SimLESS: A Secure Deduplication System Over Similar Data in Cloud Media Sharing

Mingyang Song, Zhongyun Hua, *Senior Member, IEEE*, Yifeng Zheng, *Member, IEEE*, Tao Xiang, *Senior Member, IEEE*, and Xiaohua Jia, *Fellow, IEEE*

***Abstract*— With the growing popularity of cloud computing, sharing media data through the cloud has become a common practice. Due to high information redundancy, media data take up a significant amount of storage space. Moreover, similar media data may have the same visual effect, resulting in unnecessary duplication. Thus, it can greatly improve the cloud storage efficiency by performing deduplication to the similar media data stored on the cloud. However, data privacy is a growing concern in cloud-based service. In this paper, we present SimLESS, a secure deduplication system for similar data in cloud media sharing. SimLESS allows the cloud to perform deduplication over the encrypted similar media data of different distributors while protecting the confidentiality and ownership of the data. When uploading a media file, SimLESS allows the distributor to set a distance threshold, and the cloud performs deduplication only when there is a file on the cloud whose distance from the file being uploaded is smaller than the threshold. Additionally, we provide fine-grained access control for distributors to ensure that only authorized media consumers can access the data. Furthermore, our system prevents any distributor from claiming ownership of a media file using only the tag of a similar file. We formally analyze the security of SimLESS and implement a system prototype to evaluate its performance. Our experimental results demonstrate that the computation and communication costs of SimLESS are practically affordable.**

Manuscript received 31 May 2023; revised 13 August 2023 and 24 January 2024; accepted 24 March 2024. Date of publication 27 March 2024; date of current version 7 May 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3103500, in part by the National Natural Science Foundation of China under Grant 62071142, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515010714, in part by Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies under Grant 2022B1212010005, in part by Shenzhen Science and Technology Program under Grant ZDSYS20210623091809029, in part by the Research Grants Council of Hong Kong under Grant R1012-21 and Grant CityU 11213920. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Antonino Nocera. *(Corresponding author: Zhongyun Hua.)*

Mingyang Song and Yifeng Zheng are with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Guangdong 518055, China (e-mail: songmingyang2022@gmail.com; yifeng.zheng@hit.edu.cn).

Zhongyun Hua is with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Guangdong 518055, China, and also with Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Shenzhen, Guangdong 518055, China (e-mail: huazyum@gmail.com).

Tao Xiang is with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: txiang@cqu.edu.cn).

Xiaohua Jia is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China, and also with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China (e-mail: csjia@cityu.edu.hk).

Digital Object Identifier 10.1109/TIFS.2024.3382603

***Index Terms*— Similar media data, secure deduplication, access control, cloud media sharing.**

I. Introduction

THE rapidly developed cloud computing technologies have greatly facilitated the communication between cloud users [1], [2]. A cloud user can receive information and share his/her data with other users through cloud servers at anytime and anywhere [3], [4]. With the easy accessibility and rich information of media data (e.g., images and videos), users tend to browse and share media data through cloud services. Nowadays, many cloud service providers, such as Google Drive [5], Dropbox [6], and Alibaba Cloud [7], provide users with data sharing services, where a distributor can upload his/her data and share them to some specified consumers. Among these data, there exist a significant portion of similar media data. According to the report in [8], about 0.25 billion images are uploaded on Facebook daily and around 14%-52% data crawled from a single trail are similar content. Besides, due to the high information redundancy in media data, similar media data often present the same visual effect. Thus, deduplicating similar media data not only benefits the storage efficiency in cloud media sharing, but also reduces communication cost for users since they no longer need to upload similar media files.

Coming with the cloud-based service are the acute privacy concerns [9], [10]. When uploading the plaintext media data to the cloud, the private media data are easily gained by the cloud and even by other unauthorized consumers, which seriously threatens the confidentiality of the media data and even the privacy of the media data distributor [11], [12]. Thus, considering the data confidentiality, the media data distributors may tend to encrypt their data before uploading them to the cloud [13], [14]. Besides, the access permission management should be considered in cloud media sharing. The scheme should guarantee that only the consumers specified by the media data distributors can access the media content. As a result, a secure solution is required in cloud media sharing to protect the confidentiality of media data as well as allow the cloud to deduplicate the similar media data.

To date, many secure deduplication schemes over encrypted data have been developed and these schemes can be divided into two categories: deduplication over identical data [15], [16], [17], [18], [19], [20] and deduplication over similar

data [21], [22], [23]. To support secure deduplication over identical data, the users are required to encrypt their data using some deterministic encryption strategies, including message-locked encryption (MLE) [15], [16], [17] and key server-aided MLE [18], [19], [20]. These encryption strategies derive encryption key from the plaintext itself and thus users can encrypt identical plaintexts to the same ciphertexts such that the cloud can perform deduplication. However, these schemes cannot support similar data deduplication, as slightly different plaintexts will be encrypted to completely different ciphertexts. Thus, it requires more delicate designs to achieve secure deduplication over similar data.

To enable secure deduplication over similar data, several schemes have been developed [21], [22], [23] but they still have some drawbacks. The scheme in [21] cannot support data deduplication across different groups of users. The scheme in [22] requires previous users to stay online to perform password-authenticated key exchange (PAKE) with subsequent users, which is practically unreasonable. To achieve similar data deduplication in the ciphertext domain, the scheme in [23] requires users to encrypt their files using only exclusive-OR (XOR) operation such that similar files can be encrypted to similar ciphertexts for deduplication. Since a same key is used to encrypt all the blocks of a file, the XOR operation cannot provide high-security protection for the file content. Recently, Huang et al. [11] proposed SMACD to perform secure deduplication in cloud media sharing. But SMACD is designed for identical media data and cannot deduplicate encrypted similar media data. Besides, it does not consider the media data ownership security, which means that a media distributor can claim the ownership of a file to the cloud using only the tag of the file. Our scheme aims to achieve secure deduplication for similar data and protect their ownership security in cloud media sharing.

To achieve secure deduplication over similar data in cloud media sharing, the following challenges must be addressed for practical usability. (1) The cloud should check the similarity between a new media file with other encrypted media files stored on the cloud without knowing any content of the file. For practical consideration, the similarity checking should not rely on the previous distributors who uploaded similar files, as they may go offline after uploading their data. (2) Subsequent distributors and authorized consumers must be able to correctly recover the encryption key. The cloud only stores the ciphertext uploaded by the initial distributor and does not keep the ciphertexts of the subsequent distributors. Therefore, valid distributors and authorized consumers must be able to recover the encryption key used by the initial distributor to correctly decrypt ciphertexts. (3) When a distributor claims ownership of a file stored on the cloud, the cloud should be able to verify that the distributor indeed owns a similar media file. However, previous proof of ownership (PoW) techniques [24], [25], [26] cannot be used to verify the ownership of similar media files, as they prevent a valid subsequent distributor with a similar media file from passing the verification process.

In this paper, we introduce SimLESS, a new secure deduplication system designed specifically for similar media data in cloud media sharing. We develop a fuzzy duplication detection strategy that enables the cloud to compare the similarity-preserving hash value of a new media file with those of the encrypted media files stored on the cloud. To prevent unnecessary deduplication, the distributor can set a distance threshold, and the cloud performs deduplication only when the distance between the file to be uploaded and at least one media file on the cloud is smaller than the threshold. Thus, the distributor can determine the fuzzy deduplication degree based on the file's importance. We also develop a secure encryption key sharing mechanism using the ciphertext-policy attribute-based encryption (CP-ABE) [27], [28] and fuzzy extractor [29]. This mechanism allows all valid distributors and their authorized consumers to recover the encryption key used by the initial distributor, ensuring that they can decrypt the ciphertexts stored on the cloud correctly. Furthermore, we design a fuzzy PoW protocol based on fuzzy extractor to protect media data ownership security. This protocol enables the cloud to verify that a subsequent distributor indeed owns a similar media file.

The contributions of this paper are summarized as follows.

- We propose a novel secure deduplication system for similar media data, which, to our best knowledge, is the first system that supports deduplication over encrypted similar data in cloud media sharing.
- Our system allows the distributor of a media file to set a distance threshold between his/her file and the media files stored on the cloud, providing great feasibility for determining the degree of similar deduplication based on the file's importance.
- Furthermore, we conduct a formal analysis of our scheme from the aspects of media data recovery and PoW for similar media data. Our security proofs demonstrate that our scheme protects the confidentiality of media content and media data ownership security.

The rest of the paper is organized as follows. Section II introduces some preliminaries used in this paper. Section III presents the problem formalization of SimLESS. Section IV presents the design of SimLESS in detail. Section V-B provides the formal analysis of correctness and security of SimLESS. We implement a prototype of SimLESS on a commercial cloud platform and evaluate its performance in Section VI. Section VII reviews some related work and we conclude our study in Section VIII.

II. Preliminaries

This section presents some preliminaries used in our SimLESS and Table I lists some important notations in this paper.

A. Bilinear Pairing of Composite Order

For a given security parameter κ, run the composite bilinear parameter generator $\mathcal{G}en(\kappa)$ and get $(p, q, N, \mathbb{G}_1, \mathbb{G}_2, \mathrm{e})$, where p, q are two primes with κ-bit length, $N = p \cdot q$ and $\mathrm{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear pairing [30] with the following three properties:

- *Bilinearity*: $\mathrm{e}(\delta_1^a, \delta_2^b) = \mathrm{e}(\delta_1, \delta_2)^{ab}$ for $\forall \delta_1, \delta_2 \in \mathbb{G}_1$ and $\forall a, b \in \mathbb{Z}_N$.

TABLE I
IMPORTANT NOTATIONS

Notation	Description
U	The universe set of attributes.
SK_{at_i}	The attribute secret key of attribute at_i.
$\mathcal{T}$	An access policy tree.
$\mathbb{M}$	A generation matrix of linear secret sharing.
$\rho(\cdot)$	A function mapping integers to attributes.
PK	The set of public parameters.
F	A plaintext media file.
C	The ciphertext of symmetric encryption.
CT	The ciphertext of attribute encryption.
PA	The auxiliary data set of PoW.
tag	The media file tag.
tk	The trapdoor key.
$\mathbb{G}_1, \mathbb{G}_2$	Two multiplicative cyclic groups.
$\mathbb{Z}_N$	A residue class ring.
κ	The system security parameter.
T	A threshold of Hamming distance.
$\mathrm{Dis}_{\mathrm{Ham}}(\cdot,\cdot)$	The Hamming distance of two inputs.

- *Non-degeneracy*: $\mathrm{e}(g, g)$ is a generator of $\mathbb{G}_2$ if g is a generator of $\mathbb{G}_1$.
- *Computability*: For $\forall \delta_1, \delta_2 \in \mathbb{G}_1$, there exists an algorithm to compute $\mathrm{e}(\delta_1, \delta_2) \in \mathbb{G}_2$.

Definition 1 (The Bilinear Diffie-Hellman (BDH) Assumption [31]): Given $g, g^a, g^b, g^c \in \mathbb{G}_1$ (for unknown $a, b, c \in \mathbb{Z}_N$), there is no probabilistic polynomial-time algorithm that can compute $\mathrm{e}(g, g)^{a \cdot b \cdot c}$ with non-negligible advantage.

Definition 2 (The Computational Diffie-Hellman (CDH) Assumption [32]): Given $g, g^a, g^b \in \mathbb{G}_1$ (for unknown $a, b \in \mathbb{Z}_N$), there is no probabilistic polynomial-time algorithm that can compute $g^{a \cdot b}$ with non-negligible advantage.

B. Secure Sketch and Fuzzy Extractor

Let $\mathcal{M}$, m, t, θ, and ϵ be a metric space (i.e., input space) with the distance function dis, min-entropy of input data, maximum distance between two inputs, bit length of a random string, and entropy loss of secure sketch, respectively.

Definition 3 (Secure Sketch [29]): An $(\mathcal{M}, t, m, \theta, \epsilon)$-secure sketch consists of a sketch procedure (SS) *and a recover procedure* (Rec)*, and has the following properties*:

- $\mathrm{SS}(w, r) \rightarrow \{s\}$. *The sketch procedure takes inputs as $w \in \mathcal{M}$ and random string $r \in \{0,1\}^\theta$, and returns a string $s \in \{0,1\}^\theta$.*
- $\mathrm{Rec}(w', s) \rightarrow \{w\}$. *The recover procedure takes inputs as $w' \in \mathcal{M}$ and s. The output w is correct if* $\mathrm{dis}(w, w') \leq t$. *Otherwise, the output is incorrect.*

Any adversary owning s can recover w with a probability smaller than $2^{-(m-\epsilon)}$.

