FCDedup: A Two-Level Deduplication System for Encrypted Data in Fog Computing

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

***Abstract*—Distributed fog computing has received increasing attention recently and fog-assisted cloud storage can provide a real-time service to collect and manage large-scale data for the applications of Internet of Things. Encrypted data deduplication over cloud storage can significantly save storage space of the cloud server while protecting the confidentiality of the outsourced data. Previous encrypted data deduplication schemes are mostly designed for traditional cloud storage with a two-layer architecture and cannot be applied to the emerging fog-assisted cloud storage that has a more complex three-layer architecture (i.e., cloud server, fog node and endpoint device). In this paper, we design, analyze and implement FCDedup, a new encrypted data deduplication scheme for fog-assisted cloud storage. FCDedup is a two-level deduplication system that enables each fog node to detect duplicated encrypted data uploaded by different endpoint devices, as well as enables cloud server to detect duplicated encrypted data from different fog nodes. By doing so, FCDedup can achieve both intra-deduplication within a single data owner and inter-deduplication across different data owners. FCDedup is also designed to prevent cloud server and fog nodes launching the brute-force attacks, and to guarantee the reliability of files downloaded from the cloud. Formal analysis is provided to justify its deduplication correctness and security. Besides, we implement a prototype of FCDedup using Alibaba Cloud as backend storage. Our evaluations demonstrate that FCDedup is completely compatible with existing cloud storage systems and achieves modest performance overhead.**

***Index Terms*—Brute-force attacks, data reliability, encrypted data deduplication, fog-assisted cloud storage.**

Manuscript received 3 February 2023; revised 2 June 2023; accepted 18 July 2023. Date of publication 25 July 2023; date of current version 4 August 2023. This work was supported in part by the National Key R&D 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 Grants 2021A1515110027, 2021A1515011406 and 2023A1515010714, in part by Shenzhen Science and Technology Program under Grants RCBS20210609103056041 and JCYJ20220531095416037, in part by the Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies under Grant 2022B1212010005, and in part by Shenzhen Science and Technology Program under Grant ZDSYS20210623091809029. Recommended for acceptance by V. Cardellini. *(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 the Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Shenzhen 518055, China (e-mail: huazhongyun@hit.edu.cn).

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, Kowloon Tong, Hong Kong, and also with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Guangdong 518055, China (e-mail: csjia@cityu.edu.hk).

Digital Object Identifier 10.1109/TPDS.2023.3298684

I. Introduction

In THE Big Data era, large volumes of digital data are exponentially produced. To save storage costs, data owners are more and more willing to outsource their data to cloud servers for storage. However, many data stored on the cloud are duplicated [1]. For example, the data duplication rates on standard file systems are more than 50% [2] and on patient health records can reach 75% [3], [4]. Thus, data deduplication can help cloud service providers (CSP) to greatly reduce the storage costs. Besides, the cloud storage users are concerned about the confidentiality of their outsourced data, since these data may contain many private information. Thus, they are willing to encrypt their outsourced data before uploading them to cloud server. As a result, data deduplication over encrypted domain is significant for cloud server to reduce the storage costs and for data owners to protect the data confidentiality.

Traditional two-layer cloud storage may be inefficient to handle enormous volumes of data in a timely manner [5], [6] and thus cannot provide a real-time service for some large-scale data collection applications of Internet of Things (IoT) [7]. For example, in a health monitoring system, the wearable medical sensors generate medical data that are outsourced to the cloud servers for storage and the authorized hospitals, doctors or patients can download these data. Because the wearable medical sensors with limited performance generate a large amount of medical data continuously, it will cause heavy computation and bandwidth costs to these sensors if directly uploading these data to the cloud server [8], [9]. Fog computing is a distributed computing infrastructure located between the cloud and endpoint devices, and it has the advantages of low latency, location awareness, and etc [10], [11], [12]. With these attractive features, three-layer fog-assisted cloud storage can provide a real-time service for collecting large-scale data [13], [14], [15]. In a fog-assisted cloud storage system, endpoint devices (e.g., IoT devices) are deployed at the edge of the network to collect data and outsource the collected data to a cloud server for storage via fog nodes. Since the fog nodes close to the endpoint devices share the data processing and management costs for the endpoint devices during data storage [13], some fog-assisted cloud storage systems have been used to collect and manage large-scale data in a timely manner [16].

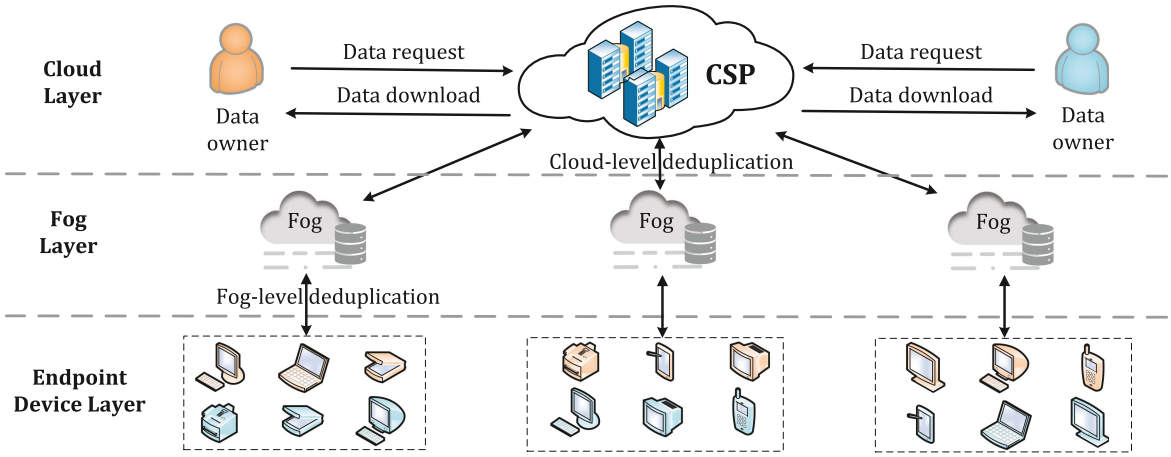


Fig. 1. System model of our proposed FCDedup.

Recently, many encrypted data deduplication schemes have been developed for traditional cloud storage [17], [18], [19], [20], [21], [22], [23], [24], [25] and these schemes can be roughly divided into directly compatible schemes [17], [18], [19], [20] and key server-assisted schemes [21], [22], [23], [24], [25] based on whether additional key server is required or not. The first category encrypts the outsourced data using the message-locked encryption (MLE) [17], where the encryption key is derived from the plaintext data itself and different users can encrypt an identical data block into the same ciphertext for deduplication. However, the deterministic MLE inherently suffers from the brute-force attacks (BFA) [21]. To defense BFA, the second category introduces additional key server(s) to participate in generating encryption key, and thus can well protect the outsourced data. Existing encrypted data deduplicaiton schemes are mostly designed for traditional cloud storage with a two-layer architecture. However, fog-assisted cloud storage contains three layers (i.e., cloud server, fog node and endpoint device) and its architecture is more complex than traditional cloud storage. Thus, it requires more ingenious design to perform encrypted data deduplicaiton on fog-assisted cloud storage than on traditional cloud storage. It is also almost impossible to deploy trusted key servers in fog computing since an endpoint device only communicates with the near fog node in practice. Therefore, these previous schemes are inapplicable to fog-assisted cloud storage that has a different system architecture from traditional cloud storage.

Besides, the data reliability verification should be specifically designed in fog-assisted cloud storage, which is not considered in previous encrypted data deduplication schemes. Since data loss may happen on the cloud due to some objective reasons such as hardware failure [26], a data owner should have the ability to verify the consistency between his/her downloaded files with the original files uploaded to the cloud. In traditional cloud storage, a data owner can easily verify the data reliability, because he/she is the data uploader and holds the correct file fingerprints for verification. However, in fog-assisted cloud storage, the files are uploaded by endpoint devices and it is difficult for a data owner to obtain the correct data fingerprints from his/her endpoint devices due to some practical reasons (e.g., the offline or failure of some devices). Thus, verifying data reliability in the fog-assisted cloud storage becomes challenging for the data owner.

