. Before that, we would like to give the description of notations used in the proposed scheme in Table 1.

TABLE 1: Notations used in the proposed scheme

Notation	Description
$\{U_1, \ldots, U_w\}$	w users in the system
$(N, \mathbb{G}, \mathbb{G}_T, e)$	parameters of the composite bilinear pairing
(mpk, msk)	master public and private keys
(pk_i, sk_i)	the public and private key pair of the user U_i
H_1, H_2	two cryptographic hash functions
δ	the size of the Bloom filter
$\{h_1, \ldots, h_f\}$	f hash functions in the Bloom filter
BF_i	the BF used to represent the dataset of U_i
$InitBF_i(\delta)$	the initialization operation for BF_i
$AddBF_i(\cdot)$	the element addition operation in BF_i
$QueryBF_i(\cdot)$	the element query operation in BF_i
$a_i \in \mathbb{G}_T$	the secret value selected by U_i randomly
C_{a_i}	the ciphertext of secret a_i generated by U_i
$C^j_{a_i}$	the re-encrypted ciphertext for U_j based on C_{a_i}
$\mathrm{SE}_{b_i}(\cdot)$	symmetric encryption algorithm with a key b_i
C^i_m	the ciphertext of the data m generated by U_i
T^i_m	the tag for checking the duplicate of the data m
τ^i_m	the token of the ciphertext C^i_m
$link^i_m$	the logical link to the data m outsourced by U_i
$O_i \subseteq \mathcal{U}$	the authorization set defined by U_i
$I_i \subseteq \mathcal{U}$	the access set authorized to U_i
$\mu_i \in \{0,1\}$	upload decision for duplicate lookup in U_i's dataset
$\mu \in \{0,1\}$	the final decision on whether to upload data

4.1 System Initialization

In our scheme, the system initialization mainly includes three parts: (1) KGC selects system parameters and generates private keys for users and the CSP; (2) each user sets his or her own authorization set and encrypts the selected symmetric key; (3) CSP initializes the Bloom filter and determines the access set of each user.

(1) KGC generates system parameters and the public and private key pair for each user as follows.

- Take a security parameter κ_0 as input, and output the tuple $(N = pq, \mathbb{G}, \mathbb{G}_T, e)$ by running the composite bilinear-parameter generation algorithm $\mathcal{G}en(\kappa_0)$.
- Randomly choose two generators $g, x \in \mathbb{G}$ and a number $\alpha \in \mathbb{Z}_N$, compute $g_1 = g^\alpha \in \mathbb{G}$. In addition, let $z = e(g, g)^p$ and $y = x^q$. The master public and private keys are $mpk = (g_1, z, y)$ and $msk = \alpha$, respectively.
- For each user $U_i \in \mathcal{U}$, generate the corresponding public and private key pair as $(pk_i, sk_i) = (g^{d_i \alpha^{-1}}, d_i)$, where $d_i \in \mathbb{Z}_N$ is a random number.
- Choose two cryptographic hash functions: $H_1 : \mathbb{G}_T \rightarrow \{0,1\}^{\kappa_1}$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{\kappa_1}$, where κ_1 is the bit length of the symmetric key.

Finally, the KGC keeps the master private key α secret, and sends d_i and p to U_i and the CSP by secure channels, respectively. The KGC publishes system parameters $(N, \mathbb{G}, \mathbb{G}_T, e, g_1, z, y, H_1, H_2, pk_1, pk_2, \cdots, pk_w)$.

Note that, as mentioned in Section 2.2, we assume that the KGC is completely trusted and would not be compromised by any attacker, so the security of our scheme can be guaranteed. Actually, we can enhance our scheme to improve reliability against the compromise of the KGC. Simply, our approach is to extend the system initialization so that a number of KGCs cooperate in generating the private keys. Specifically, we can achieve this goal by exploiting (t, n)-threshold secret sharing techniques, like Shamir (t, n)-threshold technique [33]. Based on the security of the (t, n)-threshold secret sharing technique, it is assured the security of our scheme unless at least t KGCs are compromised. Due to page limitations, we do not give the details here, but will discuss them in future work.

(2) Each user $U_i \in \mathcal{U}$ initializes the parameters as folllows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2987793, IEEE Transactions on Dependable and Secure Computing

Initialization				Data upload	
User	Authorization set	Access set	Key encapsulated ciphertext	Bloom filter	Data item
U_1	O_1	I_1	C_{a_1}	BF_1	$(\tau^1_{m_1}, link^1_{m_1}), \cdots, (\tau^1_{m_{n_1}}, link^1_{m_{n_1}})$
U_2	O_2	I_2	C_{a_2}	BF_2	$(\tau^2_{m_1}, link^2_{m_1}), \cdots, (\tau^2_{m_{n_2}}, link^2_{m_{n_2}})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
U_i	O_i	I_i	C_{a_i}	BF_i	$(\tau^i_{m_1}, link^i_{m_1}), \cdots, (\tau^i_{m_{n_i}}, link^i_{m_{n_i}})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
U_w	O_w	I_w	C_{a_w}	BF_w	$(\tau^w_{m_1}, link^w_{m_1}), \cdots, (\tau^w_{m_{n_w}}, link^w_{m_{n_w}})$

Fig. 3: Format of the outsourced data stored on the CSP.

- Define an authorization set $O_i \subseteq \mathcal{U}$ that all users in O_i can access the data uploaded by U_i. Naturally, U_i always has access to his or her own dataset, so $U_i \in O_i$.
- Randomly choose an element $a_i \in \mathbb{G}_T$ as a secret, and compute its ciphertext C_{a_i} as

$$C_{a_i} = (g_1^{\beta_i} y^{r_i}, a_i z^{\beta_i}) = (g^{\alpha\beta_i} y^{r_i}, a_i z^{\beta_i}), \qquad (2)$$

where $\beta_i, r_i \in \mathbb{Z}_N$ are two random numbers, $g_1 = g^\alpha$, and $g^{\alpha\beta_i} y^{r_i} \in \mathbb{G}$ is generated by the BGN encryption.

U_i stores $b_i = H_1(a_i)$ and sends (U_i, O_i, C_{a_i}) to the CSP.

(3) After receiving (U_i, O_i, C_{a_i}) from each user U_i, the CSP completes the following initialization operations.

- For each user $U_i \in \mathcal{U}$, initialize an empty access set I_i and check whether $U_i \in O_j$ for $j = 1, 2, \ldots, w$, and $i \neq j$. If $U_i \in O_j$, then add U_j to the set I_i, i.e.,

$$I_i = \{U_j | U_i \in O_j, for\ j = 1, 2, \ldots, w,\ and\ i \neq j\}.$$

That is, U_i has access to all users in the access set I_i.
- Set an appropriate value for δ and choose f independent hash functions $\{h_1, \ldots, h_f\}$, where $h_l : \{0,1\}^* \to \{1, 2, \ldots, \delta\}$ for $l = 1, \ldots, f$. For each user U_i, initialize the corresponding filter BF_i by executing $InitBF_i(\delta)$.

Finally, the CSP publishes f hash functions $\{h_1, \ldots, h_f\}$, and stores all tuples $(U_i, O_i, I_i, C_{a_i}, BF_i)$, for $i = 1, 2, \ldots, w$, as shown in Fig. 3.

In our scheme, the user revocation is equal to the change of the authorized set. Specifically, if the authorized set is changed, for example, U_i is deleted from the authorized set O_j, i.e., $U_i \notin O_j$, then this user losses the access to subsequent data uploaded by U_j. The simple method to achieve this goal is that after the change of the authorized set, in addition to informing the CSP about the new authorized set $O_j^* = O_j \backslash U_i$, the data owner U_j also needs to re-select the private key to encrypt the subsequent uploaded data. Based on the design of our scheme, the CSP would not help U_i to generate the re-encrypted ciphertext (see Eq. (4)), and thus U_i cannot obtain the re-selected key, which means that U_i can no longer decrypt the subsequent ciphertexts uploaded by U_j.

4.2 Data Upload

In this section, we present the details about our authorized deduplication scheme. Before that, we introduce the overall protocol of the authorized deduplication in Section 4.2.1.

4.2.1 Authorized deduplication protocol

In our authorized deduplication protocol, the deduplication mainly includes two parts: intra-deduplication and inter-deduplication.

Based on Algorithm 1, when each user U_i wants to upload the data m, the CSP firstly conducts the intra-deduplication in U_i's dataset (see lines 1-3). If any duplicate exists, the final decision μ is set to 0, which means that the data m does not need to be uploaded and the protocol can be terminated. Otherwise, the CSP further eliminates the redundancy by performing the inter-deduplication (see lines 5-24). Particularly, if U_i is not authorized by U_j, the CSP cannot eliminate the duplicate even if m is already stored in cloud. Therefore, the CSP only needs to check whether the duplicate exists in datasets that U_i can access (see line 5).

Algorithm 1 Authorized deduplication protocol

When U_i wants to upload the data, CSP performs the following steps to check the duplicate:

1: Check whether the duplicate exists in U_i's dataset.
2: **if** the duplicate exists **then**
3: **return** the final decision value $\mu = 0$.
4: **else**
5: **for** each user $U_j \in I_i$ **do**
6: Check the relationship between O_i and O_j.
7: **if** $O_i \subseteq O_j$ or $O_j \subset O_i$ **then**
8: Check whether the duplicate exists in U_j's dataset.
9: **if** the duplicate exists **then**
10: **if** $O_i \subseteq O_j$ **then**
11: Set $\mu_j = 0$.
12: **end if**
13: **if** $O_j \subset O_i$ **then**
14: Set $\mu_j = 1$ and delete the corresponding duplicate in U_j's dataset.
15: **end if**
16: **else**
17: Set $\mu_j = 1$.
18: **end if**
19: **else**
20: Set $\mu_j = 1$.
21: **end if**
22: **end for**
23: Compute the final decision value $\mu = \bigwedge_{U_j \in I_i} \mu_j$.
24: **return** μ.
25: **end if**

During the inter-deduplication, due to access control, the CSP determines whether it is necessary to check the duplicate in U_j's dataset according to the inclusion relationship of two authorization sets O_i and O_j (see line 6). Specifically, if O_i and O_j satisfy the inclusion relationship, i.e., $O_i \subseteq O_j$ or $O_j \subset O_i$, the CSP checks the duplicate in U_j's dataset (see lines 7-18). Further, if the duplicate exists, the CSP eliminates the duplicate between U_i and U_j's datasets. Meanwhile, to ensure access control for all users belonging to the set $O_i \cup O_j$, the CSP stores the unique copy in the dataset of the user whose authorization set is the superset.