To obtain a secure sketch, a practical way is to utilize a $(\sigma, \theta, 2t+1)$-error correcting code algorithm C, where σ is the code length, θ is the codeword number, and $2t+1$ is the minimum distance. On input r and w, the secure sketch computes the encoded $c = \mathrm{C}(r)$ and sets $s = w - c$.

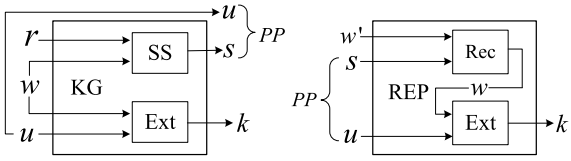


Fig. 1. Depiction of fuzzy extractor.

To compute $\mathrm{Rec}(w', s)$, one can extract $c' = w' - s$, and then decode c'. Because $\mathrm{dis}(w, w') \leq t$, so is $\mathrm{dis}(c, c') \leq t$, and the correct r can be recovered. Finally, we can obtain $w = s + \mathrm{C}(r)$.

The fuzzy extractor constructed based on the above secure sketch defined as the Definition 3 enables the generation of the same string k from two similar messages w and w'. Fig. 1 depicts the syntax of fuzzy extractor.

Definition 4 (Fuzzy Extractor [29]): An $(\mathcal{M}, t, \theta, \sigma, \epsilon, m)$-fuzzy extractor consists of a key generate algorithm (KG) *and a reproduce algorithm* (REP)*, with the following properties*:

- $\mathrm{KG}(w, r, u) \rightarrow \{k, PP\}$: *It takes inputs as $w \in \mathcal{M}$ and two random strings $r, u \in \{0,1\}^\theta$, and outputs public parameters $PP = \{u, s = \mathrm{SS}(w, r)\}$ and an extracted string $k = \mathrm{Ext}(w, u)$.*
- $\mathrm{REP}(w', PP) \rightarrow \{k\}$: *It first takes inputs as $w' \in \mathcal{M}$ and the string s, and recovers $w = \mathrm{Rec}(w', s)$. Then it outputs the same string $k = \mathrm{Ext}(w, u)$.*

The fuzzy extractor has the following properties: (1) It can guarantee that $\mathrm{REP}(w', PP) \rightarrow \{k\}$ if $\mathrm{dis}(w, w') \leq t$ and $\mathrm{KG}(w, r, u) \rightarrow \{k, PP\}$; (2) For the w with min-entropy m, any adversary can only obtain k with a probability smaller than $2^{-(m-\epsilon)}$ even knowing PP; (3) It can execute KG and REP in polynomial time.

C. Similarity-Preserving Hash

The similarity-preserving hash function $\tilde{\mathrm{h}}$ is a special hash function. The similarity of two files can be preserved in their hash values [23]. The similarity-preserving hash function has the following properties:

- Digest similarity: The generated digests are similar if the input data are similar.
- One-wayness: It is difficult to recover the preimage using the digest.

The similarity-preserving hash function can be used to identify two similar files. To implement a similarity-preserving hash function, one can first extract some features of the input data, and then aggregate these features to a binary string. The binary string can be used as the similarity-preserving hash value.

D. Linear Secret Sharing Scheme

We give the formal definition of the linear secret sharing scheme (LSSS) as follows.

Definition 5 (Linear Secret Sharing Schemes [33]): A secret sharing scheme over a set of parties is called LSSS over $\mathbb{Z}_N$ if it satisfies the following conditions.

- *The shares are over $\mathbb{Z}_N$.*

- *The operation of sharing the secret μ into γ shares is performed as*

$$\boldsymbol{\lambda} = \mathbb{M} \cdot \boldsymbol{v},$$

where $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_\gamma)$ are γ generated shares, $\boldsymbol{v} = (\mu, v_2, \cdots, v_\iota)$ and $v_2, \cdots, v_\iota$ are random integers over $\mathbb{Z}_N$, and $\mathbb{M}$ is the generation matrix of size $\gamma \times \iota$. A party $\rho(i)$ holds the share (i, λ_i).

LSSS is usually used in CP-ABE, in which the secret is shared over a set of attributes. An LSSS has the linear reconstruction property. We assume that S is an authorized attribute set and I is the indices set of the attributes in S, namely $I = \{i | \rho(i) \in S\}$. Then, for the valid shares $\{\lambda_i\}_{i \in I}$, there exist constants $\{w_i \in \mathbb{Z}_N\}_{i \in I}$ to recover the secret as $\mu = \sum_{i \in I} w_i \lambda_i$ [33]. Furthermore, the constants $\{w_i\}_{i \in I}$ can be found in polynomial time corresponding to the size of the matrix $\mathbb{M}$.

The access structures of CP-ABE can be represented as access trees, where the interior nodes are AND and OR gates, and the leaf nodes are the attributes [27]. An access tree structure can also be described as a generation matrix of LSSS and a function ρ mapping a row index i to attribute $\rho(i)$. This kind of representation is named as LSSS access structure. Some standard techniques have been developed to convert an access tree into its corresponding LSSS access structure such as the method in [34]. The number of rows in the generation matrix of LSSS equals to the number of leaf nodes in the access tree.

III. Problem Statement

A. System Model and Definitions

Fig. 2 shows the system model of our SimLESS, which consists of four types of entities: media distributor, media consumer, attribute authority, and cloud.

- **Media distributor:** A media distributor shares media files with his/her authorized media consumers through the cloud. To protect data confidentiality, the media distributor encrypts his/her media files before uploading. Different distributors may share similar media files under different access policies. A distributor can set a distance threshold for each file. If the distance between a shared file and a similar media file stored on the cloud is smaller than the threshold, the distributor can replace his/her file with the stored file to reduce his/her communication bandwidth and storage cost. The distributor who uploads a new file is referred to as the initial distributor, while the one who uploads a similar file is called the subsequent distributor.
- **Media consumer:** A media consumer can access and restore the media data that is authorized to him/her from the cloud.
- **Attribute authority:** The attribute authority is responsible for authorizing attributes to media consumers. It generates an attribute secret key for consumers with the corresponding attribute. The attribute authority generates the public parameters of the system.
- **Cloud:** The cloud receives encrypted media data from media distributors and distributes them to authorized media consumers. It also performs deduplication over encrypted similar media data to reduce storage costs.

We can formally define our SimLESS over the aforementioned system model.

Definition 6: SimLESS is a nine-tuple of polynomial-time algorithms (**Setup, TagGen, FDupDet, Encrypt, Decrypt, RelVerify, FPoW.AuxGen, FPoW.Proof**)*, which are defined as follows*:

- **Setup**$(\kappa, U) \rightarrow \{PK, msk, \{SK_{at_i}\}_{at_i \in U}\}$. *The algorithm takes a security parameter κ and the universe set U of attributes as inputs, and outputs the set of public parameters PK, the master secret key msk, and the attribute secret keys $\{SK_{at_i}\}_{at_i \in U}$.*
- **TagGen**$(F, PK) \rightarrow \{tag_F\}$. *The algorithm takes the media file F and public parameters PK as inputs, and outputs a tag tag_F of the media file.*
- **FDupDet**$(T_1, T_2, tag_F) \rightarrow \{(1/0, \perp)/(1, tag_{F'})\}$. *The algorithm takes the file tag tag_F, the distance threshold T_1, and the reliability threshold T_2 set by the media distributor as inputs. It outputs $(1, \perp)$ if the cloud stores a similar file F' with a reliability score exceeding T_2. It outputs $(1, tag_{F'})$ if the similar file F' has a reliability score lower than or equal to T_2. If there is no similar file stored on the cloud, the algorithm outputs $(0, \perp)$.*
- **Encrypt**$(F, PK, \mathcal{T}) \rightarrow \{C_F, CT_F, PP_0\}$. *The algorithm takes the media file F, the public parameters PK and an access policy tree $\mathcal{T}$ as inputs, and outputs the media data ciphertext C_F, its encryption key's ciphertext CT_F encrypted under the access policy $\mathcal{T}$, and the output parameters PP_0 of fuzzy extractor.*
- **Decrypt**$(C_F, CT_F, PK, \{SK_{\rho(i)}\}_{i \in I}) \rightarrow \{F\}$. *The algorithm takes the secret keys $\{SK_{\rho(i)}\}_{i \in I}$ of attributes $\{\rho(i)\}_{i \in I}$ that satisfy the access policy $\mathcal{T}$, the ciphertexts $\{C_F, CT_F\}$, and the public parameters PK as inputs, and outputs the plaintext media file F.*
- **RelVerify**$(F, PP_0, C_{F'}, CT_{F'}) \rightarrow \{dis\}$. *The algorithm takes the media file F, the output parameters PP_0 of fuzzy extractor, and the ciphertexts $C_{F'}$, $CT_{F'}$ as inputs, and outputs the real distance dis between F and F'.*
- **FPoW.AuxGen**$(F, PK) \rightarrow \{PA_F\}$. *The algorithm takes the media file F and the public parameters PK as inputs, and outputs the auxiliary data PA_F for fuzzy PoW.*
- **FPoW.Proof**$(F, PK, PA_{F'}) \rightarrow \{1/0\}$. *The algorithm takes media file F, the public parameters PK, and the auxiliary data $PA_{F'}$ of PoW as inputs. The algorithm outputs the ownership checking result.*
- **Dec.AuxGen**$(F, \mathcal{T}, PK, PP_0) \rightarrow \{KA_F\}$. *The algorithm takes media file F, an access policy tree $\mathcal{T}$, the public parameters PK, and the output parameter PP_0 of fuzzy extractor as inputs. The algorithm outputs the auxiliary data KA_F for key recovery.*

B. Threat Model and Security Definitions

The threats of our model are from the cloud, media distributors, and media consumers. We discuss these possible threats in our system as follows:

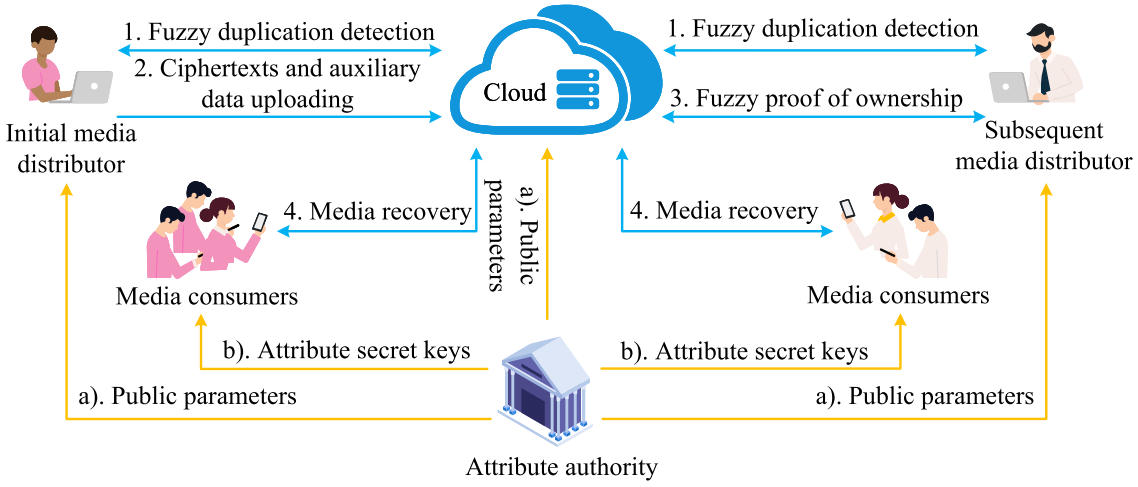


Fig. 2. System model.

- **Cloud:** The cloud is expected to execute the protocols honestly. However, it may attempt to access the private contents of its stored encrypted media files using its background knowledge.
- **Media distributor:** An initial media distributor may launch the duplicate-faking attack (DFA) to disrupt the consistency between the media file and its file tag, aiming to deceive subsequent distributors owning similar files to obtain a falsified copy. A subsequent distributor may launch the hash-only attack [35], which involves the distributor claiming ownership of a media file stored on the cloud using only the tag of a similar media file.
- **Media consumer:** A media consumer may attempt to gain unauthorized access to the private contents of the encrypted media files. The media consumer may do so by first intercepting the communication channel and obtaining the media data ciphertexts, and then recovering the contents of the ciphertexts using his/her background knowledge.

Our SimLESS aims to achieve: (1) PRV-CDA security [15] (data privacy under chosen distribution attacks) with the assumption that the plaintext space is unpredictable; (2) media date ownership security under the hash-only attack; (3) tag consistency under DFA.