To address the aforementioned problems in fog-assisted cloud storage, we propose FCDedup, a new encrypted data deduplication scheme over fog-assisted cloud storage. We design a two-level duplication detection mechanism that enables each fog node to detect duplicated encrypted data uploaded by the endpoint devices of the same data owner, and also enables the cloud server to detect duplicated encrypted data from different fog nodes belonging to the same or different data owners. By doing so, FCDedup can achieve both intra-deduplication within a single data owner and inter-deduplication across different data owners. We develop a random encryption key sharing mechanism that can allow all the data owners of an identical data block to recover the random encryption key used by an endpoint device while providing the ciphertext with a high ability to resist BFA. Besides, we design a data verification mechanism to verify the number and content of the downloaded files compared to the files uploaded by the endpoint devices.

The contributions and novelty of this paper are summarized as follows.

- We propose FCDedup, the first encrypted data deduplication scheme to achieve both intra-deduplication and inter-deduplication on fog-assisted cloud storage. FCDedup can perform secure deduplication at both fog nodes and cloud server, and thus can greatly improve storage efficiency and reduce communication overhead.
- We provide a data verification mechanism that allows a data owner to verify the file content and file number between the files downloaded from the cloud with the files uploaded by the endpoint devices, which can guarantee the reliability of files downloaded from the cloud.
- We formally prove the correctness of FCDedup and its security guarantees. FCDedup is designed to be easily compatible with existing commercial cloud storage systems, and we implement and evaluate a prototype of FCDedup using Alibaba Cloud [27] as backend storage to demonstrate such advantage.

TABLE I
IMPORTANT NOTATIONS

Notation	Description
O_d	The d^{th} data owner.
F_b	The b^{th} fog node.
$D_{d,a}$	The a^{th} endpoint device of data owner O_d.
PK_C, SK_C	The public key and secret key of CSP.
PK_{F_b}, SK_{F_b}	The public key and secret key of F_b.
PK_{O_d}, SK_{O_d}	The public key and secret key of O_d.
$SK_{D_{d,a}}$	The secret key of $D_{d,a}$.
$R_{d,a}$	The registration ticket of $D_{d,a}$.
p, q	Two large primes.
$\mathbb{Z}_N$	Residue class ring.
$\mathbb{G}_1, \mathbb{G}_2$	Two multiplicative cyclic groups.
λ	The bit length of an encryption key.
κ	The security parameter of bilinear pairing.
ι	The bit length of a short hash value.
$H_1(\cdot)$, $H_2(\cdot)$	Two collision-resistant hash functions.
$H_3(\cdot)$	A normal hash function.
$H_4(\cdot)$	A short hash function.
$e(\cdot,\cdot)$	Bilinear pairing e: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
$\lvert\xi\rvert$	The bit length of arbitrary element ξ.

The remainder of this paper is organized as follows. Section II describes the system model, threat model, and design goals and challenges. Section III presents the details of our FCDedup. Section IV provides the correctness and security proofs of FCDedup. Section V implements a prototype of FCDedup in a commercial cloud platform and evaluates its performance. Section VI reviews some related works. Finally, we conclude our study in Section VII.

II. PROBLEM STATEMENT

A. System Model

As shown in Fig. 1, our system contains four types of entities: data owner, endpoint device, fog node, and CSP.

- *Data owner:* A data owner is an institution or individual that deploys many endpoint devices to collect data. The data owner employs the CSP to store these collected data.
- *Endpoint device:* The endpoint devices are deployed by data owners to collect data. The collected data are encrypted and then uploaded to the CSP via fog nodes. Different endpoint devices may collect the same data.
- *Fog node:* The fog nodes are some public network infrastructures located at the edge of the network. They receive encrypted data from the endpoint devices within their coverage areas and forward the received data to the CSP. A fog node detects duplicated encrypted data uploaded by the endpoint devices of the same data owner (i.e., fog-level deduplication).
- *CSP:* The CSP provides storage service for data owners. It receives encrypted data from different fog nodes and detects duplicated encrypted data belonging to the same or different data owners (i.e., cloud-level deduplication).

B. Threat Model

The threats in our system are from the aspects of the fog nodes and CSP, and we describe them as follows.

- *Fog node:* The fog nodes can honestly execute the predefined protocols but are curious about the private content of their received encrypted data. Specifically, the fog nodes may launch BFA to guess the content of the received ciphertext using the background knowledge of the plaintext space.
- *CSP:* The CSP can honestly execute the predefined protocols but is also curious about the private content of its received encrypted data. It may guess the private content of the received ciphertext by launching BFA. Besides, the encrypted data stored on the cloud may be lost or damaged due to some objective reasons such as hardware failure. To maintain its commercial reputation, it may deceive the data owners (e.g., send the data owners forged data or claim that the endpoint devices did not upload the data) when the data are lost or damaged.

C. Design Goals and Challenges

1) Design Goals: Our FCDedup aims to perform encrypted data deduplication over fog-assisted cloud storage and has the following design goals.

- *Functionality:* The scheme supports both *fog-level deduplication* and *cloud-level deduplication*. To achieve this, we design a two-level duplication detection mechanism, which enables each fog node to detect duplicated encrypted data uploaded by the endpoint devices of the same data owner and also enables the cloud server to detect duplicated encrypted data from different fog nodes belonging to the same or different data owners. As a result, our scheme can achieve both *intra-deduplication* and *inter-deduplication*.
- *Security:* The security goals of our scheme include *data confidentiality* and *data reliability*. (1) The data confidentiality indicates that neither the CSP nor fog nodes can obtain the private content of their received encrypted data. (2) The data reliability contains file content reliability and file number reliability, indicating that a data owner can verify the file content and file number between the downloaded files with the files uploaded by the endpoint devices.

2) Design Challenges: To achieve the aforementioned design goals, the following questions should be considered and addressed by special designs.

$\mathcal{Q}1$: *How to achieve data deduplication at the fog nodes and cloud server while ensuring a high data confidentiality?* To reduce the communication and storage costs as much as possible in fog-assisted cloud storage, data deduplication should be performed at both the fog nodes and cloud server. Since the data tags used for duplication detection contain some deterministic information of the data (e.g., the deterministic hash value of plaintext data), the CSP and fog nodes may use the data tags to launch BFA. Thus, we should design a secure two-level duplication detection mechanism, in which the used data tags have the ability to resist BFA.

$\mathcal{Q}2$: *How to securely share an encryption key among an endpoint device and the valid data owners?* To improve data security against BFA, previous secure deduplication schemes either introduce additional key servers to participate in generating the encryption key or use random encryption keys [28]. However, it is almost impossible to deploy trusted key servers in fog computing since an endpoint device (e.g., biomedical sensors and environment monitoring sensors) only communicates with the near fog node in practice. Thus, our scheme uses the random encryption keys generated by the endpoint devices to resist BFA. However, this causes a new challenging problem that is securely sharing an encryption key among an endpoint device and the valid data owners. Therefore, we should design an encryption key generation and sharing mechanism, which allows the valid data owners of an identical file to securely recover the encryption key.

$\mathcal{Q}3$: *How to verify the file number and file content between the downloaded files with the files uploaded by the endpoint devices?* In fog-assisted cloud storage, the files are encrypted and uploaded by the endpoint devices and the data owner does not hold file fingerprints. It is also difficult for the data owners to obtain the data fingerprints from their endpoint devices due to some practical reasons (e.g., the offline or failure of some endpoint devices). Thus, a special mechanism should be designed to allow a data owner to check the file content and file number to guarantee the data reliability.

III. The Design of FCDedup

A. Notations and Preliminaries

Suppose that there are δ data owners $\{O_d\}_{(1\leq d\leq \delta)}$ and β fog nodes $\{F_b\}_{(1\leq b\leq \beta)}$, and a data owner O_d deploys α endpoint devices $\{D_{d,a}\}_{(1\leq a\leq \alpha)}$. Each endpoint device is registered under a fog node according to its location. Table I lists the important notations used in this paper.

1) Bilinear Pairing of Composite Order: Given a security parameter κ, run the composite bilinear parameter generator $\mathcal{G}en(\kappa)$ and get $(p, q, N, \mathbb{G}_1, \mathbb{G}_2, e)$, where p, q are κ-bit length primes, $N = p \cdot q$ and $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear pairing [29] with the following three properties:

- *Bilinearity:* $\forall x, y \in \mathbb{G}_1$ and $\forall a, b \in \mathbb{Z}_N$, $e(x^a, y^b) = e(x, y)^{ab}$.
- *Non-degenerate:* If g is a generator of $\mathbb{G}_1$, then $e(g, g)$ is a generator of $\mathbb{G}_2$.
- *Computability:* $\forall x, y \in \mathbb{G}_1$, there exists an efficient algorithm for computing $e(x, y) \in \mathbb{G}_2$.