- If $O_i \subseteq O_j$, then the unique copy needs to be stored in U_j's dataset. Thus, set $\mu_j = 0$ (see lines 10-12).
- If $O_j \subset O_i$, then the unique copy needs to be stored in U_i's dataset. Thus, set $\mu_j = 1$. Besides, the CSP deletes the found duplicate in U_j's dataset for eliminating the redundancy (see lines 13-15).

However, if $O_i \nsubseteq O_j$ and $O_j \nsubseteq O_i$, then the CSP cannot perform data deduplication. In general, we can find at least two users $U_k \in O_i$ and $U_l \in O_j$ such that $U_k \notin O_j$ and $U_l \notin O_i$, i.e., U_k cannot access U_j's dataset, and U_l cannot access U_i' dataset. Suppose the encrypted data of m is already stored in U_j's dataset, i.e., C_m^j, if U_i does not upload the encrypted data of m after the deduplication, U_i can access

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2987793, IEEE Transactions on Dependable and Secure Computing

C_m^j but U_k cannot, which violates access control. In this case, the CSP does not check the duplication in U_j' dataset and directly sets $\mu_j = 1$ (see lines 19-20).

After checking the duplicate in the datasets belonging to all users in I_i, the CSP obains the final decision μ on whether U_i needs to upload the encrypted data (see lines 23-24). More precisely, for all users in I_i, the encrypted data needs to be uploaded only when $\mu = 1$, i.e., $\mu_j = 1$ for each user $U_j \in I_i$.

4.2.2 Intra-deduplication

When U_i wants to upload m, U_i generates a tag T_m^i and sends it to the CSP to perform the intra-deduplication as follows.

Step 1: U_i generates T_m^i. U_i first encrypts the data m, and then generates T_m^i with the computed ciphertext as follows.

- Based on the symmetric key $b_i = H_1(a_i)$ (see Section 4.1), compute the ciphertext as

$$C_m^i = \mathrm{SE}_{b_i}(m), \tag{3}$$

  where $\mathrm{SE}_{b_i}(\cdot)$ represents the symmetric encryption algorithm with the symmetric key b_i, e.g., AES.
- With f hash functions $\{h_1, h_2, \ldots, h_f\}$, compute

$$T_m^i = \big(h_1(C_m^i), h_2(C_m^i), \ldots, h_f(C_m^i)\big).$$

- Send the tag T_m^i to the CSP for intra-deduplication.

Step 2: CSP performs intra-deduplication. The CSP determines whether a duplicate exists by executing $QueryBF_i(T_m^i)$.

- If $QueryBF_i(T_m^i) \to 1$, it means the duplicate exists. Thus, the CSP directly sets $\mu = 0$ and returns it to U_i.
- If $QueryBF_i(T_m^i) \to 0$, the CSP sets $link_m^i = \emptyset$, and conducts the following inter-deduplication.

4.2.3 Inter-deduplication

The CSP conducts the inter-deduplication to further check the duplicate in datasets uploaded by other users but accessible to U_i. More precisely, for each user $U_j \in I_i$, the CSP determines whether it is necessary to check the duplicate in U_j's dataset by checking the inclusion relationship between O_j and O_i.

Step 1: CSP checks O_j and O_i's inclusion relationship.

- If $O_i \subseteq O_j$ or $O_j \subset O_i$, the CSP re-encrypts U_j's key encapsulated ciphertext $C_{a_j} = (g_1^{\beta_j} y^{r_j}, a_j z^{\beta_j})$ with the secret parameter p and U_i's public key $pk_i = g^{d_i \alpha^{-1}}$:

$$e\big((g_1^{\beta_j} y^{r_j})^p, pk_i\big) = e(g,g)^{d_i \beta_j p} = z^{d_i \beta_j}, \tag{4}$$

  where $g_1 = g^\alpha$ and $z = e(g,g)^p$. The re-encrypted ciphertext for U_i is $C_{a_j}^i = (z^{d_i \beta_j}, a_j z^{\beta_j})$. Then, the CSP returns $C_{a_j}^i$ to the user U_i.
- Otherwise, the CSP directly sets $\mu_j = 1$. In this case, the next steps 2 and 3 will be skipped.

Step 2: U_i generates the tag T_m^j. If U_i receives $C_{a_j}^i$, U_i generates the tag used for checking the duplicate in U_j's dataset.

- Based on $C_{a_j}^i = (z^{d_i \beta_j}, a_j z^{\beta_j})$, U_i recovers the secret a_j selected by U_j with the private key $sk_i = d_i$ as

$$a_j = a_j \cdot z^{\beta_j} \big/ \big(z^{d_i \beta_j}\big)^{d_i^{-1}}, \tag{5}$$

  and computes the symmetric key $b_j = H_1(a_j)$.
- With the symmetric key b_j, U_i encrypts the data m as $C_m^j = \mathrm{SE}_{b_j}(m)$, and then computes the tag $T_m^j = \big(h_1(C_m^j), h_2(C_m^j), \ldots, h_f(C_m^j)\big)$.
- Then, U_i stores b_j locally, and sends T_m^j to the CSP.

Step 3: CSP checks the duplicate. Once the tag T_m^j is received, the CSP checks whether the duplicate exists in U_j's dataset by executing $QueryBF_j(T_m^j)$.

- If $QueryBF_j(T_m^j) \to 0$, then the duplicate does not exist in U_j's dataset. Thus, the CSP sets $\mu_j = 1$.
- If $QueryBF_j(T_m^j) \to 1$, then the duplicate exists in U_j's dataset. The CSP computes the token τ_m^j with T_m^j as

$$\tau_m^j = H_2\big(h_1(C_m^j) \| h_2(C_m^j) \| \cdots \| h_f(C_m^j)\big), \tag{6}$$

  which is used to locate the record of this duplicate. Then, the CSP finds the same token by comparing τ_m^j to previously stored tokens in U_j's dataset, e.g., $(\tau_{m_k}^j, link_{m_k}^j)$, which implies $m_k = m$. Based on *Step 1*, there are two cases to be discussed.
  - When $O_i \subseteq O_j$, the CSP sets $\mu_j = 0$ and $link_m^i = link_m^i \cup link_{m_k}^j$.
  - When $O_j \subset O_i$, the CSP sets $\mu_j = 1$.

Step 4: CSP returns the final decision. After obtaining the decision $\mu_j \in \{0,1\}$ for each $U_j \in I_i$, the CSP computes the final decision

$$\mu = \bigwedge_{U_j \in I_i} \mu_j. \tag{7}$$

Obviously, the value of μ is either 1 or 0, and $\mu = 1$ only if $\mu_j = 1$ for all $U_j \in I_i$. Finally, the CSP returns μ to the user U_i. Note that the CSP also returns $link_m^i$ when $\mu = 0$.

4.2.4 Data processing

After receiving μ, if $\mu = 0$, U_i does not upload C_m^i. In this case, the CSP also does not perform any update operations when the duplicate is found in intra-deduplication. If the duplicate is found in inter-deduplication, i.e., $\mu_j = 0$ for some user $U_j \in I_i$, with the previously received tag $T_m^i = \big(h_1(C_m^i), \ldots, h_f(C_m^i)\big)$ (see Section 4.2.2), the CSP updates the data record of U_i as follows:

- Update the Bloom filter BF_i by executing $AddBF_i(T_m^i)$.
- Compute a token $\tau_m^i = H_2\big(h_1(C_m^i) \| \cdots \| h_f(C_m^i)\big)$.
- Store $(\tau_m^i, link_m^i)$, where $link_m^i = link_m^i \cup link_{m_k}^j$ (see *Step 3* in Section 4.2.3).

When $\mu = 1$, U_i uploads the ciphertext C_m^i. In this case, the CSP checks the tag consistency after receiving C_m^i.

- With the previously received tag T_m^i, the CSP computes a token $\tau_m^i = H_2\big(h_1(C_m^i) \| \cdots \| h_f(C_m^i)\big)$.
- Based on the received ciphertext C_m^i, the CSP computes the corresponding verifiable token as

$$\tau_{verify} = H_2\big(h_1(C_m^i) \| h_2(C_m^i) \| \cdots \| h_f(C_m^i)\big).$$

- The CSP checks whether $\tau_m^i = \tau_{verify}$ holds. If it holds, the CSP updates the data record of U_i:
  - Update the BF_i by executing $AddBF_i(T_m^i)$.
  - Store the C_m^i in U_i's dataset and use $link_m^i$ to point the corresponding logical link.
  - Add $(\tau_m^i, link_m^i)$ to U_i's dataset, as shown in Fig. 3, and then return $link_m^i$ to U_i.

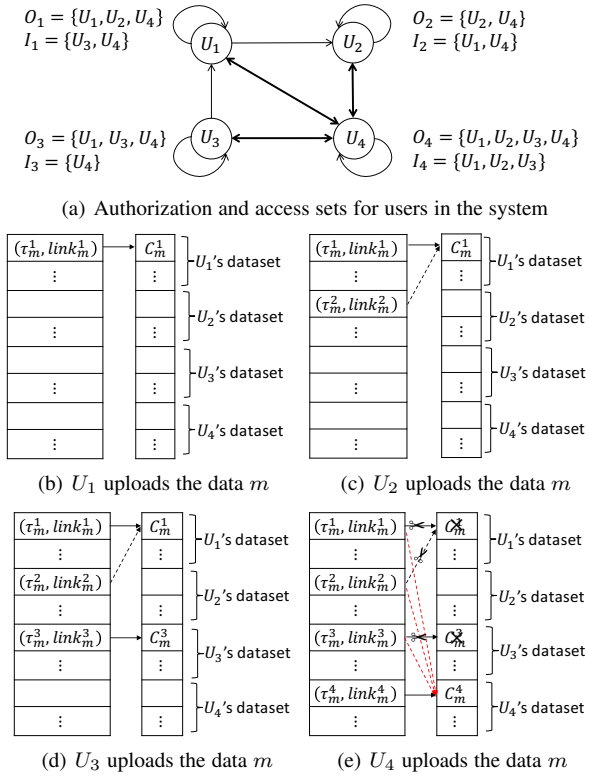
  If $\tau_m^i \neq \tau_{verify}$, then it implies the ciphertext C_m^i and tag T_m^i are inconsistent. Thus, the CSP directly drops the uploaded data and does not conduct any update operations.

Additionally, when tag consistency is satisfied, for the case that $QueryBF_j(T_m^j) \rightarrow 1$ and $\mu_j = 1$ (see *Step 3* in Section 4.2.3), the CSP checks whether $link_{m_k}^j$ points to U_j's dataset.