Definition 7: The PRV-CDA security of SimLESS can be modeled by a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

- Setup*: The challenger $\mathcal{C}$ runs* **Setup**(κ, U) *with the inputs κ, U, and sends the output PK to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ chooses several sets of attributes $\{A_i\}_{1\le i\le \ell}$ and sends them to the challenger. Then $\mathcal{C}$ sends attribute secret keys $\{SK_{at}\}_{at\in A_i}$, $(1 \le i \le \ell)$ to $\mathcal{A}$.*
- Challenge*: The adversary $\mathcal{A}$ chooses an access policy tree $\mathcal{T}$ such that none of the sets $\{A_i\}_{1\le i\le \ell}$ satisfying the access policy. The challenger $\mathcal{C}$ randomly picks a bit $\eta \in \{0, 1\}$ and chooses two media files m_0 and m_1. $\mathcal{C}$ executes* **Encrypt**$(m_\eta, PK, \mathcal{T})$ *to generate C_{m_η} and CT_{m_η} under the access policy $\mathcal{T}$, and computes tag_{m_η} using the* **TagGen**(m_η, PK) *function. Then $\mathcal{C}$ sends $(C_{m_\eta}, CT_{m_\eta}, tag_{m_\eta})$ to $\mathcal{A}$.*
- Guess*: The adversary $\mathcal{A}$ outputs a guess $\hat{\eta}$ of η.*

The advantage of the adversary $\mathcal{A}$ in the above game can be defined as Eq. (1). The media content is secure if $\mathcal{A}$ has a negligible advantage in this game.

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{PRV-CDA}}(\kappa) = 2 \cdot \Pr[\mathcal{A}(\hat{\eta} = \eta)] - 1 \tag{1}$$

Definition 8: SimLESS can achieve media data ownership security if any probabilistic polynomial-time adversary $\mathcal{A}$ has negligible advantage $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{OS}}(\kappa)$ to win the game $\mathrm{Game}_{\mathcal{A}}^{\mathrm{OS}}$ defined in Fig. 3, where the message space $\mathcal{M}(\kappa)$ is sufficiently large such that plaintexts are unpredictable.

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{OS}}(\kappa) = \Pr[\mathrm{Game}_{\mathcal{A}}^{\mathrm{OS}}(\kappa) = 1] \tag{2}$$

Definition 9: SimLESS is tag consistent under DFA if for any probabilistic polynomial-time adversary $\mathcal{A}$, the advantage $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{TC}}(\kappa)$ of the adversary winning the game $\mathrm{Game}_{\mathcal{A}}^{\mathrm{TC}}$ defined in Fig. 3 is negligible.

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{TC}}(\kappa) = \Pr[\mathrm{Game}_{\mathcal{A}}^{\mathrm{TC}}(\kappa) = 1] \tag{3}$$

We will theoretically prove the security of our SimLESS in Section V-B.

C. Design Goals

In this study, we aim to achieve fuzzy deduplication over encrypted similar media data in cloud media sharing, which achieves the following specific goals.

- **Functionality:** Our SimLESS aims to provide secure deduplication over similar media data. The cloud is able to perform deduplication over encrypted similar media data shared by different media distributors. Furthermore, a media distributor can set a distance threshold for each of his/her files to determine the maximum allowable deduplication distance between the file with the files stored on the cloud.
- **Security:** The security goals of SimLESS include achieving PRV-CDA security, ownership security, and resistance

Ownership security game: $\text{Game}_{\mathcal{A}}^{\text{OS}}$	Tag consistency game: $\text{Game}_{\mathcal{A}}^{\text{TC}}$
$(PK, msk, \{SK_{at_i}\}_{at_i \in U}) \leftarrow \text{Setup}(\kappa, U)$	$(PK, msk, \{SK_{at_i}\}_{at_i \in U}) \leftarrow \text{Setup}(\kappa, U)$
$F \leftarrow \mathcal{M}(\kappa)$	$(F, F', C_{F'}, CT_{F'}, PP_0, tag_{F'}) \leftarrow \mathcal{A}(PK)$
$PA_F \leftarrow \text{FPoW.AuxGen}(F, PK)$	If $(F = \perp) \vee (F' = \perp) \vee (C_{F'} = \perp) \vee (CT_{F'} = \perp)$
$tag_F \leftarrow \text{TagGen}(F, PK)$	$\vee (PP_0 = \perp) \vee (tag_{F'} = \perp)$ Ret false
$F' \leftarrow \mathcal{A}(PK, tag_F, \mathcal{M})$	$tag_F \leftarrow \text{TagGen}(F, PK)$
$r \leftarrow \{0,1\}^{\theta}, u \leftarrow \{0,1\}^{\theta}$	$T \leftarrow \text{Dis}_{\text{Ham}}(tag_F, tag_{F'})$
$(k, PP) \leftarrow \text{KG}(F, r, u)$	$T' \leftarrow \text{Dis}_{\text{Ham}}(\tilde{h}(F), \tilde{h}(F'))$
If $k \neq \text{REP}(F', PP) \wedge$	If $T \geq \text{Re}$ lVerify$(F, PP_0, C_{F'}, CT_{F'})$
$1 \leftarrow \text{FPoW.Rroof}\ (F', PK, PA_F)$	$\wedge (T' > T)$ Ret true
Ret true	Else Ret false
Else Ret false	

Fig. 3. Security games for ownership security and tag consistency.

to DFA. (1) PRV-CDA security indicates that neither the cloud nor unauthorized consumers can access the private media content using their background knowledge. (2) Data ownership security guarantees that a distributor cannot claim the ownership of a media file using only the tag of a similar media file. (3) The resistance to DFA means that the scheme can safeguard subsequent media distributors from losing their media files under DFA.

IV. The Design of SimLESS

A. Design Overview

Considering the design goals in Section III-C, three main issues should be carefully designed in SimLESS, including fuzzy duplication detection over encrypted similar data, encryption key sharing, and media data ownership verification.

To achieve fuzzy duplication detection, the cloud needs to detect the similarity between a new media file and the encrypted media file stored on the cloud utilizing their tags. Therefore, we use the similarity-preserving hash value of the media file as the file tag. When the cloud receives a tag and a distance threshold of a media file from a distributor, it can obtain the distance between the new media file and each of its stored encrypted media files by comparing their tags. Then, it can detect duplication by checking whether the distance is less than the threshold. If the threshold is smaller than all the distances, the cloud can determine that there is no similar file stored on the cloud. Otherwise, the cloud finds the media file that has the smallest distance with the new file and regards it as the similar file of the new file.

Sharing the encryption key with authorized consumers can be challenging for a distributor. For similar media files, the cloud stores only the encrypted media file uploaded by the initial distributor, and subsequent distributors need to share the encryption key used by the initial distributor with their authorized consumers. However, the subsequent distributors do not know the encryption key of the initial distributor. To address this issue, our design includes a trapdoor implemented by the initial distributor when encrypting the encryption key. This allows more authorized access policies to be efficiently and securely included. In order for subsequent distributors to add their authorized access policies, we utilize a fuzzy extractor to enable the initial distributor and subsequent distributors to extract the same trapdoor key from their similar media files. Subsequent distributors can then use this trapdoor key to generate some auxiliary data, which can be used by their authorized consumers to recover the encryption key from the ciphertext of encryption key.

To ensure the security of media date ownership, it is important for the cloud to verify that a subsequent distributor actually owns a similar media file that matches the claimed file stored on the cloud. Previous PoW techniques were mainly designed for verifying the ownership of identical files by checking if a user owns the same ciphertext with the one stored on the cloud. However, for similar files, a subsequent distributor with a similar media file cannot generate the same ciphertext as the one stored on the cloud, which makes previous PoW methods ineffective. To address this issue, the initial distributor uses a fuzzy extractor to generate a secret seed from his/her media file, and then uses the seed to generate some auxiliary data that can be used by the cloud to verify subsequent distributor ownership. If a subsequent distributor does own a similar media file, he/she can use the same fuzzy extractor to generate the same secret seed from his/her file and computes a correct ownership proof using the seed. However, if the subsequent distributor does not have a similar media file, he/she cannot obtain the same secret seed and will fail the cloud's ownership verification.

The most direct way to resist DFA is to verify the consistency between the file tag and ciphertext uploaded by the initial media distributor. However, this bcomes challenging in the context of fuzzy deduplication, because it requires preserving the similarity of plaintexts in both file tags and ciphertexts. As a result, our scheme achieves DFA resistance indirectly by assigning a reliability score to each file stored on the cloud. When a media distributor uploads a file for the first time, this file's reliability score is set to 0. When a subsequent media distributor uploads a file, he/she can set a reliability threshold based on the importance of the file, and deduplication occurs if the cloud stores a similar file with a reliability score exceeding this threshold. If the cloud's similar file has a reliability score lower than or equal to the threshold, the subsequent media distributor downloads and decrypts its ciphertext, and then verifies its similarity with the file to be uploaded. If the

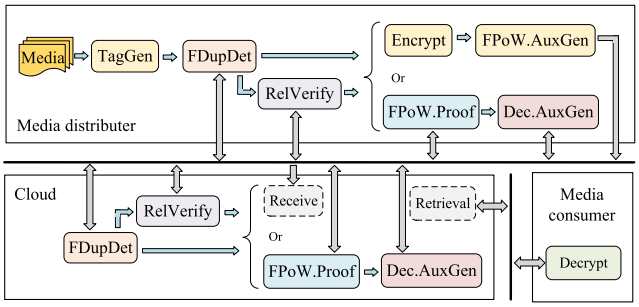


Fig. 4. System workflow of SimLESS.

verification passes, deduplication is allowed and the reliability score of the similar file is increased by 1. Otherwise, the cloud treats the file to be uploaded as a new file and set its reliability score to 0.

B. System Workflow

Fig. 4 demonstrates the system workflow of our SimLESS and we present each of the steps as follows.

- The attribute authority executes **Setup** and generates public parameters, its master secret key and secret keys of attributes. We do not show this process in Fig. 4, as it is executed only once at the beginning of the system.
- When a media distributor shares a media file to his/her authorized consumers, he/she first executes **TagGen** to generate a tag of the file, and then executes **FDupDet** to perform fuzzy duplication detection by interacting with the cloud.
- If the media file to be uploaded is similar to a file stored on the cloud, but the cloud's similar file has a reliability score lower than or equal to the threshold set by the media distributor, the cloud and the media distributor executes **RelVerify** to check the reliability of the similar file. If the verification fails, the cloud treats the file to be uploaded as a new file and sets its reliability score to 0. The media distributor executes the data uploading process as an initial media distributor. Otherwise, if the verification passes, the media distributor executes the interaction process with the cloud, behaving as a subsequent media distributor.
- If the media file to be uploaded is similar to a media file stored on the cloud and the similar media file has a reliability score exceeding the threshold set by the media distributor, deduplication is permitted and the media distributor is regarded as a subsequent media distributor. The cloud executes **FPoW.Proof** to verify the ownership of the distributor using the auxiliary data generated and uploaded by the initial distributor. If the ownership verification is valid, the distributor also generates some auxiliary data for his/her authorized consumers to recover the encryption key by performing the **Dec.AuxGen** algorithm with the cloud.
- If the media file to be uploaded is not similar to any media file stored on the cloud, the media distributor is regarded as an initial media distributor. The distributor performs **Encrypt** to encrypt the encryption key and media file. The distributor also performs **FPoW.AuxGen** to generate some auxiliary data that can be used by the cloud to verify the ownership of other subsequent distributors.
- The authorized consumers can download the ciphertexts of the encryption key and media file. They execute **Decrypt** to recover the encryption key and decrypt the ciphertext of media file using the key.

C. Detailed Design

We present the design of each algorithm shown in the Definition 6 in detail.

1) System Setup: In this process, the attribute authority runs **Setup**$(\kappa, U) \rightarrow \{PK, msk, \{SK_{at_i}\}_{at_i \in U}\}$ to generate public parameters, its master secret key and attribute secret key of each attribute $at_i \in U$.