Definition 1 (The Computation Diffie-Hellman (CDH) assumption [30]): Given g, g^x and $g_1 \in \mathbb{G}_1$, for unknown $x \in \mathbb{Z}_N$, there is no probabilistic polynomial-time algorithm that can compute g_1^x with non-negligible advantage.

Definition 2 (The Divisible Computation Diffie-Hellman (DCDH) assumption [31]): Given g, g^x and $g^y \in \mathbb{G}_1$, there is no probabilistic polynomial-time algorithm that can compute $g^{y/x}$ with non-negligible advantage.

2) Hash Collision: A hash function can map a binary stream of arbitrary length to a deterministic binary stream with a specified length. A hash function with long outputs has a stronger ability to resist hash collision, while with short outputs may have many collisions. Due to this property, a hash function with long outputs is usually used to identify data, while with short outputs is usually used to resist BFA [32].

B. Design Overview

Here, we present how to address the challenges described in Section II-C2.

1) Two-Level Duplication Detection: A fog node may receive duplicated encrypted data from different endpoint devices and the cloud server may also receive duplicated encrypted data from different fog nodes. These duplicated encrypted data may belong to the same data owner or different data owners. Thus, deduplication at the fog nodes and cloud server can reduce the communication and storage costs. Our system aims to perform intra-deduplication, but does not perform inter-deduplication at the fog nodes. This is because designing a secure cross-owner duplication detection mechanism at the fog nodes may cause heavy additional computation burden to the endpoint devices. For practical considerations, the cross-owner duplicated encrypted data at each fog node are detected at the cloud server. Besides, the cloud server further detects the duplicated encrypted data from different fog nodes. As a result, our system aims to perform intra-deduplication at the fog nodes and perform both intra-deduplication and inter-deduplication at the cloud server.

The intra-deduplication at a fog node can be achieved by designing a special fog-level data tag. Specifically, the fog-level data tag is designed to involve both a data-related deterministic value (e.g., the long hash value) and a secret value chosen by a data owner in the setup process. When some endpoint devices belonging to the same data owner upload the duplicated encrypted data to a fog node, the fog node obtains the exactly same fog-level data tags and thus can detect duplication. Since the fog-level data tag contains a secret value chosen by the data owner, it can efficiently prevent the fog node launching BFA.

To achieve both the intra-deduplication and inter-deduplication at the cloud server, the cloud-level data tag is designed to contain the signing key of the first fog node that uploads the encrypted data, an random value and a data-related deterministic value. The signing key is used to prevent the cloud server launching BFA, the random value is chosen by the endpoint device to prevent the fog node launching BFA, while the data-related deterministic value is computed by the endpoint device and it is used by the CSP to identify the duplicated encrypted data.

When uploading a new data block, the endpoint device generates a short hash value and a base value that contains a random value and the data-related deterministic value. The fog node uses the short hash value to find the potential fog nodes that may have uploaded the same data block by interacting with the cloud server. A joint public key is computed between every two fog nodes in the system initialization phase. Thus, when finding some potential fog nodes, the fog node can compute a tag for each potential fog node that contains the signing key of the potential fog node and the base value. After receiving these data tags, the cloud server compares each received data

tag and its stored related cloud-level data tag. The duplication is detected if their data-related deterministic values are the same. If no duplication is detected, a cloud-level data tag is generated for the current data block and stored on the cloud server.

2) Encryption Key Sharing: Our system should ensure that either the cloud server or the fog node cannot derive the encryption key to prevent them launching BFA. To achieve this effect, we design a random encryption key that consists of two random values and the hash value of the plaintext. The first random value is stored on the cloud server and the second one is stored on the fog node during the first data uploading. After two-level duplication detection, the endpoint device can know whether the same data block has been stored or not. When the data has not been uploaded, yet, the endpoint device sends the first random value to the cloud server via the fog node and the second one to the fog node using a public key cryptosystem. Besides, the endpoint device encrypts the hash value of the plaintext using the public key of the data owner and stores the encrypted hash value on the cloud server via the fog node. The fog node encrypts its stored random value using the public key of the data owner and also stores the encrypted random value on the cloud server.

If data duplication has been detected, the subsequent endpoint device encrypts the hash value of the plaintext using the public key of the current data owner and stores the encrypted hash value on the cloud. The system requires that the fog node first uploading the data block encrypts its stored random value using the public key of the current data owner and then sends the generated encrypted random value to the cloud server.

As a result, for each ciphertext, the cloud server stores the following information for each valid owner: the first random value, the encrypted second random value and encrypted hash value that are encrypted using the public key of the data owner. When a data owner retrieves the data, the cloud server sends these information to the data owner along with the ciphertext. Note that the first random value is sent through public key cryptography. Then the data owner can use his/her secret key to recover the two random values and the hash value of the plaintext. Thus, the encryption key can be recovered.

3) Data Reliability Verifying: The data owner should have the ability to verify the file number and file content between the files downloaded from the cloud server with the files uploaded by his/her endpoint devices. A straightforward way is to check whether the file fingerprints are matched or not. However, since the files stored on the cloud server are encrypted and uploaded by the endpoint devices, it is difficult for a data owner to obtain file fingerprints from its endpoint devices due to some practical reasons (e.g., the offline or failure of some devices).

To achieve this ability, our system requires an endpoint device to generate verification auxiliary data for each collected file using its secret key shared by the data owner in the setup process, and store it on the cloud server during the data uploading. The hash value of the plaintext is included in the auxiliary data in encrypted form and it is used to verify the file content. Since the cloud cannot obtain the secret key of the endpoint device, the CSP cannot falsify content verification auxiliary data to deceive the data owner when the encrypted data is lost. In the data retrieval process, our scheme also requires the related fog node to generate verification auxiliary data for the data owner to verify the file number. The number of files uploaded by each endpoint device is included in the verification auxiliary data in signature form, which cannot be falsified by the CSP. Then file number verification can be achieved.

C. System Initialization

1) Setup of the Cloud and Fog Nodes: The CSP initializes some public parameters and its secret key using its security parameter κ as follows.

- Run the composite bilinear parameter generator $\mathcal{G}en(\kappa)$ and obtain $(p, q, N, \mathbb{G}_1, \mathbb{G}_2, e)$.
- Select a generator $g \in \mathbb{G}_1$, compute the public key $PK_C = g^q$, and set the secret key $SK_C = p$.
- Select four hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_N$, $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_3 : \mathbb{G}_1 \rightarrow \{0,1\}^{\lambda}$, and $H_4 : \{0,1\}^* \rightarrow \{0,1\}^{\iota}$, where λ is the bit length of the encryption key and ι is the bit length of short hash value.

The $(PK_C, N, \mathbb{G}_1, \mathbb{G}_2, g, e, H_1, H_2, H_3, H_4)$ are the public parameters of the system. The secret key SK_C and the prime q are kept by the CSP.

A fog node F_b randomly chooses a secret key $SK_{F_b} \in \mathbb{Z}_N^*$ and computes its public key $PK_{F_b} = g^{SK_{F_b}^{-1}}$, which is sent to all the entities. After obtaining the public keys $\{PK_{F_{b'}}\}_{(1 \le b' \le \beta, b \ne b')}$ of the other $\beta - 1$ fog nodes, the fog node F_b computes the joint public keys $\{UK_{b',b} = PK_{F_{b'}}^{SK_{F_b}}\}_{(1 \le b' \le \beta, b \ne b')}$ and publishes $\{UK_{b',b}\}_{(1 \le b' \le \beta, b \ne b')}$. The secret key and joint public keys are used to generate cloud-level data tags.

2) Setup of Data Owners: A data owner O_d deploys some endpoint devices $\{D_{d,a}\}_{(1 \le a \le \alpha)}$ in the system as follows.

- Randomly choose a secret key $SK_{O_d} \in \mathbb{Z}_N^*$ and compute the public key $PK_{O_d} = g^{SK_{O_d}}$.
- Randomly choose a secret value $SV_{O_d} \in \mathbb{Z}_N^*$.
- For each endpoint device $D_{d,a}$, the data owner randomly chooses a secret key $SK_{D_{d,a}} \in \mathbb{Z}_N^*$ for it and computes a registration ticket $R_{d,a} = g^{SV_{O_d} - SK_{D_{d,a}}}$.