- If yes, the CSP deletes the corresponding encrypted data $C_{m_k}^j$ and sets $link_{m_k}^j = link_m^i$.
- If $link_{m_k}^j$ points to other user's dataset, e.g., $link_{m_k}^j = link_{m_k}^l$ and $j \neq l$, then the CSP updates $link_{m_k}^j = link_{m_k}^j \cup link_m^i$, and would not delete any data since the encrypted data of m is not stored in U_j's dataset.

Note that regardless of whether the duplicate is found in the inter-deduplication, the CSP will add $(\tau_m^i, link_m^i)$ in U_i's dataset and update BF_i by calling $AddBF_i(T_m^i)$. The effect of this update operation is when U_i wants to upload the same data m next time, the CSP can quickly find the duplicate in the intra-deduplication without requiring further inter-deduplication.

Finally, U_i deletes m and stores $(label_m, link_m^i)$, where $label_m$ is the label of m, e.g., the name or feature used to identify m. The reason for using $label_m$ is to make it easy for users to identify each data. Besides, U_i also stores the symmetric key b_i generated by himself and b_j authorized by each user $U_j \in I_i$. Note that in the authorized access scenario, the authorized users can certainly obtain the key of the data owner. That is, in our scheme, regardless of whether the upload request is sent, the CSP always helps U_i to obtain all symmetric keys of users in the access set I_i by computing Eqs. (4) and (5).



(a) Authorization and access sets for users in the system

(b) U_1 uploads the data m

(c) U_2 uploads the data m

(d) U_3 uploads the data m

(e) U_4 uploads the data m

Fig. 4: An example of our authorized deduplication scheme.

4.2.5 An example of data upload

To elaborate on our scheme, we give a simple example of data upload in Fig. 4. Suppose there are four users $\{U_1, U_2, U_3, U_4\}$ in the system, and the related authorization set O_i and access set I_i of each user U_i, $i = 1, 2, 3, 4$, are shown in Fig. 4(a). Without loss of generality, we suppose that each user's dataset is empty at the beginning, and U_1, U_2, U_3 and U_4 try to upload the data m in turn. Obviously, U_1 as an initial uploader can successfully upload the encrypted data C_m^1, as shown in Fig. 4(b).

Then, U_2 sends T_m^2 to request upload m. The CSP finds the duplicate does not exist in U_2's dataset during the intra-deduplication (see Section 4.2.2). In this case, the CSP needs to perform the inter-deduplication in datasets of U_1 and U_4 (see Section 4.2.3). Since $O_2 \subset O_1$ and $O_2 \subset O_4$, the CSP checks the duplicate in U_1 and U_4's datasets. Specifically, the CSP generates the re-encrypted ciphertextes $C_{a_1}^2$ and $C_{a_4}^2$, and sends them to U_2. After that, U_2 and the CSP conduct the *Step 2* and *Step 3*, respectively. Since $O_2 \subset O_1$ and the same data has already been stored in U_1's dataset, i.e., C_m^1, the CSP sets $\mu_1 = 0$ and $link_m^2 = link_m^1$. Although $O_2 \subset O_4$, there is no data stored in U_4's dataset, so the CSP directly sets $\mu_4 = 1$. Finally, the CSP computes the final decision $\mu = \mu_1 \wedge \mu_4 = 0$ and returns it to U_2. Meanwhile, the CSP updates U_2's data record by executing $AddBF_2(T_m^2)$ and adding $(\tau_m^2, link_m^2)$, as shown in Fig. 4(c). In this case, U_2 does not upload the encrypted data C_m^2.

Next, U_3 tries to upload m. Similarly, the CSP does not find the duplicate in U_3's dataset and needs to perform the inter-deduplication in U_4's dataset. Since there is no duplicate in U_4's dataset, $\mu_4 = 1$. Thus, the final decision is $\mu = \mu_4 = 1$, which means that U_3 needs to upload C_m^3 and the CSP updates U_3's dataset after passing tag consistency verification (see Fig. 4(d)).

After a while, when U_4 wants to upload m, the CSP first finds the duplicate does not exist in U_4's dataset. Then, the CSP performs the inter-deduplication. Since O_4 is the superset of O_1, O_2 and O_3, the CSP needs to check the duplicate in datasets of U_1, U_2 and U_3. Particularly, the CSP finds the duplicate in U_1's dataset, and then sets $\mu_1 = 1$ due to $O_1 \subset O_4$. Similarly, the CSP would set $\mu_2 = 1$ and $\mu_3 = 1$. At last, the CSP sends $\mu = \mu_1 \wedge \mu_2 \wedge \mu_3 = 1$ to U_4. Thus, U_4 needs to upload C_m^4. According to data processing in Section 4.2.4, if tag consistency is satisfied, the CSP deletes C_m^1 and C_m^3 in the datasets of U_1 and U_3, respectively. Meanwhile, the CSP sets $link_m^1 = link_m^4$ and $link_m^3 = link_m^4$, respectively. Since the $link_m^2$ points to other user's dataset, i.e., $link_m^2 = link_m^1$, the ciphertext is not actually stored in U_2's dataset. In this case, the CSP sets $link_m^2 = link_m^1 \cup link_m^4 = link_m^4$. Finally, under the assurance of access control, the CSP only stores one copy of the data m, as shown in Fig. 4(e).

4.3 Data Download

After a period of time, when U_i wants to download an outsourced data m, he or she uses $lable_m$ to find the stored link $link_m^i$, and then directly downloads the encrypted data based on $link_m^i$. Note that the $link_m^i$ points to either the dataset of U_i or the dataset of another user, like U_j. Thus, the downloaded data may be C_m^i or C_m^j.

- If the downloaded data is C_m^i, U_i decrypts it with his or her own symmetric key b_i.
- If the downloaded data is C_m^j, U_i decrypts it with the authorized symmetric key b_j.

Note that we do not consider the message authentication in this paper due to page limitations. In fact, this requirement can be easily achieved by using the message authentication code technique.

5 Security Analysis

In this section, we analyze the security properties of our scheme. Particularly, our analysis includes four aspects: data confidentiality, access control, tag consistency and resistance to brute-force attacks.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2987793, IEEE Transactions on Dependable and Secure Computing

5.1 Data Confidentiality

Our scheme aims to ensure that any adversary including the semi-trusted CSP and unauthorized users cannot get data content from outsourced data. Specifically, we prove that our scheme is semantically secure against the CSP or unauthorized users. Note that as described in Section 2.2, the CSP would not collude with any user due to the reputation. For the outsourced data $(C_{a_i}, T_m^i, \tau_m^i, C_m^i)$, T_m^i and τ_m^i are generated from C_m^i, so it comes down to whether the CSP or unauthorized users can obtain m from the symmetric ciphertext $C_m^i = \mathrm{SE}_{b_i}(m)$. Due to the security of AES algorithm [34], the only way is to obtain $b_i = H_1(a_i)$. Since the secret a_i is encapsulated in $C_{a_i} = \left(g_1^{\beta_i} y^{r_i}, a_i z^{\beta_i}\right)$, the crux of the security in our scheme is the advantage of obtaining a_i. Therefore, in the following parts, we focus on the security analysis related to a_i.

First, we prove that our scheme is semantically secure for unauthorized users under the SD assumption (see Definition 3).

Theorem 1. *The proposed scheme is semantically secure for unauthorized users if the SD assumption holds.*

Proof. Suppose a polynomial-time adversary $\mathcal{A}$ (corrupting an unauthorized user) can attack our scheme with advantage $\epsilon(\kappa_0)$, then we can build an algorithm $\mathcal{B}$ that can break the SD assumption with the same advantage as follows.

- **Init:** Given the parameters $(N, \mathbb{G}, \mathbb{G}_T, e, g, z, y)$[1], $\mathcal{B}$ randomly selects a master private key $\alpha \in \mathbb{Z}_N$, and computes $g_1 = g^\alpha \in \mathbb{G}$. Note that in our scheme, $\mathcal{B}$ would generate the public and private key pair for each user, even for an unauthorized user. Thus, $\mathcal{B}$ generates $(pk_{\mathcal{A}}, sk_{\mathcal{A}}) = (g^{d_{\mathcal{A}}\alpha^{-1}}, d_{\mathcal{A}})$ for $\mathcal{A}$, where $d_{\mathcal{A}} \in \mathbb{Z}_N$ is a random number. Since data owners do not affect the security analysis, without loss of generality, we consider one data owner instead of ω data owners for the sake of simplicity, i.e., $\mathcal{B}$ only generates one public key $pk = g^{d\alpha^{-1}}$ instead of $(pk_1, \ldots, pk_\omega)$ (see Section 4.1, System initialization). Finally, $\mathcal{B}$ gives the public parameters $(N, \mathbb{G}, \mathbb{G}_T, e, g_1, z, y, pk, pk_{\mathcal{A}})$ together with $d_{\mathcal{A}}$ to $\mathcal{A}$.
- **Challenge:** $\mathcal{A}$ selects two messages $m_0, m_1 \in \mathbb{G}_T$, and then submits them to $\mathcal{B}$. $\mathcal{B}$ flips a fair binary coin b^*, and returns an encryption of $m_{b^*} \in \{m_0, m_1\}$. The ciphertext is output as
$$C_{b^*} = (g_1^\beta y^r, m_{b^*} \cdot z^\beta),$$
where $\beta, r \in \mathbb{Z}_N$.
- **Guess:** $\mathcal{A}$ outputs a guess b' of b^*. If $b' = b^*$, $\mathcal{B}$ outputs 1 to indicate that y is uniform in the subgroup $\mathbb{G}_p$; otherwise, $\mathcal{B}$ outputs 0 to indicate that y is uniform in $\mathbb{G}$.