- The attribute authority runs the composite bilinear parameter generator $\mathcal{G}en(\kappa)$ and obtains $(p, q, N, \mathbb{G}_1, \mathbb{G}_2, \mathrm{e})$, where κ is a randomly selected security parameter.
- It randomly selects a generator $g \in \mathbb{G}_1$ and three exponents $\alpha, \beta, \tau \in \mathbb{Z}_N$, and then randomly chooses a generator $g_{at_i} \in \mathbb{G}_1$ for each attribute $at_i \in U$.
- It computes its master secret key $msk = g^{\alpha}$, public parameters $\mathrm{e}(g, g)^{\alpha}$, g^{β}, and secret keys $sk^{(1)}_{attr} = g^{\alpha} g^{\tau\beta}$, $sk^{(2)}_{attr} = g^{\tau}$ and $sk_{at_i} = g^{\tau}_{at_i}$ for each $at_i \in U$. For each attribute $at_i \in U$, it sends the secret key $SK_{at_i} = \{sk^{(1)}_{attr}, sk^{(2)}_{attr}, sk_{at_i}\}$ to its corresponding media consumers through secret channels. It also chooses an integer ω as the degree of polynomials in fuzzy PoW. The public parameters are $PK = \{N, g, \mathrm{e}(g, g)^{\alpha}, g^{\beta}, \{g_{at_i}\}_{at_i \in U}, \omega\}$.
- It chooses a fuzzy extractor $(\mathrm{KG}, \mathrm{REP})$, a similarity-preserving hash function $\tilde{\mathrm{h}} : \{0,1\}^* \rightarrow \{0,1\}^{\ell}$, two hash function $\mathrm{H}_1 : \{0,1\}^* \rightarrow \{0,1\}^{\eta}$, $\mathrm{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_N$ and a pseudo-random function $\pi : \{1, \cdots, N\} \times \{0,1\}^* \rightarrow \mathbb{Z}_N$, where ℓ is the bit length of media file tag and η is the bit length of encryption key. The $(\mathrm{KG}, \mathrm{REP}, \tilde{\mathrm{h}}, \mathrm{H}_1, \mathrm{H}_2, \pi, \mathrm{e})$ are public functions.

Algorithm 1 describes the system setup process for the attribute authority.

2) Tag Generation: When sharing a media file F, a media distributor first executes **TagGen**$(F, PK) \rightarrow \{tag_F\}$ to generate a tag for fuzzy duplication detection as follows.

- For an image file F, the distributor can compute its similarity-preserving hash value $\boldsymbol{h} = \tilde{\mathrm{h}}(F)$ as the media file tag tag_F.
- For a video file F, the media distributor extracts its keyframes $\widetilde{F} = \{kf_i\}_{1 \le i \le n}$ (e.g., n keyframes), and computes the similarity-preserving hash value $\boldsymbol{h} = \tilde{\mathrm{h}}(\widetilde{F})$ as the media file tag tag_F.

3) Fuzzy Duplication Detection: After generating the tag tag_F, a media distributor and the cloud execute the fuzzy duplication detection **FDupDet**$(T_1, T_2, tag_F) \rightarrow \{(1/0, \perp)/(1, tag_{F'})\}$ as follows.

- The media distributor sets a maximum allowable Hamming distance T_1 and a reliability threshold T_2 based on the importance of the file, and sends (T_1, T_2, tag_F) to the cloud.

Algorithm 1 System Setup

Input: κ: security parameter.
U: the universe set of attributes.
Output: PK: public parameters.
msk: master secret key.
$\{SK_{at_i}\}_{at_i \in U}$: attribute secret keys.

Attribute authority:

$\mathcal{G}en(\kappa) \rightarrow (p, q, N, \mathbb{G}_1, \mathbb{G}_2, \mathrm{e})$.
Choose an integer ω.
Randomly choose a generator $g \in \mathbb{G}_1$.
Randomly choose three exponents $\alpha, \beta, \tau \in \mathbb{Z}_N$.
$msk = g^{\alpha}$, $sk_{attr}^{(1)} = g^{\alpha} g^{\tau\beta}$, $sk_{attr}^{(2)} = g^{\tau}$.
for each $at_i \in U$ **do**
  Randomly choose $g_{at_i} \in \mathbb{G}_1$.
  $sk_{at_i} = g_{at_i}^{\tau}$.
  $SK_{at_i} = \{sk_{attr}^{(1)}, sk_{attr}^{(2)}, sk_{at_i}\}$.
  Send SK_{at_i} to its corresponding media consumers.
$PK = \{N, g, \mathrm{e}(g, g)^{\alpha}, g^{\beta}, \{g_{at_i}\}_{at_i \in U}, \omega\}$.

- The cloud compares tag_F with its stored tags and identifies the media file F' whose tag $tag_{F'}$ has the smallest Hamming distance (denoted as T_{min}) with tag_F. If $T_{min} \leq T_1$ and the reliability score of the file F' exceeds the threshold T_2, the cloud regards the media distributor as a subsequent distributor and returns $(1, \perp)$. If $T_{min} \leq T_1$ and the reliability score of the file F' is lower than or equal to T_2, the cloud sends $(1, tag_{F'})$ to the media distributor. If $T_{min} > T_1$, the cloud classifies the media distributor as an initial distributor and returns $(0, \perp)$.

An initial distributor needs to upload the ciphertext C_F of the media file, the ciphertext CT_F of the encryption key, and auxiliary data PA_F for fuzzy PoW. A subsequent distributor needs to prove his/her media data ownership to the cloud. Receiving $(1, tag_{F'})$ means that the media distributor needs to further verify the reliability of the similar file F'.

4) Encryption: An initial distributor encrypts the media file and the encryption key using a standard symmetric encryption and CP-ABE, respectively. The **Encrypt**$(F, PK, \mathcal{T}) \rightarrow \{C_F, CT_F, PP_0\}$ function is described as follows.

- The initial distributor randomly selects an encryption key $ek \in \mathbb{G}_1$ and encrypts the media file using a standard symmetric encryption as $C_F = \mathrm{SE.Enc}(\mathrm{H}_1(ek), F)$.
- The initial distributor generates a trapdoor key tk_F by firstly executing $\mathrm{KG}(F/\widetilde{F}, r_0, u_0) \rightarrow \{k, PP_0\}$ and then obtaining $tk_F = \mathrm{H}_2(k)$, where $r_0, u_0 \in \{0, 1\}^{\theta}$ are randomly selected.
- The initial distributor generates an LSSS access structure $(\mathbb{M}, \rho)$ from his/her access policy tree $\mathcal{T}$ using the method in [34], where $\mathbb{M}$ is a $\gamma \times \iota$ generation matrix and ρ is a function that associates the row indices of $\mathbb{M}$ and the attributes in $\mathcal{T}$. Then the initial distributor generates a vector $\boldsymbol{v} = (tk_F, v_2, \cdots, v_{\iota})$, where $v_2, \cdots, v_{\iota}$ are chosen from $\mathbb{Z}_N$ randomly. The initial distributor computes $(\lambda_1, \lambda_2, \cdots, \lambda_{\gamma}) = \mathbb{M} \cdot \boldsymbol{v}$.

Algorithm 2 Encryption

Input: F: outsourced file.
$PK = \{N, g, \mathrm{e}(g, g)^{\alpha}, g^{\beta}, \{g_{at_i}\}_{at_i \in U}, \omega\}$: public parameters.
$\mathcal{T}$: access policy tree.
Output: C_F: ciphertext of F.
CT_F: ciphertext of encryption key.
PP_0: output parameter of fuzzy extractor.

Media distributor:

Randomly choose $ek \in \mathbb{G}_1$.
$C_F = \mathrm{SE.Enc}(\mathrm{H}_1(ek), F)$. //SE.Enc: Symmetric encryption function; H_1: Hash function.
Randomly choose $r_0, u_0 \in \{0, 1\}^{\theta}$.
Generate LSSS access structure $(\mathbb{M}, \rho)$ from $\mathcal{T}$.
$\{k, PP_0\} = \mathrm{KG}(F/\widetilde{F}, r_0, u_0)$. //KG: Key generation function of fuzzy extractor.
$tk_F = \mathrm{H}_2(k)$. //H_2: Hash function.
Randomly choose $v_2, \cdots, v_{\iota} \in \mathbb{Z}_N$.
$(\lambda_1, \lambda_2, \cdots, \lambda_{\gamma}) = \mathbb{M} \cdot (tk_F, v_2, \cdots, v_{\iota})$.
$cm_1 = ek \cdot \mathrm{e}(g, g)^{\alpha \cdot tk_F}$, $cm_2 = g^{tk_F}$.
for $i = 1 \rightarrow \gamma$ **do**
  Randomly choose $z_i \in \mathbb{Z}_N$.
  $ca_i = g^{\beta\lambda_i} \cdot g_{\rho(i)}^{-z_i}$, $cb_i = g^{z_i}$.
$CT_F = \{cm_1, cm_2, \{ca_i, cb_i\}_{1 \leq i \leq \gamma}\}$.
Send $\{C_F, CT_F, PP_0\}$ to the cloud.

- The initial distributor computes $cm_1 = ek \cdot \mathrm{e}(g, g)^{\alpha \cdot tk_F}$ and $cm_2 = g^{tk_F}$. Then for each $i \in \{1, \cdots, \gamma\}$, the initial distributor randomly chooses $z_i \in \mathbb{Z}_N$ and computes $ca_i = g^{\beta\lambda_i} \cdot g_{\rho(i)}^{-z_i}$ and $cb_i = g^{z_i}$. Finally, the initial distributor sends the PP_0, C_F and $CT_F = \{cm_1, cm_2, \{ca_i, cb_i\}_{1 \leq i \leq \gamma}\}$ to the cloud.

Algorithm 2 presents the encryption process for the initial distributor. Note that the initial distributor can also recover the encryption key from the CT_F stored on the cloud if he/she adds his/her own attributes when constructing the access policy tree $\mathcal{T}$.

5) Decryption: An authorized consumer can recover the plaintext media content from C_F and CT_F if his/her attributes $\{\rho(i)\}_{i \in I}$ (I is the index set of his/her attributes) satisfy the access policy tree $\mathcal{T}$. The detailed decryption function **Decrypt**$(C_F, CT_F, PK, \{SK_{\rho(i)}\}_{i \in I}) \rightarrow \{F\}$ is described as follows.

- The media consumer computes Eq. (4) and obtains $\mathrm{e}(g, g)^{\alpha \cdot tk_F}$.

$$\mathrm{e}(g, g)^{\alpha \cdot tk_F} = \frac{\mathrm{e}(cm_2, sk_{attr}^{(1)})}{\prod_{i \in I} (\mathrm{e}(ca_i, sk_{attr}^{(2)}) \cdot \mathrm{e}(cb_i, sk_{\rho(i)}))^{w_i}} \tag{4}$$

- The media consumer computes Eq. (5) and obtains the encryption key ek.

$$ek = \frac{cm_1}{\mathrm{e}(g, g)^{\alpha \cdot tk_F}} \tag{5}$$

- The media consumer decrypts C_F and obtains the plaintext media file F using the decryption process of the

Algorithm 3 Auxiliary Data Generation

Input: F: outsourced file.
$PK = \{N, g, \mathrm{e}(g,g)^{\alpha}, g^{\beta}, \{g_{at_i}\}_{at_i \in U}, \omega\}$: public parameters.
Output: PA_F: Auxiliary data.

Media distributor:
Randomly choose $r_1, u_1 \in \{0,1\}^{\theta}$.
$\{seed, PP_1\} = \mathrm{KG}(F/\widetilde{F}, r_1, u_1)$. //KG: Key generation function of fuzzy extractor.
for $i = 1 \rightarrow \omega$ **do**
  $a_i = \pi(i, seed)$. //π: Pseudo-random function.
  Compute g^{a_i}.
$PA_F = \{PP_1, \{g^{a_i}\}_{0 \le i \le \omega}\}$.
Send PA_F to the cloud.

Algorithm 4 Fuzzy Ownership Proof

Input: F: outsourced file.
$PK = \{N, g, \mathrm{e}(g,g)^{\alpha}, g^{\beta}, \{g_{at_i}\}_{at_i \in U}, \omega\}$: public parameters.
$PA_{F'} = \{PP_1, \{g^{a_i}\}_{0 \le i \le \omega}\}$: Auxiliary data.
Output: $\{1/0\}$: ownership verification result.

Cloud server:
Randomly choose $\xi_1, \xi_2 \in \mathbb{Z}_N$.
Compute y_1 and y_2 using Eq. (7).
Send $\{PP_1, \xi_1, \xi_2\}$ to the media distributor.
Media distributor:
$\{seed'\} = \mathrm{REP}(F/\widetilde{F}, PP_1)$. //REP: Reproduce function of fuzzy extractor.
$\{a'_i = \pi(i, seed')\}_{0 \le i \le \omega}$. //$\pi$: Pseudo-random function.
Generate a polynomial $\varphi'(x)$ using Eq. (8).
$proof = g^{\varphi'(\xi_1)\cdot\varphi'(\xi_1)}$.
Send $proof$ to the cloud.
Cloud server:
if $\mathrm{e}(proof, g) \stackrel{?}{=} \mathrm{e}(y_1, y_2)$ **then**
  Return 1.
else
  Return 0.

used standard symmetric encryption algorithm, namely $F = \mathrm{SE.Dec}(\mathrm{H}_1(ek), C_F)$

6) Reliability Verification: If the cloud stores a similar file and its reliability score is lower than or equal to the threshold T_2 set by the media distributor, the media distributor needs to verify the ciphertext $C_{F'}$ of the similar file. The process of **RelVerify**$(F, PP_0, C_{F'}, CT_{F'}) \rightarrow \{dis\}$ is presented as follows.