The secret key $SK_{D_{d,a}}$ is kept by the endpoint device $D_{d,a}$ and the data owner. When the endpoint device is registered with a fog node F_b, the data owner sends the registration ticket $R_{d,a}$ to the fog node F_b.

To better understand the proposed FCDedup, we provide an example for the storage structure on the cloud and fog nodes in Fig. 2. For the cloud, there are two index tables and metadata files for storing data. The encrypted data blocks are stored independently, and they are not included in Fig. 2. The index structure $DB^{(C,1)}$ is used to index data blocks stored on the cloud. For each data block, the CSP stores its following information in sequence.

- *Short hash:* multiple data blocks may have the same short hash value.
- *Fog node:* the fog node that first uploads the block.
- *Data tag:* the cloud-level data tag of the data block.
- *Block index:* the unique identifier of the block stored on the cloud.

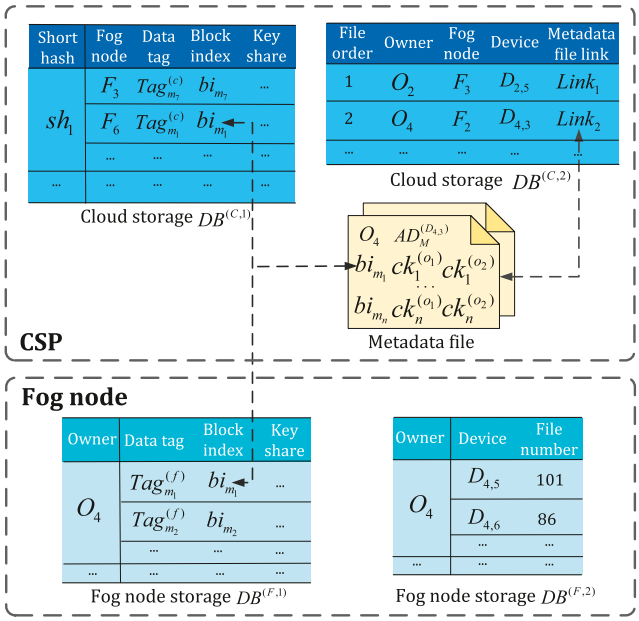


Fig. 2. Storage structure of the CSP and fog nodes.

- *Key share:* a partial share of the encryption key that encrypts the data block.

The index structure $DB^{(C,2)}$ is used to record the file information. For a file including multiple data blocks, the CSP stores its following information in sequence.

- *File order:* the unique file order.
- *Owner:* the owner of the file.
- *Fog node:* the fog node that uploads the file.
- *Device:* the endpoint device that uploads the file.
- *Metadata file link:* each file has a metadata file that records the verification auxiliary data generated by the endpoint device, the unique indices of the file's data blocks, and the encrypted key shares of the file's data blocks.

For a fog node, the index structure $DB^{(F,1)}$ is used to record data blocks. For each data block, the fog node stores its following information.

- *Owner:* the owner of the data block.
- *Data tag:* the fog-level data tag of the data block.
- *Block index:* the unique identifier of the data block stored on the cloud.
- *Key share:* a partial share of the encryption key for the data block.

Besides, the fog node also records the number of files uploaded by every endpoint device under its coverage area in the index structure $DB^{(F,2)}$.

D. Two-Level Duplication Detection Mechanism

We design a two-level duplication detection mechanism, which allows an endpoint device to determine whether the data to be uploaded already exists on the cloud or not. The duplication detection can be achieved through two phases (i.e., fog-level duplication detection and cloud-level duplication detection). Suppose that an endpoint device $D_{d,a}$ prepares to upload a file $M = m_1||m_2||\cdots||m_n$ to the cloud via a fog node F_b, the fog-level duplication detection and cloud-level duplication detection are described as follows.

1) Fog-Level Duplication Detection: We present the details of the fog-level duplication detection process as follows.

Endpoint device $D_{d,a}$: For each data block m_i of M, the endpoint device $D_{d,a}$ randomly selects $r_i \in \mathbb{Z}_N$, and computes g^{r_i} and $g^{SK_{D_{d,a}}+H_1(m_i)} \cdot {PK_{F_b}}^{r_i}$. Then it sends $\{g^{r_i}, g^{SK_{D_{d,a}}+H_1(m_i)} \cdot {PK_{F_b}}^{r_i}\}_{(1\le i\le n)}$ to the fog node F_b.

Fog node F_b: For each data block, the fog node F_b computes the fog-level data tag $Tag_{m_i}^{(f)}$ using (1). Then it determines logical indices set S_f of non-duplicated data blocks by checking whether $\{Tag_{m_i}^{(f)}\}_{(1\le i\le n)}$ exists on the fog node or not. Finally, the fog node sends the S_f to the endpoint device for further cloud-level duplication detection

$$Tag_{m_i}^{(f)} = R_{d,a} \cdot \frac{g^{SK_{D_{d,a}}+H_1(m_i)} \cdot {PK_{F_b}}^{r_i}}{g^{r_i \cdot SK_{F_b}^{-1}}} = g^{SV_{O_d}+H_1(m_i)} \tag{1}$$

2) Cloud-Level Duplication Detection: After obtaining the non-duplicated data block logical indices set S_f, the endpoint device $D_{d,a}$ generates a short hash value $sh_{m_i} = H_4(m_i)$ and a base value $bv_{m_i} = H_2(m_i) \cdot {PK_C}^{\epsilon_i}$ for each m_i ($i \in S_f$), where $\epsilon_i \in \mathbb{Z}_N$ is a random value. The endpoint device $D_{d,a}$ sends the $\{sh_{m_i}, bv_{m_i}\}_{(i\in S_f)}$ to the fog node F_b. Then for each $i \in S_f$, the fog node F_b interacts with the cloud to perform the cloud-level duplication detection, which is described as follows.

Fog node F_b: The fog node F_b sends the short hash value sh_{m_i} to the cloud to find the potential fog nodes, which have uploaded data owning the same short hash value sh_{m_i}.

CSP: The CSP retrieves the cloud-level data tags $\{Tag_{m_j}^{(c)}\} = DB_{Tag}^{(C,1)}[sh_{m_i}]$ and the fog nodes $\{F_{b'}\} = DB_{FN}^{(C,1)}[sh_{m_i}]$ of the data blocks $\{m_j\}$ from the data structure $DB^{(C,1)}$, and then sends the potential fog nodes set $\{F_{b'}\}$ to the fog node F_b.

Fog node F_b: For each potential fog node $F_{b'} \in \{F_{b'}\}$, the fog node F_b computes a data tag $Tag_{m_i^j}^{(c)} = e(bv_{m_i}^{SK_{F_b}}, UK_{b,b'})$.[1] Then the fog node F_b sends all the generated data tags $\{Tag_{m_i^j}^{(c)}\}$ to the CSP.

CSP: For each potential duplicated data $m_j \in \{m_j\}$, the CSP compares the received data tag $Tag_{m_i^j}^{(c)}$ with the cloud-level data tag $Tag_{m_j}^{(c)}$ using (2). If there is no such $Tag_{m_i^j}^{(c)}$ and $Tag_{m_j}^{(c)}$ satisfying the (2), m_i is a non-duplicated block and the CSP sets the $Res = 0$. Otherwise, m_i already exists and the CSP sets the $Res = 1$. The CSP sends the Res to the fog node F_b

$$\left(Tag_{m_i^j}^{(c)}\right)^{SK_C} \stackrel{?}{=} \left(Tag_{m_j}^{(c)}\right)^{SK_C} \tag{2}$$

If the $Res = 0$, the fog node F_b adds i into S_n. Finally, the fog node F_b can obtain a set S_n to indicate non-duplicated data

[1] If $F_{b'}$ and F_b are the same fog node, the fog node F_b computes the data tag $Tag_{m_i^j}^{(c)} = e(bv_{m_i}^{SK_{F_b}}, g)$.

blocks. Then the set $S_e = S_f - S_n$ indicates duplicated data blocks at cloud-level. The fog node F_b sends S_e and S_n to the endpoint device $D_{d,a}$.

E. Encryption Key Sharing Mechanism

We design an encryption key sharing mechanism to allow a data owner to obtain the correct key when he/she downloads the data. The detailed processes are described as follows.