When y is uniform in the subgroup $\mathbb{G}_p$, then the public key and element $g_1^\beta y^r$ given to $\mathcal{A}$ are as in the real semantic security game. In this case, the advantage of $\mathcal{A}$ is $\epsilon(\kappa_0)$ by definition, i.e., $\mathcal{A}$ can obtain $z^{d_{\mathcal{A}}\beta}$ with advantage $\epsilon(\kappa_0)$, and then obtain m_{b^*} with the private key $d_{\mathcal{A}}$ by computing $m_{b^*} \cdot z^\beta / (z^{d_{\mathcal{A}}\beta})^{d_{\mathcal{A}}^{-1}}$, so we have $\Pr[\mathcal{A}(b' = b^*)] = 1/2 + \epsilon(\kappa_0)$. Since $\mathcal{B}$ outputs 1 exactly when the output b' of $\mathcal{A}$ is equal to b^*, we have

$$\begin{aligned}\Pr[\mathcal{B}(N, \mathbb{G}, \mathbb{G}_T, e, y) = 1 : y \in \mathbb{G}_p]| &= \Pr[\mathcal{A}(b' = b^*)] \\ &= \frac{1}{2} + \epsilon(\kappa_0).\end{aligned}$$

When y is uniform in $\mathbb{G}$, the element $g_1^\beta y^r$ is uniformly distributed in $\mathbb{G}$ and is independent of b^*. Hence, $\Pr[\mathcal{A}(b' = b^*)] = 1/2$, which indicates that

$$\Pr[\mathcal{B}(N, \mathbb{G}, \mathbb{G}_T, e, y) = 1 : y \in \mathbb{G}] = \Pr[\mathcal{A}(b' = b^*)] = \frac{1}{2}.$$

Therefore, we can obtain that

$$\begin{aligned}\mathrm{SD} - \mathrm{Adv}_{\mathcal{B}} =& |\Pr[\mathcal{B}(N, \mathbb{G}, \mathbb{G}_T, e, y) = 1 : y \in \mathbb{G}] - \\ & \Pr[\mathcal{B}(N, \mathbb{G}, \mathbb{G}_T, e, y) = 1 : y \in \mathbb{G}_p]| \\ =& \left|\frac{1}{2} - \left(\frac{1}{2} + \epsilon(\kappa_0)\right)\right| = \epsilon(\kappa_0).\end{aligned}$$

Based on Definition 3, we obtain $\epsilon(\kappa_0) \leq \mathtt{negl}(\kappa_0)$. □

Next, we prove that our scheme is semantically secure for the CSP under the DDBDH assumption. Since the CSP possesses the private key p, it knows g^p and can compute $(g_1^\beta y^r)^p = g_1^{p\beta} = (g^p)^{\alpha\beta}$. Thus, the ciphertext given to the CSP can be reduced to $C_m = \left((g^p)^{\alpha\beta}, m \cdot z^\beta\right)$. For convenience, we use g and $e(g, g)$ instead of g^p and $z = e(g, g)^p$ in the following proof, respectively. Thus, for the CSP, the encryption operation given in Eq. (2) can be reduced to $C_m = \left((g^p)^{\alpha\beta}, mz^\beta\right) = \left(g^{\alpha\beta}, m \cdot e(g, g)^\beta\right)$.

Theorem 2. *The proposed scheme is semantically secure for the CSP if the DDBDH assumption holds.*

Proof. Suppose a polynomial-time adversary $\mathcal{A}$ (corrupting the semi-trusted CSP) can attack our scheme with advantage $\epsilon(\kappa_0)$, then we can build an algorithm $\mathcal{B}$ that can break the DDBDH assumption with the same advantage as follows.

- **Init:** Given the parameters (g, g^a, g^b, W), $\mathcal{B}$ sets $g_1 = g^b$. Then it randomly selects $d \in \mathbb{Z}_N$, and computes $pk = g^d$. Finally, $\mathcal{B}$ gives the public parameters $(N, \mathbb{G}, \mathbb{G}_T, e, g_1, pk)$[2]to $\mathcal{A}$.
- **Challenge:** $\mathcal{A}$ selects two messages $m_0, m_1 \in \mathbb{G}_T$, and then submits them to $\mathcal{B}$. $\mathcal{B}$ flips a fair binary coin b^*, and returns an encryption of $m_{b^*} \in \{m_0, m_1\}$. The ciphertext is output as
$$C_{b^*} = (g^a, m_{b^*} \cdot W).$$
- **Guess:** Note that in our scheme, the CSP can conduct the re-encryption operation (i.e., compute Eq. (4)), thus $\mathcal{A}$ first computes $e(g^a, pk) = e(g, g)^{bdk} = e(g_1^k, pk) = e(g, g)^{\gamma k}$, [3] where $\gamma = bd$. Then, it outputs a guess b' of b^*. If $b' = b^*$, $\mathcal{B}$ outputs 1 to indicate that $W = e(g, g)^{a/b}$; otherwise, $\mathcal{B}$ outputs 0 to indicate that W is a random element from $\mathbb{G}_T$.

When $W = e(g, g)^{a/b}$, then we can get $e(g, g)^{a/b} = e(g, g)^{bk/b} = e(g, g)^k$. Thus, $\mathcal{A}$ sees a proper encryption of m_{b^*}, i.e., $C_{b^*} = (g^{bk}, m_{b^*} \cdot e(g, g)^k)$. The advantage of $\mathcal{A}$ is $\epsilon(\kappa_0)$ by definition, i.e., $\mathcal{A}$ can obtain m_{b^*} with advantage $\epsilon(\kappa_0)$ from the re-encryption ciphertext $\left(e(g, g)^{\gamma k}, m_{b^*} \cdot e(g, g)^k\right)$, so we have $\Pr[\mathcal{A}(b' = b^*)] = 1/2 + \epsilon(\kappa_0)$. Since $\mathcal{B}$ outputs 1 exactly when the output b' of $\mathcal{A}$ is equal to b^*, we have that

$$\Pr\left[\mathcal{B}\left(g, g^a, g^b, e(g, g)^{\frac{a}{b}}\right) = 1\right] = \Pr\left[\mathcal{A}(b' = b^*)\right] = \frac{1}{2} + \epsilon(\kappa_0)$$

1. Since $\mathcal{B}$ does not know the factorization of the group order N, the parameter $z = e(g, g)^p$ is also given to $\mathcal{B}$ together with g.

2. As analyzed above, the encryption operation is reduced to $C_m = \left(g^{\alpha\beta}, m \cdot e(g, g)^\beta\right)$, so we can ignore the parameter $y = x^q$ here. In our scheme, both the KGC and CSP know the the factorization of the group order N, i.e., p, they can obtain $z = e(g, g)^p$. In other words, $\mathcal{A}$ and $\mathcal{B}$ know z, and thus we also ignore z here. Note that the KGC would not generate the public/private key for the CSP, thus $\mathcal{A}$ cannot obtain the corresponding key pair $(pk_{\mathcal{A}}, sk_{\mathcal{A}})$.

3. We can think of g^a as g^{bk} for some $k \in \mathbb{Z}_N$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2987793, IEEE Transactions on Dependable and Secure Computing

When W is a random element from $\mathbb{G}_T$, then $m_{b^*}W$ is a random element of $\mathbb{G}_T$ from $\mathcal{A}$'s view, which means that $\mathcal{A}$ gains no information about b^*. Hence, $\Pr[\mathcal{A}(b'=b^*)]=1/2$, which implies that

$$\Pr\left[\mathcal{B}\left(g,g^a,g^b,W\right)=1\right]=\Pr[\mathcal{A}(b'=b^*)]=\frac{1}{2}.$$

Therefore, we can obtain that

$$\begin{aligned}\mathrm{DDBDH}-\mathrm{Adv}_{\mathcal{B}}=&|\Pr[\mathcal{B}(g,g^a,g^b,W)=1]-\\&\Pr[\mathcal{B}(g,g^a,g^b,W=e(g,g)^{a/b})=1]|\\=&\left|\frac{1}{2}-\left(\frac{1}{2}+\epsilon(\kappa_0)\right)\right|=\epsilon(\kappa_0),\end{aligned}$$

which implies that $\epsilon(\kappa_0)\leq \mathtt{negl}(\kappa_0)$ with Definition 2. □

Based on Theorems 1 and 2, the security of the proposed scheme is shown in Theorem 3.

Theorem 3. *The proposed scheme is semantically secure under the SD and DDBDH assumptions.*

Obviously, if the internal adversary with a certain secret cannot obtain the plaintext from the outsourced data, there is no doubt that the external adversary also cannot. Therefore, our scheme can ensure that any adversary including the CSP or unauthorized users cannot get data content from the outsourced data.

5.2 Resistance to Brute-Force Attacks

In this section, we analyze that our scheme can resist brute-force attacks launched by the CSP or unauthorized users based on the background knowledge of message space $\mathcal{M}$.

First, we analyze that the CSP cannot determine which content corresponds to the specific encrypted data by launching offline brute-force attacks. For the specific stored data $(C_{a_i},T_m^i,\tau_m^i,C_m^i)$, the CSP wishes to determine which data in $\mathcal{M}$ generates them. Similar to Section 5.1, T_m^i and τ_m^i are generated from the ciphertext C_m^i, so the core is to find which data corresponds to C_m^i. More specifically, for each data $m_k\in\mathcal{M}$, the CSP tries to generate the valid ciphertext $C_{m_k}^i$, and then checks whether $C_m^i=C_{m_k}^i$ to determine whether m_k is the plaintext of C_m^i. However, based on Theorem 2, the CSP cannot obtain the secret a_i (i.e., the symmetric key b_i), so it cannot generate a valid ciphertext $C_{m_k}^i$. Similarly, the CSP cannot decrypt C_m^i to obtain m without b_i.

Then, we analyze that unauthorized users cannot know whether an interested user owns some data by launching online brute-force attacks. More precisely, for any data $m_k\in\mathcal{M}$, in order to know whether an interested user U_i owns it, an unauthorized user U_j tries to upload m_k and relies on the CSP to perform data deduplication. If the duplicate exists, it means that U_i owns m_k. Otherwise, U_i does not own m_k. However, U_j cannot obtain this useful information under our scheme. The main reason is that according to Theorem 1, U_j as an unauthorized user of U_i cannot obtain U_i's symmetric key b_i. As described in Section 4.2.3, without the symmetric key b_i, U_j cannot generate a valid tag $T_{m_k}^i$ used for checking the duplicate in U_i's dataset.

5.3 Access Control

In this section, we describe that our scheme can eliminate the redundancy without violating access control. In other words, after eliminating the redundancy, it still ensures that any authorized user can correctly decrypt the outsourced encrypted data, while all unauthorized users cannot.