- The cloud retrieves cm_1 from $CT_{F'}$, and sends the parameter PP_0, the ciphertext $C_{F'}$ corresponding to F', and cm_1 to the media distributor.
- The media distributor obtains tk_F by executing $\mathrm{REP}(F/\widetilde{F}, PP_0) \rightarrow \{k\}$ and $tk_F = \mathrm{H}_2(k)$. Then the media distributor obtains the encryption key ek using Eq. (5), and decrypts $C_{F'}$ as $F' = \mathrm{SE.Dec}(\mathrm{H}_1(ek), C_{F'})$.
- The media distributor computes the Hamming distance $dis = \mathrm{Dis}_{\mathrm{Ham}}(\tilde{\mathrm{h}}(F), \tilde{\mathrm{h}}(F'))$, and sends it to the cloud.

If $dis \le T_1$, the cloud increases the reliability score of F' by 1, and checks the data ownership of the media distributor using **FPoW.Proof** algorithm. Otherwise, the cloud treats the media file to be uploaded as a new file.

7) Auxiliary Data Generation: SimLESS requires the initial distributor to generate and upload auxiliary data for the cloud to verify the ownership of subsequent distributors. The process of **FPoW.AuxGen**$(F, PK) \rightarrow \{PA_F\}$ is described as follows.

- The initial distributor generates two random strings $r_1, u_1 \in \{0,1\}^{\theta}$, and generates a seed $seed$ by executing $\mathrm{KG}(F/\widetilde{F}, r_1, u_1) \rightarrow \{seed, PP_1\}$.
- The initial distributor generates a polynomial $\varphi(x)$ as Eq. (6), where $\{a_i = \pi(i, seed)\}_{0 \le i \le \omega}$.

$$\varphi(x) = a_0 + a_1 x + \cdots + a_{\omega} x^{\omega} \tag{6}$$

- The initial distributor computes $\{g^{a_i}\}_{0 \le i \le \omega}$ and sends the PoW auxiliary data $PA_F = \{PP_1, \{g^{a_i}\}_{0 \le i \le \omega}\}$ to the cloud.

Algorithm 3 shows the auxiliary data generation process for the initial distributor.

8) Fuzzy Ownership Proof: A subsequent distributor needs to prove the ownership of media data to the cloud as Algorithm 4. The process of **FPoW.Proof**$(F, PK, PA_{F'}) \rightarrow \{1/0\}$ is described as follows.

- The cloud chooses $\xi_1, \xi_2 \in \mathbb{Z}_N$ randomly, and computes y_1 and y_2 using the following equation.

$$\begin{cases} y_1 = \prod_{i=0}^{\omega} (g^{a_i})^{\xi_1^i} \\ y_2 = \prod_{i=0}^{\omega} (g^{a_i})^{\xi_2^i} \end{cases} \tag{7}$$

  It then sends $\{PP_1, \xi_1, \xi_2\}$ to the subsequent distributor for challenging.
- The subsequent distributor generates a seed $seed'$ by executing $\mathrm{REP}(F/\widetilde{F}, PP_1) \rightarrow \{seed'\}$. It then generates a polynomial $\varphi'(x)$ as,

$$\varphi'(x) = a'_0 + a'_1 x + \cdots + a'_{\omega} x^{\omega} \tag{8}$$

  where $\{a'_i = \pi(i, seed')\}_{0 \le i \le \omega}$.
- The subsequent distributor computes the proof $proof = g^{\varphi'(\xi_1)\cdot\varphi'(\xi_2)}$ and sends it back to the cloud.
- The cloud checks the proof by verifying

$$\mathrm{e}(proof, g) \stackrel{?}{=} \mathrm{e}(y_1, y_2). \tag{9}$$

If the verification is valid, the distributor also generates some auxiliary data for his/her authorized consumers to recover the encryption key by performing the **Dec.AuxGen** algorithm with the cloud.

- The cloud sends PP_0 to the subsequent distributor.
- The subsequent distributor generates an LSSS access structure $(\mathbb{M}, \rho)$ using his/her access policy tree $\mathcal{T}$. He/she then obtains the trapdoor key tk_F by executing $\mathrm{REP}(F/\widetilde{F}, PP_0) \rightarrow \{k\}$ and $tk_F = \mathrm{H}_2(k)$.
- The subsequent distributor generates a vector $\boldsymbol{v} = (tk_F, v_2, \cdots, v_{\iota})$ where $v_2, \cdots, v_{\iota}$ are randomly selected, and computes $(\lambda_1, \lambda_2, \cdots, \lambda_{\gamma}) = \mathbb{M} \cdot \boldsymbol{v}$. For each

$i \in \{1, \cdots, \gamma\}$, he/she randomly chooses $z_i \in \mathbb{Z}_N$ and computes $ca_i = g^{\beta\lambda_i} \cdot g_{\rho(i)}^{-z_i}$ and $cb_i = g^{z_i}$. Then he/she sends the auxiliary data $KA_F = \{ca_i, cb_i\}_{1\le i\le\gamma}$ of key recovery to the cloud for storage.

The authorized consumers of the subsequent distributor can recover the encryption key from $\{cm_1, cm_2\}$ using KA_F. Note that the subsequent distributor can also recover the encryption key from $\{cm_1, cm_2\}$ if he/she adds his/her own attributes when constructing the access policy tree $\mathcal{T}$.

V. Scheme Analysis

A. Correctness

We prove the correctness of the fuzzy ownership proof and media recovery in our scheme. The fuzzy duplication detection is implemented using similarity-preserving hash function, whose correctness has been proved in [23].

1) Correctness of Fuzzy Ownership Proof: Suppose that the initial distributor of F has uploaded $PA_F = \{PP_1, \{g^{a_i}\}_{0\le i\le\omega}\}$ to the cloud. When the subsequent distributor with a similar F' receives the challenging information $\{PP_1, \xi_1, \xi_2\}$ from the cloud, he/she computes the ownership proof $proof$ and sends it back to the cloud. Then the cloud verifies the proof using Eq. (9), the two sides of the equation are shown as,

$$\begin{aligned}\mathrm{e}(proof, g) &= \mathrm{e}(g^{\varphi'(\xi_1)\cdot\varphi'(\xi_1)}, g)\\ &= \mathrm{e}\left(g^{\sum_{i=o}^{\omega}\pi(i,seed')\cdot\xi_1^i}, g^{\sum_{i=o}^{\omega}\pi(i,seed')\cdot\xi_2^i}\right)\\ \mathrm{e}(y_1, y_2) &= \mathrm{e}\left(g^{\sum_{i=o}^{\omega}\pi(i,seed)\cdot\xi_1^i}, g^{\sum_{i=o}^{\omega}\pi(i,seed)\cdot\xi_2^i}\right)\end{aligned}$$

where $\{seed, PP_1\} = \mathrm{KG}(F, r_1, u_1)$ and $\{seed'\} = \mathrm{REP}(F', PP_1)$. According to Definition 4, when F and F' are similar, the extracted $seed' = seed$. Then $\mathrm{e}(proof, g) = \mathrm{e}(y_1, y_2)$, which verifies the correctness of the fuzzy PoW method. Besides, we prove that the fuzzy PoW method can protect the security of media data ownership in the following Section V-B.2.

2) Correctness of Media Recovery: Suppose that the initial distributor has uploaded the ciphertext C_F of a media file and the ciphertext CT_F of encryption key under the LSSS access structure $(\mathbb{M}, \rho)$ to the cloud. An authorized consumer whose attributes $\{\rho(i)\}_{i\in I}$ satisfying the access policy can recover the encryption key as

$$ek = \frac{cm_1}{\mathrm{e}(g,g)^{\alpha\cdot tk_F}} = \frac{ek\cdot\mathrm{e}(g,g)^{\alpha\cdot tk_F}}{\mathrm{e}(g,g)^{\alpha\cdot tk_F}}$$

where

$$\begin{aligned}\mathrm{e}(g,g)^{\alpha\cdot tk_F} &= \frac{\mathrm{e}(cm_2, sk_{attr}^{(1)})}{\prod_{i\in I}(\mathrm{e}(ca_i, sk_{attr}^{(2)})\cdot\mathrm{e}(cb_i, sk_{\rho(i)}))^{w_i}}\\ &= \frac{\mathrm{e}(g,g)^{\alpha\cdot tk_F}\mathrm{e}(g,g)^{\tau\cdot\beta\cdot tk_F}}{\prod_{i\in I}\mathrm{e}(g,g)^{\tau\cdot\beta\cdot\lambda_i\cdot w_i}}.\end{aligned}$$

After recovering the encryption key ek, the authorized consumer can recover the plaintext media file F by decrypting C_F as $F = \mathrm{SE.Dec}(\mathrm{H}_1(ek), C_F)$.

For an authorized consumer of a valid subsequent distributor, if his/her attributes $\{\rho'(i)\}_{i\in I'}$ satisfy the access structure $(\mathbb{M}', \rho')$ of the subsequent distributor, he/she can recover the encryption key from $\{cm_1, cm_2\}$ and $KA_F = \{ca_i', cb_i'\}_{1\le i\le\gamma'}$ as

$$ek = \frac{cm_1}{\mathrm{e}(g,g)^{\alpha\cdot tk_F}} = \frac{ek\cdot\mathrm{e}(g,g)^{\alpha\cdot tk_F}}{\mathrm{e}(g,g)^{\alpha\cdot tk_F}}$$

where

$$\begin{aligned}\mathrm{e}(g,g)^{\alpha\cdot tk_F} &= \frac{\mathrm{e}(cm_2, sk_{attr}^{(1)})}{\prod_{i\in I'}(\mathrm{e}(ca_i', sk_{attr}^{(2)})\cdot\mathrm{e}(cb_i', sk_{\rho'(i)}))^{w_i'}}\\ &= \frac{\mathrm{e}(g,g)^{\alpha\cdot tk_F}\mathrm{e}(g,g)^{\tau\cdot\beta\cdot tk_F}}{\prod_{i\in I'}\mathrm{e}(g,g)^{\tau\cdot\beta\cdot\lambda_i'\cdot w_i'}}.\end{aligned}$$

Since the authorized consumer of the subsequent distributor can get the encryption key ek, he/she can decrypt C_F correctly. As a result, the authorized consumers of the initial distributor and subsequent distributors can correctly recover the plaintext media content.

B. Security

1) Security of Media Content: According to the threat model in Section III-B, both the cloud and unauthorized consumers attempt to obtain the private media content.

We prove that even a collusion between the cloud and unauthorized consumers cannot obtain the private media content if the BDH assumption holds. Let $\mathcal{C}$ be a challenger, $\mathcal{A}$ be an adversary that can obtain the private media content with advantage $\epsilon(\kappa)$.

- Setup: According to Definition 7, $\mathcal{A}$ can get the public parameters $PK = \{N, g, \mathrm{e}(g,g), g^{\beta}, \{g_{at_i}\}_{at_i\in U}\}$ and attribute secret keys $\{SK_{at}\}_{at\in A_i}$, $(1 \le i \le \ell)$ corresponding to its chosen sets of attributes $\{A_i\}_{1\le i\le\ell}$.
- Challenge: The adversary chooses an access policy tree $\mathcal{T}$ such that none of the sets $\{A_i\}_{1\le i\le\ell}$ satisfying the access policy. The challenger $\mathcal{C}$ flips a fair binary coin $\eta \in \{0, 1\}$ and chooses two media files m_0 and m_1. The challenger $\mathcal{C}$ returns the ciphertexts C_{m_η} and CT_{m_η} under the access policy $\mathcal{T}$ and the file tag tag_{m_η} to $\mathcal{A}$.
- Guess: $\mathcal{A}$ outputs a guess $\hat{\eta}$ of η. If $\eta = \hat{\eta}$, $\mathcal{C}$ outputs 1. Otherwise, $\mathcal{C}$ outputs 0.

Since $\mathcal{C}$ outputs 1 only when the output η equals to $\hat{\eta}$, we have

$$\Pr[\mathcal{A}(\eta = \hat{\eta})] = \frac{1}{2} + \epsilon(\kappa)$$

The advantage $\epsilon(\kappa)$ of $\mathcal{A}$ is from ciphertexts C_{m_η}, CT_{m_η}, file tag tag_{m_η}, public parameters PK, and attribute secret keys $\{SK_{at}\}_{at\in A_i}$, $(1 \le i \le \ell)$. Since the encryption key ek of standard symmetric encryption is randomly selected, and the two files m_0, m_1 are unpredictable messages for the adversary, neither file tag tag_{m_η} nor the ciphertext C_{m_η} can be used by $\mathcal{A}$ to test data equality (similarity), providing no advantage to $\mathcal{A}$.