Key generation: For a data block m_i, the endpoint device $D_{d,a}$ chooses two random values $\gamma_{i,1}, \gamma_{i,2} \in \mathbb{Z}_N$ and generates the encryption key as $sk_i = H_3(g^{\gamma_{i,1}} \cdot g^{\gamma_{i,2}} \cdot H_2(m_i))$. Then it encrypts m_i using an existing symmetric encryption algorithm (e.g., AES) as $c_i = Enc(sk_i, m_i)$.

Key sharing: The endpoint device $D_{d,a}$ shares the encryption key of the data block m_i as follows.

- The $D_{d,a}$ selects two random values ε_1, $\varepsilon_2 \in \mathbb{Z}_N$ and encrypts the $g^{\gamma_{i,1}}$, $g^{\gamma_{i,2}}$ to obtain $ck_i^{(c)} = (g^{\gamma_{i,1}} \cdot {PK_C}^{\varepsilon_1}, g^{\varepsilon_1})$ and $ck_i^{(f)} = (g^{\gamma_{i,2}} \cdot {PK_{F_b}}^{\varepsilon_2}, g^{\varepsilon_2})$ using the public keys of the CSP and fog node F_b, respectively.
- The $D_{d,a}$ selects a random $\varepsilon_3 \in \mathbb{Z}_N$ and encrypts $H_2(m_i)$ using the public key of the data owner O_d as $ck_i^{(o_1)} = (H_2(m_i) \cdot {PK_{O_d}}^{\varepsilon_3}, g^{\varepsilon_3})$.
- The $D_{d,a}$ sends the $ck_i^{(f)}$ to the fog node F_b and sends $ck_i^{(o_1)}$, $ck_i^{(c)}$ to the CSP via the fog node F_b.
- The F_b decrypts the $ck_i^{(f)}$ as (3) and stores $g^{\gamma_{i,2}}$

$$g^{\gamma_{i,2}} = \frac{g^{\gamma_{i,2}} \cdot {PK_{F_b}}^{\varepsilon_2}}{g^{\varepsilon_2 \cdot SK_{F_b}^{-1}}} \tag{3}$$

- The F_b selects a random $\varepsilon_4 \in \mathbb{Z}_N$ and encrypts the $g^{\gamma_{i,2}}$ using the public key of O_d as $ck_i^{(o_2)} = (g^{\gamma_{i,2}} \cdot {PK_{O_d}}^{\varepsilon_4}, g^{\varepsilon_4})$. It then uploads the $ck_i^{(o_2)}$ to the CSP.
- The CSP decrypts the $ck_i^{(c)}$ as (4) and then stores $g^{\gamma_{i,1}}$, $ck_i^{(o_1)}$, and $ck_i^{(o_2)}$

$$g^{\gamma_{i,1}} = \frac{g^{\gamma_{i,1}} \cdot {PK_C}^{\varepsilon_1}}{g^{\varepsilon_1 \cdot q}} \tag{4}$$

Key recovery: When the data owner O_d downloads the data, the CSP selects a random value $\varepsilon_5 \in \mathbb{Z}_N$ and encrypts the $g^{\gamma_{i,1}}$ to obtain the $ck_i^{(o_3)} = (g^{\gamma_{i,1}} \cdot {PK_{O_d}}^{\varepsilon_5}, g^{\varepsilon_5})$ using the public key of the data owner O_d. Then the CSP sends $ck_i^{(o_1)}$, $ck_i^{(o_2)}$, and $ck_i^{(o_3)}$ to the data owner O_d, and the data owner O_d can obtain the $g^{\gamma_{i,1}}$, $g^{\gamma_{i,2}}$ and $H_2(m_i)$ using (5), (6), and (7), respectively. Finally, the data owner obtains the encryption key $sk_i = H_3(g^{\gamma_{i,1}} \cdot g^{\gamma_{i,2}} \cdot H_2(m_i))$

$$g^{\gamma_{i,1}} = \frac{g^{\gamma_{i,1}} \cdot {PK_{O_d}}^{\varepsilon_5}}{g^{\varepsilon_5 \cdot SK_{O_d}}} \tag{5}$$

$$g^{\gamma_{i,2}} = \frac{g^{\gamma_{i,2}} \cdot {PK_{O_d}}^{\varepsilon_4}}{g^{\varepsilon_4 \cdot SK_{O_d}}} \tag{6}$$

$$H_2(m_i) = \frac{H_2(m_i) \cdot {PK_{O_d}}^{\varepsilon_3}}{g^{\varepsilon_3 \cdot SK_{O_d}}} \tag{7}$$

The data owner can decrypt the c_i using sk_i and obtain the plaintext data $m_i = Dec(sk_i, c_i)$.

F. Data Reliability Verifying Mechanism

We design the data reliability verification mechanism to allow a data owner to verify the number and content of the downloaded files compared to the files uploaded by his/her endpoint devices.

1) Auxiliary Data Generation: When uploading a file $M = m_1||m_2||\cdots||m_n$ to the CSP via the fog node F_b, the endpoint device $D_{d,a}$ computes the auxiliary data $AD_M^{(D_{d,a})}$ for its data owner to verify file content as follows.

- The device $D_{d,a}$ computes $ad_M = H_2(O_d||D_{d,a}||ord) \cdot H_2(M)$, where the ord indicates the order of file uploaded by the endpoint device $D_{d,a}$.
- The device $D_{d,a}$ generates the auxiliary data $AD_M^{(D_{d,a})} = (O_d||D_{d,a}||ord, (ad_M)^{SK_{D_{d,a}}})$ and uploads $AD_M^{(D_{d,a})}$ to the CSP via the fog node F_b.

When the data owner accesses the data collected by $D_{d,a}$, the data owner randomly selects $\delta \in \mathbb{Z}_N$ and sends $(\delta, D_{d,a}, F_b)$ to the CSP. The CSP sends the $(\delta, O_d, D_{d,a})$ to the fog node. The fog node F_b computes the auxiliary data for the data owner to verify file number as follows.

- The fog node F_b retrieves the number $fn_{D_{d,a}} = DB_{FN}^{(F,2)}[D_{d,a}]$ of the files uploaded by the device $D_{d,a}$ and computes $ad_{D_{d,a}} = H_2(F_b||D_{d,a}||fn_{D_{d,a}}||\delta)$.
- The fog node F_b generates $AD_{D_{d,a}}^{(F_b)} = (F_b||D_{d,a}||\delta, (ad_{D_{d,a}})^{SK_{F_b}^{-1}})$ and sends $AD_{D_{d,a}}^{(F_b)}$ to the CSP.

2) File Content Verifying: Suppose that the data owner O_d obtains the ciphertext $C = c_1||c_2||\cdots||c_n$ and $AD_M^{(D_{d,a})}$ generated by the endpoint device $D_{d,a}$. After decrypting the ciphertext and obtaining the $M' = m'_1||m'_2||\cdots||m'_n$, the data owner O_d computes the $ad_{M'} = H_2(O_d||D_{d,a}||ord) \cdot H_2(M')$ and verifies the file content by checking whether the (8) holds

$$e\left(ad_{M'}, g^{SK_{D_{d,a}}}\right) \stackrel{?}{=} e\left((ad_M)^{SK_{D_{d,a}}}, g\right) \tag{8}$$

If the verification is valid, the downloaded C is consistent with the file collected by the device $D_{d,a}$.

3) File Number Verifying: Suppose that the data owner O_d obtains $fn^*_{D_{d,a}}$ files uploaded by the endpoint device $D_{d,a}$, and the auxiliary data $AD_{D_{d,a}}^{(F_b)}$ generated by the F_b from the CSP. The data owner computes $ad^*_{D_{d,a}} = H_2(F_b||D_{d,a}||fn^*_{D_{d,a}}||\delta)$ and verifies whether the (9) holds

$$e\left(ad^*_{D_{d,a}}, PK_{F_b}\right) \stackrel{?}{=} e\left((ad_{D_{d,a}})^{SK_{F_b}^{-1}}, g\right) \tag{9}$$

If the verification is valid, the number of downloaded files is consistent with the number of files uploaded by the endpoint device $D_{d,a}$.

G. Workflow of FCDedup

1) Workflow of Data Uploading: Suppose that an endpoint device $D_{d,a}$ collects a file $M = m_1||m_2||\cdots||m_n$ and uploads it to the cloud via the fog node F_b, the data uploading workflow presents as follows.