First, we show that any authorized user can obtain the symmetric key selected by the data owner. In our scheme, each user U_i encrypts a random secret a_i used to generate the symmetric key, and then sends this key encapsulated ciphertext C_{a_i} together with an authorization set O_i to the CSP (see Section 4.1). Thus, it should ensure that all users in O_i can obtain a_i to correctly decrypt all encrypted data outsourced by U_i. Specifically, since the CSP owns the secret p, it can re-encrypt C_{a_i} for any user $U_j\in O_i$ with the corresponding public key $pk_j=g^{d_j\alpha^{-1}}$ as $e\big((g_1^{\beta_i}y^{r_i})^p,pk_j\big)=z^{d_j\beta_i}$ forming a re-encrypted ciphertext $C_{a_i}^j=(z^{d_j\beta_i},a_iz^{\beta_i})$. Then, U_j uses the own private key d_j to obtain the secret a_i (see Eq. (5)). Accordingly, with the symmetric key $b_i=H_1(a_i)$, authorized users can correctly decrypt the encrypted data outsourced by U_i. Furthermore, as described in Section 4.2.3, the CSP only checks the duplicate in the case that O_i and O_j are mutually contained. Specifically, if the duplicate exists, then the unique copy is always stored in the dataset of the user whose authorization set is the superset. For example, if O_i is the superset, i.e., $O_j\subset O_i$, then the unique copy stored in the cloud is uploaded by U_i. Since all users in $O_i=O_j\cup O_i$ can obtain the symmetric key b_i selected by U_i, it ensures that all users in sets O_i and O_j can access outsourced data owned by U_i and U_j when only one copy is stored.

Based on Theorem 1, if U_j is not authorized by U_i, U_j cannot obtain the secret a_i even owning the private key d_j. Thus, our scheme ensures that all unauthorized users cannot correctly decrypt the encrypted data outsourced by an interested data owner.

5.4 Tag Consistency

In secure deduplication schemes, a malicious user (corrupted by an adversary) may carry out duplicate faking attacks during data upload to prevent legitimate users from obtaining correct data. Particularly, suppose a malicious user U' defines an authorization set $O'=\mathcal{U}$ so that the access set I_i of each user U_i includes him, i.e., $U'\in I_i$. We also suppose that U' may have the same data m with a legitimate user U_i. To prevent U_i from obtaining m, U' maliciously generates a ciphertext C_{m^*} from the different data m^*, and initially uploads it with a tag T_m generated from m. If the CSP does not check tag consistency, then it would store (C_{m^*},T_m,τ_m) in U''s dataset. When U_i subsequently uploads m, the CSP would check the duplicate in U''s dataset due to $U'\in I_i$. As described in Section 4.2.3, the CSP definitely finds the duplicate after the inter-deduplication and stores C_{m^*} as the unique copy due to $O_i\subseteq O'$. As a result, U_i including authorized users cannot obtain m from the inconsistent ciphertext C_{m^*}.

Fortunately, our scheme can easily detect duplicate faking attacks. As shown in Section 4.2.4, when the CSP receives C_{m^*} from U', it generates the corresponding verifiable token $\tau_{verify}=H_2\left(h_1(C_{m^*})\|\cdots\|h_f(C_{m^*})\right)$. Then, the CSP checks whether $\tau_{m^*}=\tau_m$ holds, where $\tau_m=H_2\left(h_1(C_m)\|\cdots\|h_f(C_m)\right)$ is computed based on the tag T_m. If $\tau_{verify}\neq\tau_m$, it implies C_{m^*} and T_m are inconsistent. Thus, the CSP drops (C_{m^*},T_m,τ_m). Note that $H_2(\cdot)$ is a cryptographic hash function that can resist the hash collision. If the input is different, then the hash value should be different. However, in the Bloom filter, f hash functions $\{h_1,h_2,\ldots,h_f\}$ are not cryptographic hash functions, so the collision may exist. That is, two different ciphertexts C_{m^*} and C_m may generate two tags T_{m^*} and T_m such that $h_i(C_{m^*})=h_i(C_m)$ for $i=1,2,\ldots,f$, resulting in two same token τ_{verify} and τ_m. More specifically, for each hash function $h_i:\{0,1\}^*\rightarrow\{1,2,\ldots,\delta\}$, the probability of collision is $\Pr\left(h_i(C_{m^*})=h_i(C_m)\right)=1/\delta$. Thus, the probability of tag collision is $\Pr\left(\tau_{verify}=\tau_m\right)=\prod_{i=1}^{f}\Pr\left(h_i(C_{m^*})=h_i(C_m)\right)=$

TABLE 2: Comparison of security for authorized deduplication schemes.

Scheme	Without IS in data upload phase	Confidentiality	Resistance to brute-force attacks	Tag consistency	Access control
Li et al.'s scheme [17]	○	●	◑	○	●
Cui et al.'s scheme [18]	○	●	◑	◑	◑
Yan et al.'s scheme [19]	○	●	○	○	●
Liu et al.'s scheme [35]	●	●	●	●	○
Our scheme	●	●	●	●	●

"IS": Independent Server, "●" = satisfied, "◑"= partially satisfied, and "○"= unsatisfied.

$\left(\frac{1}{\delta}\right)^f$. Actually, the probability of collision can be negligible by reasonably selecting f and δ. For example, condisering the false positive probability $\mathcal{P}$ (see Eq. (1)), if $f = 8$ and $\delta = 2^{27}$, then with at most 2^{20} elements in Bloom filter, the false positive probability is $\mathcal{P} \approx 10^{-10}$ and the tag collision probability is $\Pr\left(\tau_{verify} = \tau_m\right) = 2^{-216}$.

5.5 Comparison

We list the comparison of the security for related deduplication schemes in Table 2, where ●, ◑ and ○ denote satisfied, partially satisfied and unsatisfied, respectively. From the table, we can see our scheme can achieve more security-related criteria than those existing schemes under a simplified and practical model. Specifically, from a system architecture viewpoint, both our scheme and Liu et al.'s scheme [35] are superior to the schemes in [17]–[19]. During the phase of data upload, neither our scheme[4] nor Liu et al.'s scheme [35] requires an extra independent server (IS), as a result both of them satisfy the requirement of without IS, marked as "●" in second column of Table 2. On the other hand, in order to resist offline brute-force attacks launched by the CSP, the system in [17] introduces a private cloud and limits data deduplication operations to the private cloud. Besides the CSP, the system in [18] also includes two extra independent servers: one is a private cloud for performing data deduplication to resist the offline brute-force attacks launched by the CSP, and the other is a trusted attribute authority (AA) for performing access control. In [19], the system needs to involve a trusted authorized party to achieve access control. Thus, none of these three schemes can meet the requirements without IS, marked as "○".

From a security perspective, our scheme achieves more security-related criteria than existing schemes. More specifically, our scheme satisfies data confidentiality, tag consistency and access control while resisting brute-force attacks. However, as shown in the fourth column of Table 2, the resistance to brute-force attacks for both schemes in [17], [18] is partially satisfied, marked as "◑", and Yan et al.'s scheme [19] cannot resist brute-force attacks, marked as "○". The reason is that both schemes in [17], [18] can only resist the offline brute-force attacks launched by the CSP but cannot resist such attacks launched by the honest-but-curious private cloud. The scheme in [19] directly hashes the message to check duplicates, and thus similar to the convergent encryption (CE) algorithm, it suffers from brute-force attacks launched by the CSP. For tag consistency, since the ciphertext is not directly related to the corresponding tag, Li et al.'s scheme [17] and Yan et al.'s scheme [19] cannot support the CSP to check the tag consistency during data upload. That is, both schemes cannot satisfy tag consistency, which is marked as "○". Because tag consistency is conducted after downloading and decrypting the ciphertext, which makes it impossible to determine whether the inconsistent data is caused during data upload (i.e., duplicate faking attacks) or at data storage, the tag consistency in [18] is partially satisfied (marked as "◑"). For the requirement of access control, Cui et al.'s scheme [18] is partially satisfied (marked as "◑"). The reason is that when two access policies are not mutually contained, the private cloud re-encrypts the ciphertext to yield a new ciphertext associated with an access structure which is the union of two access policies. In this case, the symmetric key of one data owner is leaked to the users who satisfy the access policy of another data owner but do not satisfy that owner. Accordingly, this somewhat violates access control. The scheme in [35] focuses on resisting brute-force attacks without the IS and does not consider access control that is a prevalent requirement in cloud computing. Compared to [35], our scheme releases the pressure of users, that is, previous uploaders are not required to be always online to involve in helping new uploaders obtain the symmetric key related to the generation of the ciphertext and tag. In addition, our scheme can completely resist the online brute-force attack launched by the CSP (see Theorem 2), while the scheme in [35] can only reduce the capacity of such attack through the rate-limiting strategy.

4. Note that the system model in our scheme contain a trusted KGC, but this party only participates in the system initialization phase and does not participate in the data upload phase.

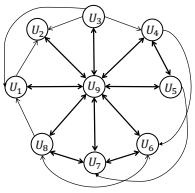
6 Performance Evaluation

In this part, we conduct experiments in Java with jPBC [36] and GMP [37] libraries running on the MacBook Pro (a 2.3 GHz Intel Core i5 and 8 GB memory) to evaluate the performance of our scheme in terms of computational, communication and storage overheads. Moreover, based on the granularity of access rights (i.e., user-level), we compare our scheme with Yan et al.'s scheme [19] and Cui et al.'s scheme [18] with regard to file-level deduplication and chunk-level deduplication. We also compare our scheme with Liu et al.'s scheme [35] to show the efficiency without an extra independent server. It is worth noting that since data deduplication is only related to the phase of data upload, we ignore related overheads of the system initialization and data download parts in the comparison.

6.1 Simulation Setup

In the simulation, 9 users $\{U_1, U_2, \ldots, U_9\}$ in total are considered in the system, and the related authorization sets and access sets are shown in Fig. 5, which includes all possible cases of the set relationship. Three datasets S_1, S_2 and S_3 with different file size are used in the simulation, where the file sizes in S_1, S_2 and S_3 are about 1 KB, 512 KB and 1 MB, respectively, and each dataset contains 100 files. For each dataset S_k, $k = 1, 2, 3$, all users upload it within 10 time periods in turn. Algorithm 2 shows the simulation process of data upload. Specifically, for each dataset S_k ($k = 1, 2, 3$), 9 users randomly selects 10 files from $S_k \backslash 10(j-1)$ (see footnote 4) to upload in turn at time t_j. After time t_{10}, each user finishes uploading 100 files of S_k. After that, each user again randomly selects 10 files from S_k to upload in turn at t_{11} and t_{12}, respectively, to verify the advantage of intra-deduplication (see lines 9-14).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2987793, IEEE Transactions on Dependable and Secure Computing



User	Authorization set O_i	Access set I_i
U_1	$\{U_1, U_2, U_9\}$	$\{U_3, U_8, U_9\}$
U_2	$\{U_2, U_9\}$	$\{U_1, U_3, U_9\}$
U_3	$\{U_1, U_2, U_3, U_4, U_9\}$	$\{U_9\}$
U_4	$\{U_4, U_5, U_6, U_9\}$	$\{U_3, U_5, U_9\}$
U_5	$\{U_4, U_5, U_7, U_9\}$	$\{U_4, U_9\}$
U_6	$\{U_6, U_7, U_8, U_9\}$	$\{U_4, U_7, U_8, U_9\}$
U_7	$\{U_6, U_7, U_8, U_9\}$	$\{U_5, U_6, U_8, U_9\}$
U_8	$\{U_1, U_6, U_7, U_8, U_9\}$	$\{U_6, U_7, U_9\}$
U_9	$\{U_1, U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9\}$	$\{U_1, U_2, U_3, U_4, U_5, U_6, U_7, U_8\}$

Fig. 5: Authorization and access sets for users in the system.