Below we prove that the adversary $\mathcal{A}$ cannot obtain any advantage from CT_{m_η}, PK, and $\{SK_{at}\}_{at\in A_i}$, $(1 \le i \le \ell)$ if the BDH assumption holds.

Since the encryption key ek is encrypted under the access policy $\mathcal{T}$ using CP-ABE and none of the sets of attributes

$\{A_i\}_{1\le i\le \ell}$ satisfies the access policy, $\mathcal{A}$ cannot obtain ek from CT_{m_η} through standard decryption process, which is guaranteed by the security of CP-ABE. Suppose that the adversary $\mathcal{A}$ can obtain encryption key ek from CT_{m_η}, PK, and $\{SK_{at}\}_{at\in A_i}$, $(1 \le i \le \ell)$ through other processes, we can construct a simulator that can solve the BDH problem using the ek.

Given g^τ, g^β, and g^{tk_F}, the simulator is required to output $\mathrm{e}(g, g)^{\tau\cdot\beta\cdot tk_F}$. The simulator can solve the BDH problem as follows.

If the adversary $\mathcal{A}$ can obtain correct encryption key ek, we have

$$\mathrm{e}(g^\alpha, g^{tk_F}) = \frac{cm_1}{ek}, \tag{10}$$

where $cm_1 = ek \cdot \mathrm{e}(g, g)^{\alpha\cdot tk_F}$ is a part of CT_{m_η}. Using the Eq. (10), we can obtain

$$\begin{aligned}\mathrm{e}(sk^{(1)}_{attr}, cm_2) &= \mathrm{e}(g^\alpha, g^{tk_F}) \cdot \mathrm{e}(g^{\tau\cdot\beta}, g^{tk_F})\\ &= \frac{cm_1}{ek} \cdot \mathrm{e}(g, g)^{\tau\cdot\beta\cdot tk_F},\end{aligned}$$

where $sk^{(1)}_{attr} = g^\alpha \cdot g^{\tau\cdot\beta}$ is a part of attribute secret key, and $cm_2 = g^{tk_F}$ is a part of CT_{m_η}. Then it is clear that

$$\mathrm{e}(g, g)^{\tau\cdot\beta\cdot tk_F} = \mathrm{e}(sk^{(1)}_{attr}, cm_2) \cdot \frac{ek}{cm_1}.$$

This indicates that we have discovered a solution to the BDH problem. However, according to Definition 1, there is no probability polynomial-time algorithm to solve the BDH problem. Consequently, the advantage $\epsilon(\kappa)$ of $\mathcal{A}$ can be ignored in polynomial time and we have

$$\mathrm{Adv}^{\mathrm{PRV-CDA}}_{\mathcal{A}}(\kappa) = 2 \cdot \Pr[\mathcal{A}(\widehat{\eta} = \eta)] - 1 = 0.$$

As a result, our SimLESS satisfies the security Definition 7 and achieves the PRV-CDA security of media content.

2) Security of Media Data Ownership: According to the Definition 8, the adversary $\mathcal{A}$ can break the ownership security, only when the two conditions $k \neq \mathrm{REP}(F', PP)$ and $1 \leftarrow \mathrm{FPoW.Rroof}(F', PK, PA_F)$ hold simultaneously.

Since an identical message can be extracted from two similar files through the fuzzy extractor, the first condition $k \neq \mathrm{REP}(F', PP)$ holds for $(k, PP) \leftarrow \mathrm{KG}(F, r, u)$, where $r \leftarrow \{0,1\}^\theta$ and $u \leftarrow \{0,1\}^\theta$ are the random input parameters of fuzzy extractor, indicating that the adversary does not have a file similar to the claimed file F. The second condition $1 \leftarrow \mathrm{FPoW.Rroof}(F', PK, PA_F)$ holds, meaning that the adversary should use the non-similar file F' to pass the ownership verification of the **FPoW.Rroof** algorithm, where PK is the public parameters and $PA_F \leftarrow \mathrm{FPoW.AuxGen}(F, PK)$ is the PoW auxiliary data generated using file F.

Since F' is not similar to F, it is infeasible for $\mathcal{A}$ to generate correct $proof = g^{\varphi'(\xi_1)\cdot\varphi'(\xi_2)}$ using F' by honestly following the procedure of the **FPoW.Proof** algorithm. Because the security of fuzzy extractor can prevent the adversary from generating the correct polynomial $\varphi'(x)$ using non-similar file F', thereby the adversary cannot obtain the correct values $\varphi'(\xi_1)$ and $\varphi'(\xi_2)$.

In this case, suppose that the adversary can generate the correct $proof$, we can discover a solution to the CDH problem using the $proof$.

Given g, $y_1 = g^{\varphi'(\xi_1)}$, $y_2 = g^{\varphi'(\xi_2)}$, the CDH problem requires the output of $g^{\varphi'(\xi_1)\cdot\varphi'(\xi_2)}$ for unknown $\varphi'(\xi_1)$ and $\varphi'(\xi_2)$. If the $proof$ generated by the adversary is correct, we have

$$\begin{aligned}\mathrm{e}(proof, g) &= \mathrm{e}(y_1, y_2)\\ &= \mathrm{e}(g^{\varphi'(\xi_1)\cdot\varphi'(\xi_2)}, g).\end{aligned} \tag{11}$$

This indicates that we have discovered a solution $proof = g^{\varphi'(\xi_1)\cdot\varphi'(\xi_2)}$ to the CDH problem.

However, according to Definition 2, there is no probability polynomial-time algorithm to solve the CDH problem. Consequently, we have

$$\mathrm{Adv}^{\mathrm{OS}}_{\mathcal{A}}(\kappa) = \Pr[\mathrm{Game}^{\mathrm{OS}}_{\mathcal{A}}(\kappa) = 1] = 0.$$

As a result, our SimLESS satisfies the security Definition 8 and achieves ownership security.

3) Resistance to DFA: According to the Definition 9, the adversary $\mathcal{A}$ can break the tag consistency, only when the two conditions $T' > T$ and $T \ge \mathbf{RelVerify}(F, PP_0, C_{F'}, CT_{F'})$ hold simultaneously, indicating

$$\begin{cases}\mathrm{Dis}_{\mathrm{Ham}}(\tilde{\mathrm{h}}(F), \tilde{\mathrm{h}}(F')) > \mathrm{Dis}_{\mathrm{Ham}}(tag_{F'}, tag_F)\\ \mathrm{Dis}_{\mathrm{Ham}}(tag_{F'}, tag_F) \ge \mathrm{Dis}_{\mathrm{Ham}}(\tilde{\mathrm{h}}(F), \tilde{\mathrm{h}}(\widehat{F'})),\end{cases} \tag{12}$$

where $\widehat{F'}$ is the recovered plaintext using the **RelVerify** algorithm. Obviously, when the adversary $\mathcal{A}$ holds the consistency between $CT_{F'}, C_{F'}, PP_0$ with F', the **RelVerify** algorithm can obtain correct $\widehat{F'} = F'$, failing to make the Eq. (12) hold. In this case, we have $\Pr[\mathrm{Game}^{\mathrm{TC}}_{\mathcal{A}}(\kappa) = 1] = 0$.

When the adversary $\mathcal{A}$ breaks the consistency between $CT_{F'}, C_{F'}, PP_0$ with F', the **RelVerify** algorithm can only obtain incorrect $\widehat{F'} \neq F'$, potentially making the Eq. (12) hold. However, if $\mathrm{Dis}_{\mathrm{Ham}}(tag_{F'}, tag_F) \ge \mathrm{Dis}_{\mathrm{Ham}}(\tilde{\mathrm{h}}(F), \tilde{\mathrm{h}}(\widehat{F'}))$ holds, subsequent media distributors would obtain $\widehat{F'}$, and its similarity to F is higher than the similarity indicated by tag_F and $tag_{F'}$, which contradicts the goal of DFA.

As a result, the SimLESS satisfies the security Definition 9 and achieves tag consistency under DFA.

C. Functionality Analysis

Our SimLESS allows the cloud to perform deduplication over encrypted similar media data, as well as protects the confidentiality of the media data. It allows a media distributor to set a distance threshold between his/her media file with the file stored on the cloud, which provides the distributor with a great feasibility to determine the similar deduplication degree according to the importance of the file. SimLESS also achieves fine-grained access control for media distributors, which can ensure that only the authorized media consumers can access the private media content. Besides, we delicately design a fuzzy PoW protocol to protect media data ownership security, which can prevent any media distributor claiming the ownership of a media file using only the tag of a similar file.

While some secure deduplication schemes have been developed to date, they are tailored for classical cloud storage

TABLE II
FUNCTIONALITY AND SECURITY COMPARISONS

Scheme	Media content security	Access control	Media data ownership security	Similar media deduplication
SMACD [11]	√	√	×	×
SimLESS	√	√	√	√

model, where the data uploader and downloader are the same entity. However, these schemes cannot be directly applied to the cloud media sharing model, in which the data uploader (i.e., media distributor) and the downloader (i.e., media consumer) are different entities. Since our SimLESS is specifically designed for the cloud media sharing model, the secure deduplication schemes tailored for the classical cloud storage model do not meet the requirements of our work. The recently proposed SMACD [11] is the only prior scheme that addresses secure deduplication in the cloud media sharing model, aligning with the focus of our work. Thus, we compare our scheme with the only similar work SMACD [11] from the functionality and security and list the results in Table II.

SMACD achieves secure deduplication over identical media data and fine-grained access control for media distributors. Due to the high information redundancy in media data, similar media data can present the same visual effect. Thus, deduplicating similar media is crucial for enhancing cloud storage efficiency in cloud media sharing. However, SMACD cannot support similar media data deduplication. Compared to deduplication over identical data, it is more difficult to achieve deduplication over encrypted similar data as slightly different plaintexts will be encrypted to completely different ciphertexts. By eliminating similar media data, our SimLESS not only benefits storage efficiency, but also reduces communication cost for users since they no longer need to upload similar media files. Besides, our scheme can ensure media data ownership security, which is not considered in SMACD.

D. Discussion

Note that a deduplication scheme may produce false positives, where files are detected as duplicates even though they are not the same. This can be caused by hash collisions. Previous secure deduplication schemes did not consider this issue, as it is considered to be a low probability event.

Our solution does not explicitly address the problem of false positives, where files detected as near-duplicate may actually not be similar. However, we can address this problem from two aspects. Firstly, if a media distributor does not delete the local file after sharing, he/she can re-upload the shared media file and mark it as a new file if a false positive is encountered. Secondly, if a distributor intends to delete the local file after sharing, we require the initial distributor of a media file to encrypt additional information (e.g., the thumbnail of an image and the keyframes of a video) using the same key as the shared file. The distributor can then restore the encryption key and decrypt the additional information to determine whether his/her file is similar to the file stored on the cloud.

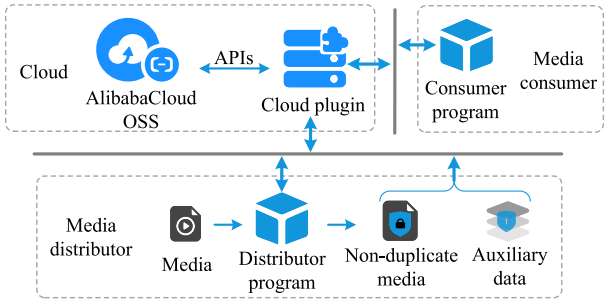


Fig. 5. Implementation of a prototype of SimLESS using AlibabaCloud OSS.

VI. IMPLEMENTATION AND EVALUATION

A. Experimental Settings

We implement a prototype of SimLESS using C++ programming language. Our implementation relies on the OpenSSL Library [36], OpenCV Library [37], Pairing Based Cryptography (PBC) Library [38], and GNU Multiple Precision Arithmetic (GMP) Library [39]. We use Alibaba cloud object storage service (AlibabaCloud OSS) [7] as the cloud backend storage. Note that the proposed SimLESS can also be applied to other CSPs such as Dropbox [6] and Google Drive [5]. These CSPs provide similar APIs with Alibaba cloud, allowing us to deploy our SimLESS using the same flow as on Alibaba cloud.