- The endpoint device $D_{d,a}$, fog node F_b and the CSP perform two-level duplication detection as the description of Section III-D. Then they can obtain a set S_e indicating the

logical indices of duplicated blocks at cloud-level and a set S_n indicating the logical indices of non-duplicated blocks at cloud-level. The endpoint device $D_{d,a}$ and fog node F_b can also obtain a set S_f indicating the logical indices of non-duplicated blocks at fog-level, and a set $S_{e'}$ indicating the logical indices of duplicated blocks at fog-level, where $S_e + S_n = S_f$, $S_f + S_{e'} = S_u$, and $S_u = \{1, 2, \ldots, n\}$ is the universe. To enable the CSP to construct the entire file, for each duplicated block m_i ($i \in S_{e'}$) at fog-level, the fog node F_b sends the unique block index bi_{m_i} to the CSP.

- For each non-duplicated block m_i ($i \in S_n$), the device $D_{d,a}$ performs key generation and key sharing as the description of Section III-E. The device $D_{d,a}$ also sends the data ciphertext to the cloud via the fog node F_b. Besides, the fog node F_b computes the cloud-level tag $Tag_{m_i}^{(c)} = e(bv_{m_i}^{SK_{F_b}}, g)$ and sends it to the cloud.
- For each duplicated block m_i ($i \in S_e + S_{e'}$), the device $D_{d,a}$ only uploads the encrypted data hash value $ck_i^{(o_1)}$ to the CSP via F_b. The CSP sends the (O_d, bi_{m_i}) to the fog node $F_{b'}$ that first uploads the data block m_i. Then $F_{b'}$ encrypts $DB_{KS}^{(F,1)}[bi_{m_i}]$ using the public key of O_d and sends the ciphertext $ck_i^{(o_2)}$ to the CSP.
- The endpoint device $D_{d,a}$ generates the auxiliary data $AD_M^{(D_{d,a})}$ of the file M as the description of Section III-F and uploads it to the CSP via the fog node F_b. Finally, the CSP can generate a metadata file MF_M of M by storing O_d, $AD_M^{(D_{d,a})}$, and $\{bi_{m_i}, ck_i^{(o_1)}, ck_i^{(o_2)}\}_{(1\le i\le n)}$ in it. The metadata file is used by the cloud to reconstruct the entire ciphertext file and also by the data owner to recover the plaintext file and verify the file content in data retrieval process.

2) Workflow of Data Retrieval: Suppose that a data owner O_d retrieves all the files collected by an endpoint device $D_{d,a}$ under the fog node F_b, the data retrieval workflow is described as follows.

- The data owner O_d randomly selects $\delta \in \mathbb{Z}_N$ and sends δ, $D_{d,a}$ and F_b to the CSP.
- For each file collected by the device $D_{d,a}$, the CSP retrieves the unique block indices $\{bi_{m_i}\}_{(1\le i\le n)}$, encrypted key shares $\{ck_i^{(o_1)}, ck_i^{(o_2)}\}_{(1\le i\le n)}$ and file content verification auxiliary data $AD_M^{(D_{d,a})}$ from the metadata file MF_M, and then constructs the ciphertext file $C = c_1||c_2||\cdots||c_n$ using the indices $\{bi_{m_i}\}_{(1\le i\le n)}$. The CSP also retrieves $\{g^{\gamma_{i,1}} = DB_{KS}^{(C,1)}[bi_{m_i}]\}_{(1\le i\le n)}$, and encrypts them to obtain $\{ck_i^{(o_3)}\}_{(1\le i\le n)}$ using the public key of the data owner. Besides, the CSP sends $(O_d, D_{d,a}, \delta)$ to the fog node F_b.
- The fog node F_b computes the auxiliary data $AD_{D_{d,a}}^{(F_b)}$ for the endpoint device $D_{d,a}$ as the description of Section II-I-F1 and sends $AD_{D_{d,a}}^{(F_b)}$ to the CSP.
- For each file collected by the endpoint device $D_{d,a}$, the CSP sends $\{ck_i^{(o_1)}, ck_i^{(o_2)}, ck_i^{(o_3)}\}_{(1\le i\le n)}$, C and $AD_M^{(D_{d,a})}$ to the data owner O_d. Besides, the CSP also sends the auxiliary data $AD_{D_{d,a}}^{(F_b)}$ generated by the fog node F_b to the data owner.
- The data owner can first verify the file number as described in Section III-F3. For each file C, the data owner can recover the encryption key and decrypt the ciphertext as the description of key recovery in Section III-E. Then the data owner can verify the file content as the description of Section III-F2.

IV. Scheme Analysis

A. Correctness

1) Fog-Level Duplication Detection: Suppose that an endpoint device $D_{d,a}$ of the owner O_d has uploaded a data block m_i to the fog node F_b. The endpoint device $D_{d,a}$ has sent the $(g^{r_i}, g^{SK_{D_{d,a}}+H_1(m_i)} \cdot PK_{F_b}{}^{r_i})$ to the fog node F_b. Then the F_b can obtain the fog-level data tag $Tag_{m_i}^{(f)} = g^{SV_{O_d}+H_1(m_i)}$ using (1). When another endpoint device $D_{d,a'}$ of the owner O_d collects the data block m_i' and sends $(g^{r_i'}, g^{SK_{D_{d,a'}}+H_1(m_i')} \cdot PK_{F_b}{}^{r_i'})$ to the fog node F_b to detect duplication, the fog node F_b can compute the $Tag_{m_i'}^{(f)}$ as (10), where $R_{d,a'}$ is the registration ticket of $D_{d,a'}$ received from the owner O_d

$$Tag_{m_i'}^{(f)} = \frac{g^{SV_{O_d}-SK_{D_{d,a'}}} \cdot g^{SK_{D_{d,a'}}+H_1(m_i')} \cdot g^{SK_{F_b}^{-1}\cdot r_i'}}{g^{r_i'\cdot SK_{F_b}^{-1}}} = g^{SV_{O_d}+H_1(m_i')} \tag{10}$$

When the m_i and m_i' are the same data, the $Tag_{m_i}^{(f)}$ equals to the $Tag_{m_i'}^{(f)}$. Then the fog node can detect that the endpoint devices $D_{d,a}$ and $D_{d,a'}$ upload the identical data block. Thus, the fog node can detect duplicated data collected by a data owner's endpoint devices under its coverage area.

2) Cloud-Level Duplication Detection: Suppose that the fog node $F_{b'}$ is the first fog node that uploads the data block m_j and it has uploaded a cloud-level data tag $Tag_{m_j}^{(c)}$ to the cloud. When a fog node F_b uploads a data block m_i, it computes a data tag $Tag_{m_i^j}^{(c)}$, and the CSP detects duplication by comparing the $Tag_{m_j}^{(c)}$ and $Tag_{m_i^j}^{(c)}$ using (2). The CSP computes the two sides of (2) as follows:

$$\left(Tag_{m_i^j}^{(c)}\right)^{SK_C} = e\left(H_2(m_i), g\right)^{SK_{F_{b'}}\cdot SK_C} \cdot e(g,g)^{\epsilon_i \cdot SK_{F_{b'}}\cdot p\cdot q} = e\left(H_2(m_i), g\right)^{SK_{F_{b'}}\cdot SK_C}$$

$$\left(Tag_{m_j}^{(c)}\right)^{SK_C} = e\left(H_2(m_j), g\right)^{SK_{F_{b'}}\cdot SK_C} \cdot e(g,g)^{\epsilon_i' \cdot SK_{F_{b'}}\cdot p\cdot q} = e\left(H_2(m_j), g\right)^{SK_{F_{b'}}\cdot SK_C}$$

Obviously, the two sides of (2) are equal only when $m_i = m_j$. Thus, the CSP can detect duplication through the (2).

B. Security

1) Data Confidentiality: Since the well-known AES-256 is used to encrypt data and it is semantically secure, the adversary can only obtain the file content through BFA. Given the plaintext

space $\mathcal{M} = \{\hat{m}_1, \ldots, \hat{m}_\tau\}$, the adversary of BFA aims to determine which data $\hat{m}_t \in \mathcal{M}$ corresponds to the targeted ciphertext c_i.