Algorithm 2 Simulation process of data upload

for ($k = 1$; $k \leq 3$; k++) **do**
  // Upload three datasets S_1, S_2 and S_3 in turn.
  for ($j = 1$; $j \leq 10$; j++) **do**
    // For each dataset S_k, all users upload it within 10 time periods.
    for ($i = 1$; $i \leq 9$; i++) **do**
      At time t_j, U_i randomly selects 10 files from $S_k \backslash 10(j-1)$.[5]
    end for
  end for
  for ($j = 11$; $j \leq 12$; j++) **do**
    // For each dataset S_k, all users again upload 10 files at time t_{11} and t_{12}, respectively.
    for ($i = 1$; $i \leq 9$; i++) **do**
      At time t_j, U_i randomly selects 10 files from S_k.
    end for
  end for
end for

Besides, for the bilinear pairing parameters [36], we choose type A1 for our scheme and type A for the other three schemes[6]. We adopt AES-256 encryption algorithm in CBC mode, i.e., $\kappa_1 = 256$ bits. We also set $|U| = |link| = |label| = 64$ bits. Furthermore, we set $f = 8$ and $\delta = 2^{27}$ so that the Bloom filter can contain up to 2^{20} data. Similar to [35], we measure deduplication effectiveness using the deduplication percentage ρ, which is defined as:

$$\rho = \left(1 - \frac{Number\ of\ all\ files\ in\ storage}{Total\ number\ of\ upload\ requests}\right) \cdot 100\%$$

6.2 Simulation Results

We compare the effectiveness of deduplication for four schemes in Fig. 6, which demonstrates that the effectiveness of chunk-level deduplication is superior to the file-level deduplication. Furthermore, the deduplication effectiveness of Cui et al.'s scheme [18] and Liu et al.'s scheme [35] is the best, followed by our scheme, the worst is Yan et al.'s scheme [19]. Although the deduplication effectiveness of our scheme is less than that of schemes in [18], [35], the gap is shrinking over time and our scheme can further satisfy access control. Specifically, Liu et al.'s scheme does not consider access control, and thus the CSP checks for duplicates in all stored data. In other words, the CSP just stores one copy of each uploaded data, ensuring the best deduplication effectiveness. Among the other three schemes, two authorization sets O_i and O_j contain three relationships: (1) $O_i \subseteq O_j$, (2) $O_j \subset O_i$ and (3) O_i and O_j are not mutually contained. More specifically, in order to ensure access control, Yan et al.'s scheme only considers deduplicating the duplicate in the case of $O_i \subseteq O_j$. For example, when U_1 requests to upload m, the CSP only checks the duplicate in the datasets of U_3 and U_9. In our scheme, we consider the first two cases. With the same example, besides the datasets of U_3 and U_9, the CSP also checks the duplicate in U_2's dataset due to $O_2 \subset O_1$. Thus, the deduplication effectiveness of our scheme is better than that of Yan et al.'s scheme. Cui et al.'s scheme takes all cases into consideration to achieve the same effectiveness as Liu et al.'s scheme at the expense of access control to some extent (see Table 2). For example, when U_4 requests to upload m previously uploaded by U_5, the private cloud would find the duplicate. However, O_4 and O_5 are not mutually contained. To ensure all users in $O_4 \cup O_5$ correctly decrypt C_m generated by U_5, the private cloud need to conduct the re-encryption operation to make U_5's symmetric key available to these users. In this case, an unauthorized user $U_6 \notin O_5$ can also obtain U_5's key, which violates access control.

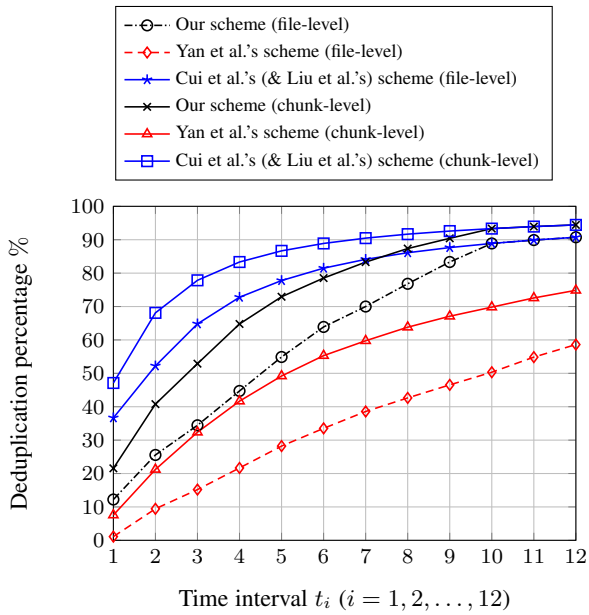


Fig. 6: Effectiveness of deduplication.

We depict the comparison of computational costs in Fig. 7. From the figure, it can be obviously shown that computational costs of chunk-level deduplication are larger than that of file-level deduplication. Further, when the message length ($|m|$) is relatively small, computational efficiency is mainly affected by deduplication-related operations (e.g., tag generation, duplicate search and verification). As a result, Fig. 7(a) demonstrates that computational costs associated with deduplication operations in our scheme are the lowest. As $|m|$ increases, the computational efficiency of our scheme is slightly lower than that of Yan et al.'s scheme (see Figs. 7(b) and 7(c)). The main reason is that our scheme generate the tag with the encrypted data, and thus computational costs are influenced by the symmetric encryption algorithm (i.e., $|m|$). From Fig. 7, we can also observe that the computational efficiency of our scheme is far superior to Liu et al.'s scheme and Cui et al.'s scheme, especially for the chunk-level deduplication. The main factors include the tag generation, the operation of duplicate verification and the time complexity of duplicate search. Specifically, Liu et al.'s scheme requires users to interactively run the same-input-PAKE protocol many times to generate the tag, which incurs high computational costs. Cui et al.'s

5. $10(j-1)$ represents the files randomly selected before time t_j, and $S_k \backslash 10(j-1)$ indicates the remaining files in S_k after removing the $10(j-1)$ files selected before time t_j.

6. For fair comparison, the PAKE protocol of Liu et al.'s scheme [35] is implemented in the cyclic multiplicative group consisting of points on an elliptic curve over a finite field, which is the same as the group of the bilinear pairing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2987793, IEEE Transactions on Dependable and Secure Computing

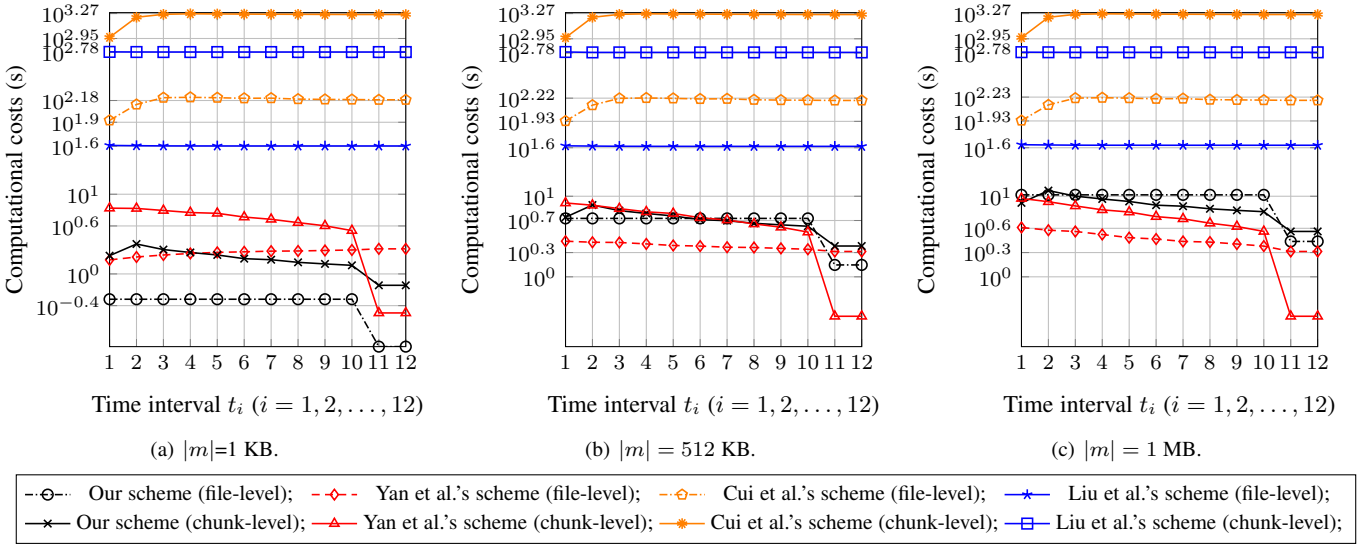


Fig. 7: Comparison of computational costs in the phase of data upload.

scheme checks duplicates by performing time-consuming pairing operations and the time complexity of duplicate search increases linearly with the number of stored files (i.e., $\mathbf{O}(2n)$ pairing operations for n stored data). On the contrary, our scheme does not require interactive protocols and the corresponding duplicate search is based on an efficient Bloom filter technique, where the time complexity of duplicate search is $\mathbf{O}(f)$ $(f \ll n)$.

We plot the comparison of communication overhead in Fig. 8. From the figure, it can be obviously seen that the communication overhead increases as the message length $|m|$ increases. When $|m|$ is relatively small, the communication overhead associated with deduplication is dominant. Specifically, as shown in Fig. 8(a), the communication overhead of our scheme is the lowest, while the communication overhead of Liu et al.'s scheme (chunk-level deduplication) is the largest. Similar to the above analysis, the interactions of the same-input-PAKE protocol in Liu et al.'s scheme would cause extra communication costs and the chunk-level deduplication would further increase the number of executions of this protocol. Figures 8(b) and 8(c) show that as $|m|$ increases, the communication overheads of our scheme is almost the same as that of Yan et al.'s scheme, and both of them are less than that of Cui et al's scheme but larger than that of Liu et al's scheme. In addition, it also shows that the communication efficiency of chunk-level deduplication is better than that of file-level deduplication, which implies that the communication efficiency is also influenced by the deduplication effectiveness. The reason is that Cui et al.'s scheme adopts the server-side deduplication [28], where users always send outsourced data to the CSP regardless of whether duplicates exist. Obviously, this type of deduplication cannot save the communication overhead even the duplicate exists. On the contrary, the other three schemes use the client-side deduplication, which can reduce communication overhead when duplicates exist. In addition, since the deduplication effectiveness of Liu et al.'s scheme is the best, the number of uploaded ciphertexts is the least, which implies the corresponding communication efficiency is the best.