Fig. 5 shows the prototype structure. The cloud server consists of the cloud plugin and AlibabaCloud OSS. The cloud plugin is developed to interact with other entities, manage the data index database implemented using MySQL [40], and call the APIs of AlibabaCloud OSS for storing and retrieving ciphertexts. We use a windows laptop with 4 cores Intel Core i7-7560U processor and 8GB of memory to perform the cloud plugin program. The programs of media distributor and consumer are deployed in a same MacOS laptop with 4 cores Intel Core i5-1038NG7 processor and 16GB of memory. Besides, the attribute authority program is deployed in a windows laptop with 4 cores Intel Core i5-6200 processor and 8GB of memory, and it is only executed once in the setup process, which is not included in Fig. 5. All the laptops are deployed in a same wireless LAN and their maximum communication bandwidth is limited to 20Mbps.

We set the bit length κ of the secure parameter as 256, the bit length η of encryption key as 256, and the bit length ℓ of tag as 256. We use the standard AES-256 to encrypt the media files. We evaluate the performance of SimLESS using the Columbia University Object Image Library (COIL) [41] and Image of LEGO Bricks dataset [42] that contain many similar images. Besides, we only use the gray feature in the similarity-preserving hash function and fixed five attributes in the access policy tree to simplify the efficiency analysis. We simulate five media distributors and evenly assign each dataset among them, allocating one-fifth of the dataset to each media distributor. We set the reliability threshold as 2 in the experiments.

B. Overall Time Cost

Since the cloud server usually has a large computation ability and the attribute authority can be offline after executing

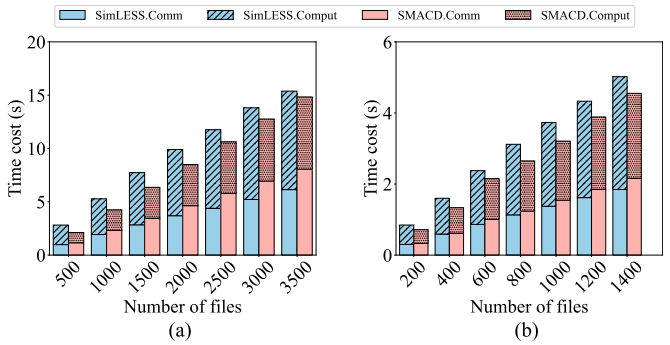


Fig. 6. Actual computation and communication time cost of media distributor. (a) The average time cost in LEGO Bricks dataset; (b) the average time cost in COIL dataset.

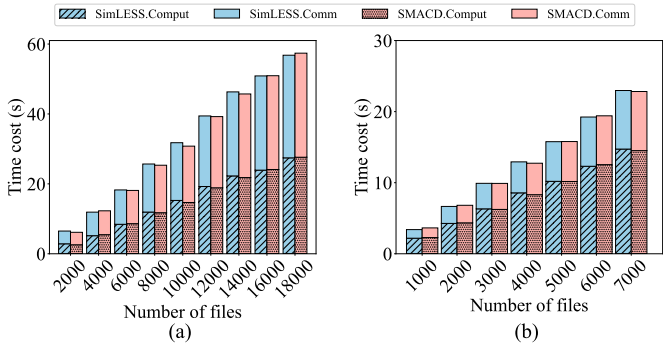


Fig. 7. Actual computation and communication time cost of media consumer. (a) The average time cost in LEGO Bricks dataset; (b) the average time cost in COIL dataset.

the setup program once, we evaluate the time costs of the media distributor and media consumer. We set the distance threshold T_1 as 8 in the overall time cost experiments.

We first measure the computation and communication time cost of the media distributor, and show the results in Fig. 6. As can be seen, the computation and communication cost grows with the increase of the number of the uploaded files. The results show that our SimLESS increases a slightly higher total time cost than SMACD. This is due to the fact that SimLESS has the features of DFA resistance, data ownership security, and fuzzy duplication. Operations such as similarity-preserving hash computation, fuzzy extraction operations, and encryption key sharing contribute to the additional computational overhead to the media distributor. Despite this, SimLESS demonstrates a lower communication time cost in comparison to SMACD. This is because SimLESS supports fuzzy deduplication over similar images, resulting in the reduced necessity to upload many similar images.

We then test the computation and communication cost of the media consumer, and Fig. 7 shows the results. The computation and communication cost grows with the increase of the number of the downloaded files. The computation and communication time cost of the media consumer in SimLESS is very similar to that of the media consumer in SMACD, because both of them require the media consumer to recover the encryption keys and decrypt the media data ciphertexts.

C. Storage Cost

We analyze the cloud storage overhead from the perspectives of deduplication ratio and storage cost. As can be seen

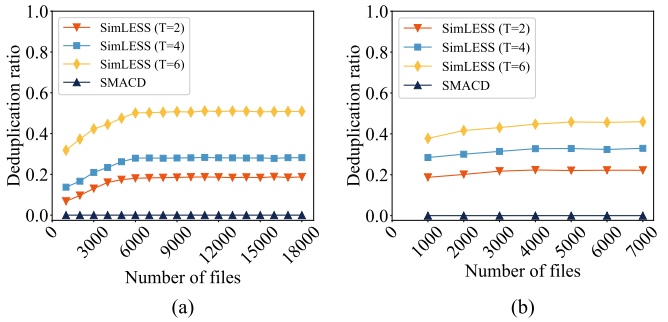


Fig. 8. Comparisons of cloud deduplication ratio. (a) The deduplication ratio in LEGO Bricks dataset; (b) the deduplication ratio in COIL dataset.

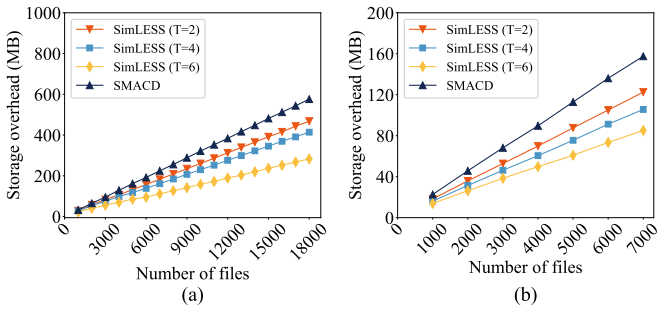


Fig. 9. Comparisons of cloud storage overhead of media data ciphertexts. (a) The cloud storage cost in LEGO Bricks dataset; (b) the cloud storage cost in COIL dataset.

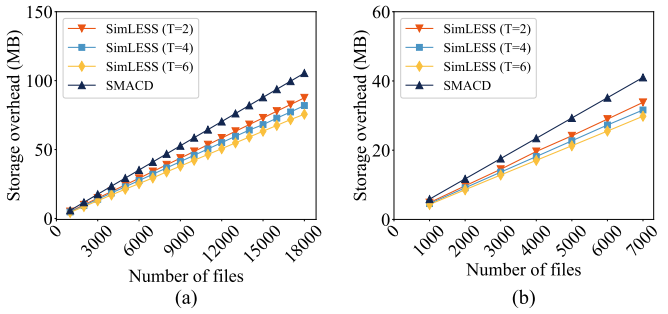


Fig. 10. Comparisons of cloud storage overhead of encryption key ciphertexts. (a) The cloud storage cost in LEGO Bricks dataset; (b) the cloud storage cost in COIL dataset.

in Fig. 8, SimLESS achieves obviously higher deduplication ratios than SMACD in both two datasets, because SimLESS supports similar data deduplication that is not considered in SMACD. Thus, as can be seen in Fig. 9, SimLESS has a significantly lower cloud storage overhead for media data than SMACD.

Besides, we also measure the cloud storage overhead for storing the ciphertexts of encryption keys. As shown in Fig. 10, our SimLESS has a lower cloud storage overhead. This is because for a duplicate image in SimLESS, a subsequent distributor only uploads some auxiliary data that are used by his/her authorized consumers to decrypt the ciphertext of the encryption key uploaded by the initial distributor. As a contrast, a subsequent distributor in SMACD should upload and store a new ciphertext of encryption key.

D. Visual Effect of Fuzzy Deduplication

A distributor can set a distance threshold between his/her media file with the file stored on the cloud, which provides

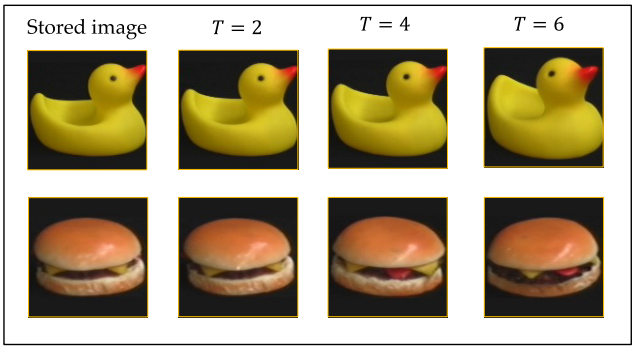


Fig. 11. Duplicate samples within different Hamming distances in COIL dataset.

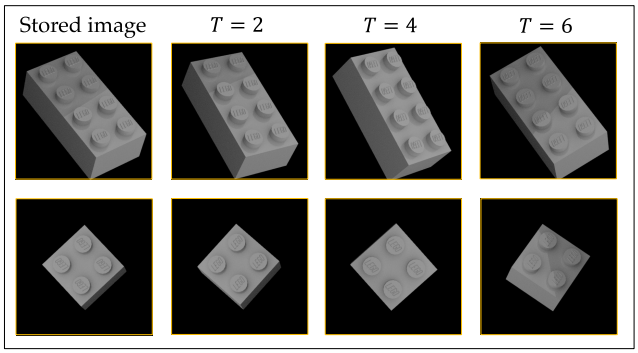


Fig. 12. Duplicate samples within different Hamming distances in LEGO Bricks dataset.

the distributor with a great feasibility to determine the similar deduplication degree according to the importance of the file. To show that fuzzy deduplication cannot affect the image visual effect, we show some image samples with different distances. For an image stored on the cloud, we find its similar images within different distances. As shown in Figs. 11 and 12, a smaller distance will result in a higher matching similar image. For example, the similar image with $T = 2$ is almost visually indistinguishable from the original image. Though there exist some differences between the original image and the similar image with $T = 6$, they are acceptable in visual. Moreover, a larger distance threshold causes a higher deduplication ratio and storage efficiency.

VII. Related Work

Secure deduplication technology can improve cloud storage efficiency while also protecting data privacy. It has been extensively studied on classical cloud storage. Convergent encryption (CE) [43] was the first secure deduplication solution, in which a user derives the encryption key from the plaintext itself. This way, identical plaintexts will be deterministically encrypted to the same ciphertexts by different users. Bellare et al. [15] formalized this encryption strategy as MLE. However, the deterministic MLE is vulnerable to brute-force attacks [44]. To enhance the security of MLE, server-aided MLE has been developed by using either a single key server [44] or a group of key servers [45], [46] to participate in key generation. Dave et al. [47] proposed a pseudorandom key-based encryption mechanism, which can improve data security against brute-force attacks. Nowadays, most existing secure deduplication schemes [48], [49], [50] are designed for client-side deduplication, since this kind of deduplication can achieve a lower communication bandwidth overhead than server-side deduplication. However, client-side deduplication schemes usually suffer from some security threats, such as hash-only attack [35]. Halevi et al. [25] pointed out that an adversary owing only the tag of a file can convince the cloud that he/she owns the file. To tackle this threat, the authors developed a Merkle tree-based PoW scheme to verify that the user indeed owns the claimed file. Later, some PoW schemes were developed for improving the efficiency [24], [51] and security [52], [53].

All the aforementioned secure deduplication schemes are designed for encrypted identical data and cannot work over encrypted similar data. Compared to identical data deduplication, to achieve deduplication over encrypted similar data is more complex as it is difficult to encrypt two similar files to the same or similar ciphertexts. Recently, some schemes [21], [22], [23] were developed to perform secure deduplication over similar data. Li et al. [21] proposed a secure fuzzy deduplication scheme for group users with the same group keys. However, it cannot work across different groups of users. Takeshita et al. [22] proposed another fuzzy deduplication scheme based on PAKE. An initial uploader should stay online and securely share his/her encryption key to subsequent uploaders using PAKE, which is impractical. Jiang et al. [23] proposed a secure fuzzy deduplication scheme FuzzyDedup using oblivious transfer. FuzzyDedup is more practical as it does not require online user to assist the uploading of subsequent uploaders. However, users are required to encrypt their files using only XOR operation such that two similar plaintexts can be encrypted to two similar ciphertexts for deduplication. Since a same key is used to encrypt all the blocks of a file, the XOR operation cannot provide high-security protection for the file content.