Attacks from CSP: First, we consider the BFA launched by the CSP using its background knowledge. The ciphertext c_i and the cloud-level data tag $Tag_{m_i}^{(c)} = e((H_2(m_i) \cdot PK_C{}^{\epsilon_i})^{SK_{F_b}}, g)$ can be used by the CSP to launch BFA.

We first prove that the cloud-level data tag cannot be used by the CSP to launch BFA. The CSP can compute $(Tag_{m_i}^{(c)})^{SK_C}$ to eliminate the random value and obtain $e(H_2(m_i), g)^{SK_{F_b} \cdot SK_C}$. To compute $e(H_2(\hat{m}_t), g)^{SK_{F_b} \cdot SK_C}$ and compare it with $e(H_2(m_i), g)^{SK_{F_b} \cdot SK_C}$, the CSP needs to obtain either SK_{F_b} or $g^{SK_{F_b}}$. It is obvious that the CSP cannot obtain the secret key SK_{F_b} of the fog node. Furthermore, the complexity of computing $g^{SK_{F_b}}$ from g, $g^{SK_{F_{b'}}^{-1}}$ and $g^{SK_{F_b} \cdot SK_{F_{b'}}^{-1}}$ equals to that of solving the DCDH problem. However, according to Definition 2, there is no probabilistic polynomial-time algorithm to solve the DCDH problem. Thus, the CSP cannot launch BFA using the tag.

Then we prove that the ciphertext c_i cannot be used by the CSP to launch BFA. The encryption key is $sk_i = H_3(g^{\gamma_{i,1}} \cdot g^{\gamma_{i,2} \cdot H_2(m_i)})$. Since the CSP only has $g^{\gamma_{i,1}}$, it cannot generate the encryption key $\hat{sk}_t = H_3(g^{\gamma_{i,1}} \cdot g^{\gamma_{i,2} \cdot H_2(\hat{m}_t)})$ to encrypt $\hat{m}_t$ and compare the ciphertext $\hat{c}_t$ with c_i. Thus, the CSP cannot launch BFA using the ciphertext.

Attacks from fog node: Next, we consider the BFA launched by the fog node F_b that uploads the ciphertext c_i. The fog-level data tag $Tag_{m_i}^{(f)} = g^{SV_{O_d} + H_1(m_i)}$, the base value $bv_{m_i} = H_2(m_i) \cdot PK_C{}^{\epsilon_i}$ and the ciphertext c_i can be used to launch BFA. Since the fog node cannot obtain the SV_{O_d}, it cannot compute $g^{SV_{O_d} + H_1(\hat{m}_t)}$ and compare it with the $Tag_{m_i}^{(f)}$. Thus, the fog node F_b cannot launch BFA using $Tag_{m_i}^{(f)}$. Since a random item $g^{q \cdot \epsilon_i}$ is involved in bv_{m_i} and the fog node cannot eliminate the random item, the fog node also cannot launch BFA using bv_{m_i}. Besides, since the fog node only has $g^{\gamma_{i,2}}$, it cannot generate the encryption key $\hat{sk}_t = H_3(g^{\gamma_{i,1}} \cdot g^{\gamma_{i,2} \cdot H_2(\hat{m}_t)})$, and thus cannot encrypt $\hat{m}_t$ to compare the ciphertext $\hat{c}_t$ with c_i. Thus, the fog node cannot launch BFA using its known information.

2) File Content Reliability: Assume that an endpoint device $D_{d,a}$ has uploaded a file M to the CSP via the fog node, and the data owner O_d obtains M' by decrypting the downloaded ciphertext. As described in Section III-F, the data owner can get the auxiliary data $AD_M^{(D_{d,a})} = (O_d || D_{d,a} || ord, (ad_M)^{SK_{D_{d,a}}})$ generated by the device $D_{d,a}$. We first prove that the auxiliary data cannot be falsified by the CSP.

Theorem 1: If the CDH problem is hard in bilinear groups, no adversary can falsify the auxiliary data generated by the endpoint device.

Proof: The adversary can obtain the following information: $(ad_M)^{SK_{D_{d,a}}}$, O_d, $D_{d,a}$ and ord, where $ad_M = H_2(O_d || D_{d,a} || ord) \cdot H_2(M)$. It aims to generate forged $AD_{M'}^{(D_{d,a})} = (O_d || D_{d,a} || ord, (ad_{M'})^{SK_{D_{d,a}}})$, where $ad_{M'} = H_2(O_d || D_{d,a} || ord) \cdot H_2(M')$. For convenience, we use g_1 and g_0 to replace $ad_{M'}$ and ad_M, respectively. The complexity of computing ${g_1}^{SK_{D_{d,a}}}$ from ${g_0}^{SK_{D_{d,a}}}$ and g_1 equals to the complexity of solving the CDH problem.

According to Definition 1, there is no probability polynomial-time algorithm to solve the CDH problem. Thus, the adversary cannot falsify the auxiliary data generated by the endpoint device.

Then we prove that the data owner can verify file content using the auxiliary data $AD_M^{(D_{d,a})}$. The data owner verifies the content reliability by checking whether (8) holds. The data owner computes the two sides of (8) as follows:

$$e\left(ad_{M'}, g^{SK_{D_{d,a}}}\right) = e\left(H_2(O_d || D_{d,a} || ord) \cdot H_2(M'), g\right)^{SK_{D_{d,a}}}$$

$$e\left((ad_M)^{SK_{D_{d,a}}}, g\right) = e\left(H_2(O_d || D_{d,a} || ord) \cdot H_2(M), g\right)^{SK_{D_{d,a}}}$$

When $M \neq M'$, the two sides of (8) are not equal. Thus, the data owner can verify the content reliability.

3) File Number Reliability: Assume that an endpoint device $D_{d,a}$ of the data owner O_d has uploaded $fn_{D_{d,a}}$ files and the CSP returns $fn_{D_{d,a}}^*$ files to O_d in the retrieval process. As described in Section III-F, the O_d can get the auxiliary data $AD_{D_{d,a}}^{(F_b)} = (F_b || D_{d,a} || \delta, (ad_{D_{d,a}})^{SK_{F_b}^{-1}})$ generated by the fog node F_b. We first prove that the CSP cannot falsify the auxiliary data.

Theorem 2: If the CDH problem is hard in bilinear groups, no adversary can falsify the auxiliary data generated by the fog node.

Proof: The adversary has the $PK_{F_b} = g^{SK_{F_b}^{-1}}$, F_b, $D_{d,a}$, δ and $fn_{D_{d,a}}^*$. It aims to generate the forged ${AD_{D_{d,a}}^{(F_b)}}^* = (F_b || D_{d,a} || \delta, (ad_{D_{d,a}}^*)^{SK_{F_b}^{-1}})$ to pass the verification of the (9), where $ad_{D_{d,a}}^* = H_2(F_b || D_{d,a} || fn_{D_{d,a}}^* || \delta)$. The complexity of computing $(ad_{D_{d,a}}^*)^{SK_{F_b}^{-1}}$ from $g^{SK_{F_b}^{-1}}$, g and $ad_{D_{d,a}}^*$ equals to that of solving the CDH problem.

According to Definition 1, there is no probability polynomial-time algorithm to solve the CDH problem. Thus, the adversary cannot falsify the auxiliary data.

Then we prove that the data owner can verify file number using the auxiliary data $AD_{D_{d,a}}^{(F_b)}$. The data owner verifies the file number reliability by checking whether (9) holds. The data owner computes the two sides of (9) as follows:

$$e\left(ad_{D_{d,a}}^*, PK_{F_b}\right) = e\left(H_2\left(F_b || D_{d,a} || fn_{D_{d,a}}^* || \delta\right), g\right)^{SK_{F_b}^{-1}}$$

$$e\left((ad_{D_{d,a}})^{SK_{F_b}^{-1}}, g\right) = e\left(H_2\left(F_b || D_{d,a} || fn_{D_{d,a}} || \delta\right), g\right)^{SK_{F_b}^{-1}}$$

When $fn_{D_{d,a}}^* \neq fn_{D_{d,a}}$, the two sides of (9) are not equal. Thus, a data owner can verify the file number reliability in our FCDedup.