Figure 9 shows the comparison of storage cost. It can be obviously shown that the storage efficiency is significantly influenced by the deduplication effectiveness and message length $|m|$. More specifically, storage costs are gradually dominated by the encrypted data as $|m|$ increases. Further, the number of stored encrypted data decreases as the deduplication effectiveness improves. Thus, from the figure, we can observe that the storage efficiency of chunk-level deduplication is better than that of file-level deduplication. Fig. 9(a) indicates that when $|m|$ is relatively small, our scheme reduces the storage costs associated with deduplication compared to Cui et al.'s scheme. When $|m|$ is relatively large, figures 9(b) and 9(c) show that the storage efficiency of Cui et al.'s scheme and Liu et al.'s scheme are the best, followed by our scheme, and the worst by Yan et al.'s scheme. Although Liu et al.'s scheme and Yan et al.'s scheme can obtain the optimal storage efficiency, Liu et al.'s scheme does not consider access control while Cui et al.'s scheme sacrifices the complete access control. By contrast, our scheme can obtain storage efficiency while ensuring access control.

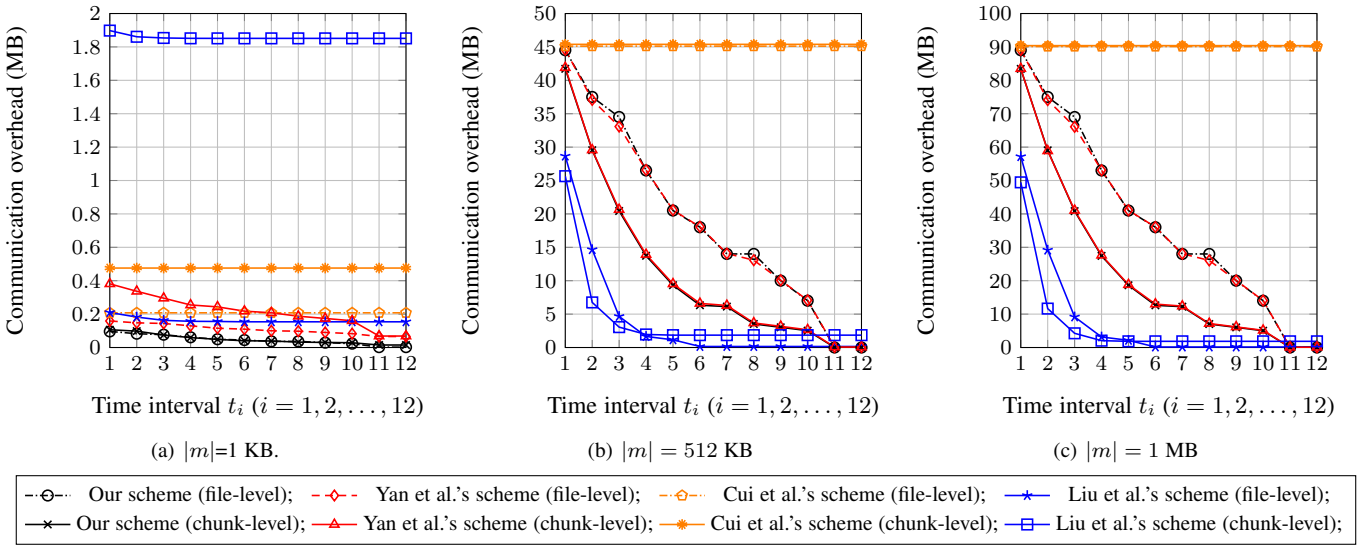
Consequently, under the guarantee of given security criteria, our scheme is indeed efficient in terms of the effectiveness of deduplication, computational cost, communication overhead and storage cost compared with the up-to-date works, which shows the significance and practical potential of our scheme to support authorized secure deduplication.

7 Related Work

Secure deduplication technique, as it can eliminate redundant data while achieving data confidentiality, has been widely developed by the research community. The related researches can be categorized into the following three aspects: data confidentiality, tag consistency and access control.

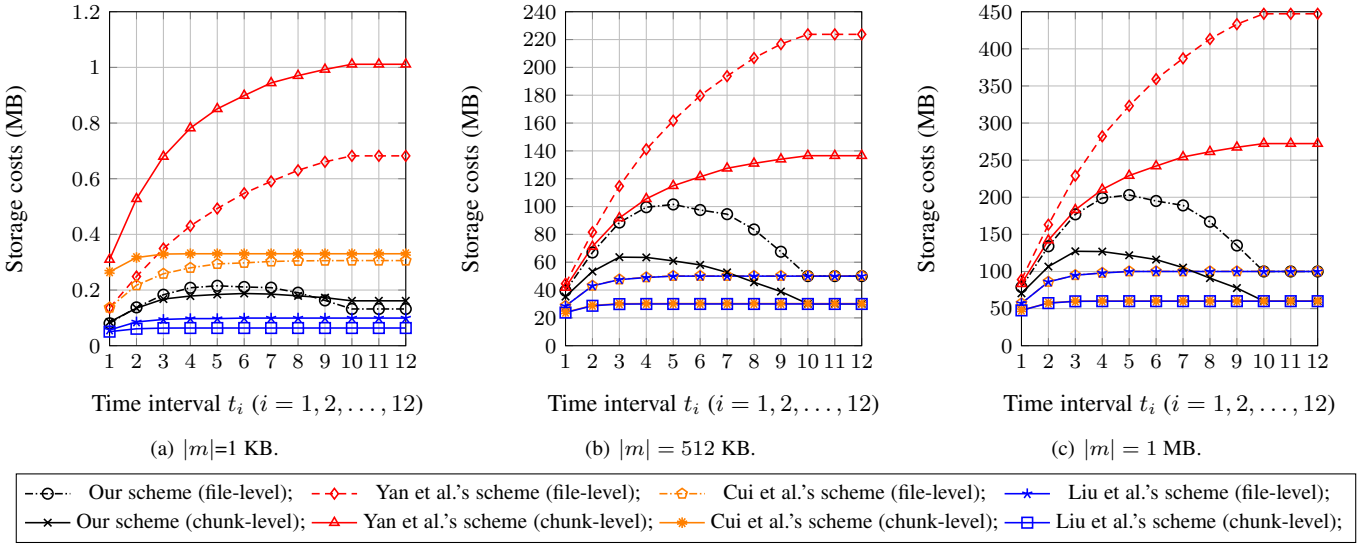
Data confidentiality: To ensure data confidentiality, Douceur et al. [6] first introduced the convergent encryption to achieve deduplication over encrypted data. Then, Bellare et al. [12] formalized convergent encryption a new cryptographic primitive called message-locked encryption and gave a semantic security proof for unpredictable messages. After that, many implementations and variants of the convergent encryption or message-locked encryption were deployed in [8], [13], [14], [38], [39]. However, for predictable messages, the message-locked encryption is vulnerable to brute-force attacks. Hence, data confidentiality would be defeated as the adversary can obtain which data corresponds to the specific ciphertext. In order to resist brute-force attacks, some server-aided secure deduplication schemes have been proposed [10], [11], [40], where each user interacts with one or a threshold number of additional key servers to obtain the convergent key. However, these schemes sacrifice either the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.2987793, IEEE Transactions on Dependable and Secure Computing



(a) $|m|$=1 KB. (b) $|m| = 512$ KB (c) $|m| = 1$ MB

Fig. 8: Comparison of communication overhead in the phase of data upload.



(a) $|m|$=1 KB. (b) $|m| = 512$ KB. (c) $|m| = 1$ MB.

Fig. 9: Comparison of storage cost in the phase of data upload.

deduplication efficiency [10], [40] or the efficiency of computation, communication and storage [11]. Liu et al. [35] deployed the secure cross-user deduplication scheme that provides strong security without requiring any additional independent servers. However, all users are requested to be constantly online and heavily involved in helping uploaders obtain the convergent key generated by the user who has already uploaded the same file. This is suitable for peer-to-peer paradigm, but less applicable for the cloud environment [11].

Tag consistency: As mentioned in [12], [41], there exists a specific attack, called duplicate faking attack, in secure deduplication, which causes legitimate users to obtain incorrect data. In more details, an adversary uploads a maliciously-generated ciphertext such that the underlying data is different from the data corresponding to the tag. As a result, the subsequent legitimate users who upload the same tag cannot extract the exact data after deduplication. The main reason for this attack is the irrelevance of tag and ciphertext, which makes it impossible for cloud service providers to check the tag consistency. To solve this issue, some schemes have taken the tag consistency into consideration [13]–[15], [24], [42]. However, in these schemes, the verification of tag consistency is conducted by legitimate users after they have downloaded and decrypted the ciphertext. In this case, when the verification is failed, it cannot directly conclude the incorrect data is caused by duplicate faking attacks because the stored data may be corrupted during data storage. To overcome this problem, Li et al. [16] directly generated the tag from the ciphertext to support cloud service providers to check the tag consistency during data upload phase.

Access control: As described in [43], with the great benefits of cloud computing, data owners prefer to outsource their encrypted data to cloud service providers and usually share their outsourced data to authorized users. In this context, several existing schemes have combined the secure deduplication with access control. For example, Li et al. [17] considered the different privileges of users in duplicate check besides the data itself, where the duplicate can be eliminated only when the privileges of files defined by users are matched. Tang et al. [44] leveraged ciphertext-policy attribute-based encryption (CP-ABE) [45] to achieve authorized secure deduplication. Cui et al. [18] achieved access control for secure deduplication by using linear secret

sharing technique [46]. Besides, Yan et al.'s scheme [19] offered secure deduplication with access control by adopting the CP-ABE and proxy re-encryption technique (PRE) [30]. However, to the best of our knowledge, these schemes cannot simultaneously achieve the efficiency, tag consistency and resistance to brute-force attacks.