Recently, Huang et al. [11] proposed the first deduplication scheme SMACD for encrypted identical media data in cloud media sharing. They designed a key sharing mechanism between media distributors and media consumers using CP-ABE, which achieves fine-grained access control to media consumers. However, SMACD still focuses on the deduplication of encrypted identical media data. Our SimLESS achieves deduplication over encrypted similar media data. It is also designed to allow the cloud to verify the ownership of subsequent distributors and thus can protect media data ownership security, which is not considered in SMACD.

VIII. Conclusion

In this paper, we propose SimLESS as a deduplication system for encrypted similar media data in cloud media sharing. SimLESS allows the cloud server to deduplicate encrypted similar media uploaded by different media distributors while also protecting media data privacy against the cloud server. Media distributors can achieve fine-grained access control to their media consumers, ensuring that only authorized consumers can access the private media content. SimLESS enables a media distributor to determine the threshold of distance

between his/her file and the file stored on the cloud. The cloud performs deduplication only when the distance between the file and the stored file is smaller than the threshold. Additionally, SimLESS achieves fuzzy PoW to prevent a media distributor from claiming the ownership of a media file using only the tag of a similar file, ensuring media data ownership security. We theoretically prove the security of SimLESS and implement a prototype using AlibabaCloud OSS as backend cloud storage. The comprehensive evaluation results demonstrate the efficiency of our design.

REFERENCES

[1] Y. Meng, C. Jiang, T. Q. S. Quek, Z. Han, and Y. Ren, "Social learning based inference for crowdsensing in mobile social networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1966–1979, Aug. 2018.

[2] T. Taleb, A. Ksentini, M. Chen, and R. Jantti, "Coping with emerging mobile social media applications through dynamic service function chaining," *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, pp. 2859–2871, Apr. 2016.

[3] K. Yang, J. Shu, and R. Xie, "Efficient and provably secure data selective sharing and acquisition in cloud-based systems," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 71–84, 2023.

[4] Z. Zhu and R. Jiang, "A secure anti-collusion data sharing scheme for dynamic groups in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 40–50, Jan. 2016.

[5] Google. (2012). *Google Drive*. [Online]. Available: https://cloud.google.com/storage?hl=zh-cn#media-content-storage-and-delivery

[6] Dropbox. (2018). *Dropbox*. [Online]. Available: https://www.dropbox.com/dropbox

[7] Alibaba. (2022). *Alibaba Cloud*. [Online]. Available: https://help.aliyun.com/product/31815.html

[8] M. Fröbe et al., "CopyCat: Near-duplicates within and between the ClueWeb and the common crawl," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 2398–2404.

[9] W. Yang and Y. Zhu, "A verifiable semantic searching scheme by optimal matching over encrypted data in public cloud," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 100–115, 2021.

[10] J. Shen, P. Zeng, K. R. Choo, and C. Li, "A certificateless provable data possession scheme for cloud-based EHRs," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1156–1168, 2023.

[11] Q. Huang, Z. Zhang, and Y. Yang, "Privacy-preserving media sharing with scalable access control and secure deduplication in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 1951–1964, May 2021.

[12] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.

[13] S. Zhang, S. Ray, R. Lu, Y. Guan, Y. Zheng, and J. Shao, "Toward privacy-preserving aggregate reverse skyline query with strong security," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2538–2552, 2022.

[14] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," *IEEE Netw.*, vol. 28, no. 4, pp. 46–50, Jul. 2014.

[15] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Proc. 32nd Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*, 2013, pp. 296–312.

[16] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, and W. Lou, "Secure and efficient cloud data deduplication with randomized tag," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 3, pp. 532–543, Mar. 2017.

[17] Z. Yan, L. Zhang, W. DING, and Q. Zheng, "Heterogeneous data storage management with deduplication in cloud computing," *IEEE Trans. Big Data*, vol. 5, no. 3, pp. 393–407, Sep. 2019.

[18] Y. Fu, N. Xiao, T. Chen, and J. Wang, "Fog-to-multicloud cooperative eHealth data management with application-aware secure deduplication," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 5, pp. 3136–3148, Oct. 2022.

[19] M. Song, Z. Hua, Y. Zheng, H. Huang, and X. Jia, "Blockchain-based deduplication and integrity auditing over encrypted cloud storage," *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 6, pp. 4928–4945, Nov. 2023, doi: 10.1109/TDSC.2023.3237221.

[20] Y. Shin, D. Koo, J. Yun, and J. Hur, "Decentralized server-aided encryption for secure deduplication in cloud storage," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 1021–1033, Nov. 2020.

[21] X. Li, J. Li, and F. Huang, "A secure cloud storage system supporting privacy-preserving fuzzy deduplication," *Soft Comput.*, vol. 20, no. 4, pp. 1437–1448, Apr. 2016.

[22] J. Takeshita, R. Karl, and T. Jung, "Secure single-server nearly-identical image deduplication," in *Proc. 29th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Honolulu, HI, USA, Aug. 2020, pp. 1–6.

[23] T. Jiang et al., "FuzzyDedup: Secure fuzzy deduplication for cloud storage," *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 3, pp. 2466–2483, May 2023, doi: 10.1109/TDSC.2022.3185313.

[24] J. Dave, A. Dutta, P. Faruki, V. Laxmi, and M. S. Gaur, "Secure proof of ownership using Merkle tree for deduplicated storage," *Autom. Control Comput. Sci.*, vol. 54, no. 4, pp. 358–370, Jul. 2020.

[25] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, Oct. 2011, pp. 491–500.

[26] K. Huang, X. Zhang, Y. Mu, F. Rezaeibagha, and X. Du, "Bidirectional and malleable proof-of-ownership for large file in cloud storage," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2351–2365, Oct. 2022.

[27] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Oct. 2006, pp. 89–98.

[28] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.

[29] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, Jan. 2008.

[30] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptogr. Conf.*, 2007, pp. 535–554.

[31] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. Annu. Int. Cryptol. Conf.*, 2001, pp. 213–229.

[32] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, Sep. 2004.

[33] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. 14th Int. Conf. Pract. Theory Public Key Cryptogr.*, 2011, pp. 53–70.

[34] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. 30th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2011, pp. 568–588.

[35] V. Rabotka and M. Mannan, "An evaluation of recent secure deduplication proposals," *J. Inf. Security Appl.*, vol. 27, pp. 3–18, Apr. 2016.

[36] OpenSSL. (2015). *Open Source Socket Layer*. [Online]. Available: https://www.openssl.org/

[37] OpenCV. (2012). *Open Source Computer Vision Library*. [Online]. Available: https://opencv.rog/

[38] B. Lynn. (2010). *The Pairing-based Cryptography (PBC) Library*. [Online]. Available: https://crypto.stanford.edu/pbc/

[39] T. Granlund. (2010). *GNU Multiple Precision Arithmetic Library*. [Online]. Available: http://gmplib.org/

[40] Oracle. (2008). *MySQL*. [Online]. Available: https://dev.mysql.com/doc/refman/5.6/en/

[41] S. Nene, S. Nayar, and H. Murase. (1988). *Columbia Object Image Library (CIOL-100)*. [Online]. Available: https://www.kaggle.com/datasets/jessicali9530/coil100

[42] Graviti Open Datasets. (2018). *Image of LEGO Bricks*. [Online]. Available: https://www.kaggle.com/datasets/joosthazelzet/lego-brick-images

[43] J. R. Douceur, A. Adya, W. J. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *Proc. 22nd Int. Conf. Distrib. Comput. Syst.*, 2002, pp. 617–624.

[44] S. Keelveedhi, M. Bellare, and T. Ristenpart, "DupLESS: Server-aided encryption for deduplicated storage," in *Proc. USENIX Secur.*, 2013, pp. 179–194.

[45] Y. Duan, "Distributed key generation for encrypted deduplication: Achieving the strongest privacy," in *Proc. 6th, Ed., ACM Workshop Cloud Comput. Secur.*, Nov. 2014, pp. 57–68.

[46] Y. Zhang, C. Xu, N. Cheng, and X. Shen, "Secure password-protected encryption key for deduplicated cloud storage systems," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 4, pp. 2789–2806, Jul. 2022.

[47] J. Dave, P. Faruki, V. Laxmi, A. Zemmari, M. Gaur, and M. Conti, "SPARK: Secure pseudorandom key-based encryption for deduplicated storage," *Comput. Commun.*, vol. 154, pp. 148–159, Mar. 2020.

[48] G. Tian et al., "Blockchain-based secure deduplication and shared auditing in decentralized storage," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 6, pp. 3941–3954, Nov. 2022.
[49] X. Liu, W. Sun, W. Lou, Q. Pei, and Y. Zhang, "One-tag checker: Message-locked integrity auditing on encrypted cloud deduplication storage," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Atlanta, GA, USA, May 2017, pp. 1–9.
[50] S. Li, C. Xu, Y. Zhang, Y. Du, and K. Chen, "Blockchain-based transparent integrity auditing and encrypted deduplication for cloud storage," *IEEE Trans. Services Comput.*, vol. 16, no. 1, pp. 134–146, Jan. 2023.
[51] J. Blasco, R. Di Pietro, A. Orfila, and A. Sorniotti, "A tunable proof of ownership scheme for deduplication using Bloom filters," in *Proc. IEEE Conf. Commun. Netw. Secur.*, San Francisco, CA, USA, Oct. 2014, pp. 481–489.
[52] R. Di Pietro and A. Sorniotti, "Boosting efficiency and security in proof of ownership for deduplication," in *Proc. 7th ACM Symp. Inf., Comput. Commun. Secur.*, May 2012, pp. 81–82.
[53] D. Pancholi, J. A. Y. Dave, and M. Bhatt, "Secure proof of ownership for deduplicated cloud storage system," *Int. J. Inf. Comput. Secur.*, vol. 21, nos. 1–2, pp. 205–228, 2023.

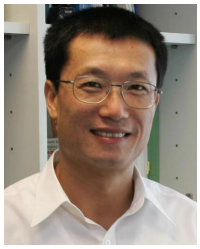
Mingyang Song received the B.E. and M.E. degrees in software engineering from Sun Yat-sen University, Guangzhou, China, in 2019 and 2021, respectively. He is currently pursuing the Eng.D. degree with the Department of Electronic Information, Harbin Institute of Technology, Shenzhen. His research interests include security and privacy related to cloud computing, applied cryptography, and blockchain.

Zhongyun Hua (Senior Member, IEEE) received the B.S. degree in software engineering from Chongqing University, Chongqing, China, in 2011, and the M.S. and Ph.D. degrees in software engineering from the University of Macau, Macau, China, in 2013 and 2016, respectively.

He is currently an Associate Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His works have appeared in prestigious venues, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, CVPR, AAAI, and ACM MM. He has published about 90 articles on the subject, receiving more than 6500 citations. His current research interests include chaotic systems, multimedia security, and secure cloud computing. He has been recognized as the "Highly Cited Researcher 2022" and "Highly Cited Researcher 2023."

Yifeng Zheng (Member, IEEE) received the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2019. He was a Post-Doctoral Researcher with the Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia, and City University of Hong Kong. He is currently an Assistant Professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His work has appeared in prestigious venues, such as ESORICS, DSN, ACM AsiaCCS, IEEE INFOCOM, IEEE ICDCS, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and IEEE TRANSACTIONS ON SERVICES COMPUTING. His current research interests include security and privacy related to cloud computing, the IoT, machine learning, and multimedia. He received the Best Paper Award in the European Symposium on Research in Computer Security (ESORICS) 2021.

Tao Xiang (Senior Member, IEEE) received the B.Eng., M.S., and Ph.D. degrees in computer science from Chongqing University, China, in 2003, 2005, and 2008, respectively. He is currently a Professor with the College of Computer Science, Chongqing University. He has published over 150 papers on international journals and conferences. His research interests include multimedia security, cloud security, data privacy, and cryptography. He served as a referee for numerous international journals and conferences.



Xiaohua Jia (Fellow, IEEE) received the B.Sc. and M.Eng. degrees from the University of Science and Technology of China in 1984 and 1987, respectively, and the D.Sc. degree in information science from The University of Tokyo, in 1991.

He is currently the Chair Professor with the Department of Computer Science, City University of Hong Kong. He is an Adjunct Professor with Harbin Institute of Technology, Shenzhen, while performing this work. His research interests include cloud computing and distributed systems, computer networks, and mobile wireless networks. He is a fellow of the IEEE Computer Society. He is the General Chair of ACM MobiHoc 2008, the Area-Chair of IEEE INFOCOM 2010, the TPC Co-Chair of IEEE GlobeCom 2010-Ad Hoc and Sensor Networking Symposium, and the Panel Co-Chair of IEEE INFOCOM 2011. He is an editor of the *World Wide Web Journal* and IEEE TRANSACTIONS ON COMPUTERS.