8 Conclusion

In this paper, we have proposed an efficient secure deduplication scheme with user-defined access control. Specifically, our scheme does not need to introduce an additional authorized server or use the hybrid cloud architecture to achieve the authorized deduplication. In our scheme, only the CSP can manage access rights on behalf of data owners without threatening data confidentiality. Besides, our scheme introduces the Bloom filter to efficiently complete the duplicate check. Detailed security analyses demonstrate that our scheme can achieve data confidentiality, access control, tag consistency and resistance to brute-force attacks at the same time. Further, extensive performance evaluations on file-level deduplication and chunk-level deduplication show the efficiency of our scheme, in terms of the effectiveness of deduplication, computational cost, communication overhead and storage cost.

References

[1] "Cisco global cloud index: Forecast and methodology, 2016-2021 white paper," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html.

[2] J. Gantz and D. Reinsel, "The digital universe decade-are you ready," https://hk.emc.com/collateral/analyst-reports/idc-digital-universe-are-you-ready.pdf.

[3] H. Biggar, "Experiencing data de-duplication: Improving efficiency and reducing capacity requirements," *The Enterprise Strategy Group.*, 2007.

[4] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," *TOS*, vol. 7, no. 4, pp. 14:1–14:20, 2012.

[5] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 40–47, 2010.

[6] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *ICDCS*, 2002, pp. 617–624.

[7] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A secure data deduplication scheme for cloud storage," in *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, 2014, pp. 99–118.

[8] P. Puzio, R. Molva, M. Önen, and S. Loureiro, "Cloudedup: Secure deduplication with encrypted data for cloud storage," in *IEEE 5th International Conference on Cloud Computing Technology and Science, CloudCom 2013, Bristol, United Kingdom, December 2-5, 2013, Volume 1*, 2013, pp. 363–370.

[9] S. Keelveedhi, M. Bellare, and T. Ristenpart, "Dupless: Server-aided encryption for deduplicated storage," in *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, 2013, pp. 179–194.

[10] M. Miao, J. Wang, H. Li, and X. Chen, "Secure multi-server-aided data deduplication in cloud computing," *Pervasive and Mobile Computing*, vol. 24, pp. 129–137, 2015.

[11] Y. Shin, D. Koo, J. Yun, and J. Hur, "Decentralized server-aided encryption for secure deduplication in cloud storage," *IEEE Trans. Services Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[12] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, 2013, pp. 296–312.

[13] X. Liu, W. Sun, W. Lou, Q. Pei, and Y. Zhang, "One-tag checker: Message-locked integrity auditing on encrypted cloud deduplication storage," in *2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017*, 2017, pp. 1–9.

[14] R. Chen, Y. Mu, G. Yang, and F. Guo, "BL-MLE: block-level message-locked encryption for secure large file deduplication," *IEEE Trans. Information Forensics and Security*, vol. 10, no. 12, pp. 2643–2652, 2015.

[15] S. Jiang, T. Jiang, and L. Wang, "Secure and efficient cloud data deduplication with ownership management," *IEEE Trans. Services Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[16] J. Li, X. Chen, X. Huang, S. Tang, Y. Xiang, M. M. Hassan, and A. Alelaiwi, "Secure distributed deduplication systems with improved reliability," *IEEE Trans. Computers*, vol. 64, no. 12, pp. 3569–3579, 2015.

[17] J. Li, Y. K. Li, X. Chen, P. P. C. Lee, and W. Lou, "A hybrid cloud approach for secure authorized deduplication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1206–1216, 2015.

[18] H. Cui, R. H. Deng, Y. LI, and G. Wu, "Attribute-based storage supporting secure deduplication of encrypted data in cloud," *IEEE Trans. Big Data*, pp. 1–1, 2017.

[19] Z. Yan, L. Zhang, W. Ding, and Q. Zheng, "Heterogeneous data storage management with deduplication in cloud computing," *IEEE Trans. Big Data*, pp. 1–1, 2017.

[20] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, 2007, pp. 535–554.

[21] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, 2005, pp. 325–341.

[22] B. H. Bloom, "Spacetime trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[23] X. Liu, K. R. Choo, R. H. Deng, R. Lu, and J. Weng, "Efficient and privacy-preserving outsourced calculation of rational numbers," *IEEE Trans. Dependable Sec. Comput.*, vol. 15, no. 1, pp. 27–39, 2018.

[24] J. Hur, D. Koo, Y. Shin, and K. Kang, "Secure data deduplication with dynamic ownership management in cloud storage," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 3113–3125, 2016.

[25] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Trans. Computers*, vol. 65, no. 8, pp. 2386–2396, 2016.

[26] Z. Yan, W. Ding, X. Yu, H. Zhu, and R. H. Deng, "Deduplication on encrypted big data in cloud," *IEEE Trans. Big Data*, vol. 2, no. 2, pp. 138–150, 2016.

[27] X. Yang, R. Lu, K.-K. R. Choo, F. Yin, and X. Tang, "Achieving efficient and privacy-preserving cross-domain big data deduplication in cloud," *IEEE Trans. Big Data*, vol. PP, no. 99, pp. 1–1, 2017.

[28] Y. Shin, D. Koo, and J. Hur, "A survey of secure data deduplication schemes for cloud storage systems," *ACM Comput. Surv.*, vol. 49, no. 4, pp. 74:1–74:38, 2017.

[29] F. Bao, R. H. Deng, and H. Zhu, "Variations of diffie-hellman problem," in *Information and Communications Security, 5th International Conference, ICICS 2003, Huhehaote, China, October 10-13, 2003, Proceedings*, 2003, pp. 301–312.

[30] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, 2006.

[31] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[32] A. Z. Broder and M. Mitzenmacher, "Survey: Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2003.

[33] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[34] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[35] J. Liu, N. Asokan, and B. Pinkas, "Secure deduplication of encrypted data without additional independent servers," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, 2015, pp. 874–885.

[36] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*. Kerkyra, Corfu, Greece, June 28 - July 1: IEEE, 2011, pp. 850–855.

[37] "Gmp library," https://gmplib.org.

[38] P. Anderson and L. Zhang, "Fast and secure laptop backups with encrypted de-duplication," in *Uncovering the Secrets of System Administration: Proceedings of the 24th Large Installation System Administration Conference, LISA 2010, San Jose, CA, USA, November 7-12, 2010*, 2010.

[39] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, "Message-locked encryption for lock-dependent messages," in *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, 2013, pp. 374–391.

[40] Y. Duan, "Distributed key generation for encrypted deduplication: Achieving the strongest privacy," in *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, CCSW '14, Scottsdale, Arizona, USA, November 7, 2014*, 2014, pp. 57–68.

[41] M. W. Storer, K. M. Greenan, D. D. E. Long, and E. L. Miller, "Secure data deduplication," in *Proceedings of the 2008 ACM Workshop On Storage Security And Survivability, StorageSS 2008, Alexandria, VA, USA, October 31, 2008*, 2008, pp. 1–10.

[42] D. Koo and J. Hur, "Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing," *Future Generation Comp. Syst.*, vol. 78, pp. 739–752, 2018.
[43] Y. Yang, H. Zhu, H. Lu, J. Weng, Y. Zhang, and K. R. Choo, "Cloud based data sharing with fine-grained proxy re-encryption," *Pervasive and Mobile Computing*, vol. 28, pp. 122–134, 2016.
[44] H. Tang, Y. Cui, C. Guan, J. Wu, J. Weng, and K. Ren, "Enabling ciphertext deduplication for secure cloud storage and access control," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016*, 2016, pp. 59–70.
[45] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, 2007, pp. 321–334.
[46] A. B. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, 2011, pp. 568–588.

Xue Yang received the Ph.D degree in Information and Communication Engineering from Southwest Jiaotong University, Chengdu, China, in 2019. She was a visiting student at the Faculty of Computer Science, University of New Brunswick, Canada, from 2017 to 2018. She is currently a Postdoctoral Fellow in the Tsinghua Shenzhen International Graduate School, Tsinghua University, China. Her research interests include big data security and privacy, applied cryptography and federated learning.

Rongxing Lu (S'09-M'10-SM'15) is an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before joining UNB in August 2016, he also worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Rongxing Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious "Governor Generals Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (Com-Soc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. He is presently a senior member of IEEE Communications Society. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise, and was the recipient of 8 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Vice-Chair (Publication) of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

Jun Shao received the Ph.D. degree from the Department of Computer Science and Engineering at Shanghai Jiao Tong University, Shanghai, China in 2008. He was a postdoc in the School of Information Sciences and Technology at Pennsylvania State University, USA from 2008 to 2010. He is currently a professor of the School of Computer Science and Information Engineering at Zhejiang Gongshang University, Hangzhou, China. His research interests include network security and applied cryptography.

Xiaohu Tang (M'04) received the B. S. degree in applied mathematics from the Northwest Polytechnic University, Xi'an, China, the M.S. degree in applied mathematics from the Sichuan University, Chengdu, China, and the Ph.D. degree in electronic engineering from the Southwest Jiaotong University, Chengdu, China, in 1992, 1995, and 2001 respectively.

From 2003 to 2004, he was a research associate in the Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology. From 2007 to 2008, he was a visiting professor at University of Ulm, Germany. Since 2001, he has been in the School of Information Science and Technology, Southwest Jiaotong University, where he is currently a professor. His research interests include coding theory, network security, distributed storage and information processing for big data.

Dr. Tang was the recipient of the National excellent Doctoral Dissertation award in 2003 (China), the Humboldt Research Fellowship in 2007 (Germany), and the Outstanding Young Scientist Award by NSFC in 2013 (China). He serves as Associate Editors for several journals including IEEE TRANSACTIONS ON INFORMATION THEORY and IEICE Transactions on Fundamentals, and served on a number of technical program committees of conferences.

Ali A. Ghorbani (SM'–) has held a variety of positions in academia for the past 35 years. He has been the Dean of the Faculty of Computer Science since 2008. He is currently the Canada Research Chair (Tier 1) in Cybersecurity. He is also the Director of the Canadian Institute for Cybersecurity. He has developed a number of technologies that have been adopted by high-tech companies. He co-founded two startups, Sentrant and EyesOver in 2013 and 2015, respectively. He is the Co-Inventor on three awarded patents in the area of network security and web intelligence and has published over 200 peer-reviewed articles during his career. He has supervised over 160 research associates, post-doctoral fellows, and graduate and undergraduate students during his career. His book, Intrusion Detection and Prevention Systems: Concepts and Techniques, (Springer, 2010). Since 2010, he has obtained over 10M to fund six large multi-project research initiatives. He was twice one of the three finalists for the Special Recognition Award at the 2013 and 2016 New Brunswick KIRA Award for the knowledge industry. In 2007, he received the University of New Brunswick's Research Scholar Award. He is the Co-Editor-In-Chief of *Computational Intelligence Journal*.