C. Discussion

Our FCDedup can be applied to many application scenarios such as health monitoring shown in Fig. 3. In a health monitoring

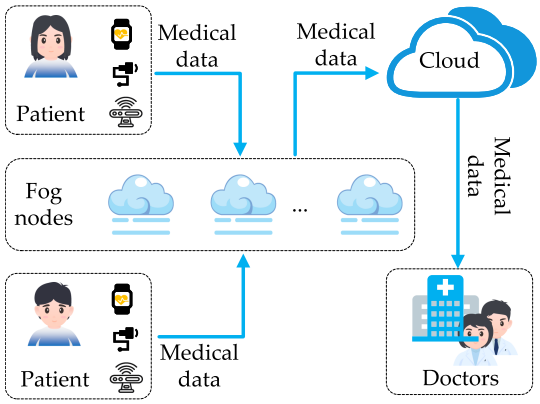


Fig. 3. Health monitoring application scenario.

system, the wearable medical sensors (i.e., endpoint devices) generate and send medical data to the cloud for storage through fog nodes. The wearable medical sensors encrypt the collected medical data to protect data confidentiality before uploading. The data owners (e.g., the authorized hospitals or doctors) can download these encrypted medical data and recover the original content. Our FCDedup can offer benefits to this scenario in terms of communication cost and storage cost. By performing two-level duplication detection, both the fog nodes and cloud can detect duplicated data blocks using the data tags before uploading. For duplicated data blocks, the endpoint devices or the fog nodes only need to transmit auxiliary information instead of the entire block. Since medical data such as computed tomography often contain many duplicated data blocks, our FCDedup can significantly reduce the storage cost of the cloud server and the bandwidth cost of the system.

Note that the number of fog nodes is not a bottleneck affecting the efficiency of our system. Because only the generation of joint public keys involves the pairwise communication between each couple of fog nodes, which is only executed once at the system initialization process. Once the joint public keys are generated, they can be used directly in duplication detection without further pairwise communication. Thus, our FCDedup can be applied to a fog-assisted storage system with any number of fog nodes.

V. Performance Evaluation

A. Complexity Analysis

1) Computation Cost: Let h_{bp}, h_{enc}, h_{dec}, h_{m^*}, h_h, h_e, h_d, h_a, and h_{eq} denote the computation costs for a bilinear pairing operation, one-time AES-256 encryption operation, one-time AES-256 decryption operation, a multiplication operation over $\mathbb{G}_1$, a hash operation, an exponentiation operation, a division operation over $\mathbb{G}_1$, an addition operation over $\mathbb{Z}_N$, and an equality detection operation over two elements of $\mathbb{G}_2$, respectively. *Data uploading*: The fog-level duplication detection causes $3h_e + h_{m^*} + h_h + h_a$ and $h_{m^*} + h_e + h_d$ computation cost to the endpoint device and fog node, respectively. The cloud-level duplication detection causes $2h_h + h_{m^*} + h_e$, $h_e + No_{cn}h_{bp}$ and $No_{cn}(2h_e + h_{eq})$ computation cost to the endpoint device, fog node and CSP, respectively, where No_{cn} is the number of data blocks that have the same short hash value with the data block to be uploaded. If the data block is duplicated, the endpoint device only needs to encrypt the data hash value, which has a cost of $h_h + 2h_e + h_{m^*}$. If the data is non-duplicated, the endpoint device needs to perform the key generation, encryption and key sharing with a total cost of $2h_h + 8h_e + 5h_{m^*} + h_{enc}$. The fog node retrieves its key share with a cost of $h_d + h_e$ and encrypts the key share with a cost of $2h_e + h_{m^*}$. The CSP also retrieves its key share with a computation cost of $h_d + h_e$.

Thus, for a duplicated block at fog-level, it causes $5h_e + 2h_{m^*} + 2h_h + h_a$ and $h_{m^*} + h_e + h_d$ computation cost to the endpoint device and fog node, respectively. For a duplicated block at cloud-level, it causes $6h_e + 3h_{m^*} + 4h_h + h_a$, $2h_e + h_{m^*} + h_d + No_{cn}h_{bp}$, and $No_{cn}(h_{eq} + 2h_e)$ computation cost to the endpoint device, fog node, and CSP, respectively. For a non-duplicated block, it causes $12h_e + 7h_{m^*} + 5h_h + h_a + h_{enc}$, $5h_e + 2h_{m^*} + 2h_d + No_{cn}h_{bp}$, and $No_{cn}(h_{eq} + 2h_e) + h_e + h_d$ computation cost to the endpoint device, fog node, and CSP, respectively.

Data retrieval: we assume that a data owner retrieves No_f files (each file has n data blocks) collected by an endpoint device and analyze the computation cost. The CSP encrypts its $No_f \cdot n$ key shares with a cost of $No_f \cdot n(2h_e + h_{m^*})$. The fog node generates auxiliary data with a cost of $h_h + h_e$. The data owner checks the file number, recovers the decryption keys, decrypts the No_f files, and checks the file content, which causes $No_f \cdot n(3h_e + 3h_d + 2h_{m^*} + h_h + h_{dec}) + (No_f + 1)(2h_{bp} + h_{eq}) + (2No_f + 1)h_h + No_f h_{m^*} + h_e$ computation cost to the data owner.

2) Communication Overhead: Data uploading: During the fog-level duplication detection, the communication overhead between the endpoint device and fog node is $2|\mathbb{G}_1|$ bits. During the cloud-level duplication detection, the communication costs of the endpoint device, fog node, and CSP are $\iota + |\mathbb{G}_1|$ bits, $No_{cn}(|\mathbb{G}_2| + |F|) + 2\iota + |\mathbb{G}_1|$ bits, and $No_{cn}(|F| + |\mathbb{G}_2|) + \iota$ bits, respectively. For a non-duplicated data block, the system should transfer two encrypted key shares, encrypted data hash value, ciphertext and data owner information. This causes $6|\mathbb{G}_1| + |c|$ bits, $12|\mathbb{G}_1| + 2|c| + |O|$ bits, and $6|\mathbb{G}_1| + |c| + |O|$ bits communication cost to the endpoint device, fog node, and CSP, respectively. The CSP also needs to send a unique block index with $|bi|$ bits to the fog node. For a duplicated data block, the system only transfers the encrypted data hash value and data owner information. This causes $2|\mathbb{G}_1|$ bits, $4|\mathbb{G}_1| + |O|$ bits, and $2|\mathbb{G}_1| + |O|$ bits communication cost to the endpoint device, fog node, and CSP, respectively. The CSP also needs to retrieve an encrypted key share from the first fog node that uploads the data block, which has a communication cost of $|bi| + |O| + 2|\mathbb{G}_1|$ bits. Besides, if the data block is duplicated at fog-level, the fog node needs to send the block index with $|bi|$ bits to the CSP.

Thus, for a duplicated data block at fog-level, the communication costs for the endpoint device, fog node and the CSP are $4|\mathbb{G}_1|$ bits, $6|\mathbb{G}_1| + |bi| + |O|$ bits, and $4|\mathbb{G}_1| + 2|bi| + 2|O|$ bits, respectively. For a duplicated data block at cloud-level, the communication costs for the endpoint device, fog node and CSP are $5|\mathbb{G}_1| + \iota$ bits, $No_{cn}(|\mathbb{G}_2| + |F|) + |O| + 2\iota + 7|\mathbb{G}_1|$

bits, and $No_{cn}(|F| + |\mathbb{G}_2|) + 4|\mathbb{G}_1| + \iota + |bi| + 2|O|$ bits, respectively. For a non-duplicated data block, the communication costs for the endpoint device, the fog node and the CSP are $9|\mathbb{G}_1| + |c| + \iota$ bits, $No_{cn}(|\mathbb{G}_2| + |F|) + |O| + |bi| + 2\iota + 15|\mathbb{G}_1| + 2|c|$ bits, and $No_{cn}(|F| + |\mathbb{G}_2|) + \iota + 6|\mathbb{G}_1| + |bi| + |O| + |c|$ bits, respectively.

Data retrieval: We assume that a data owner retrieves No_f files (each file has n data blocks) collected by an endpoint device and analyze the communication overhead. Transferring the data downloading request and the file number verification auxiliary data causes $|\mathbb{Z}_N| + |D| + |F| + |AD_D|$ bits, $|\mathbb{Z}_N| + |O| + |D| + |AD_D|$ bits and $2|\mathbb{Z}_N| + 2|D| + |F| + |O| + 2|AD_D|$ bits communication costs to the data owner, the fog node and the CSP, respectively. Transferring the ciphertext, key shares and file content verification auxiliary data causes $n \cdot No_f|c| + 6n \cdot No_f|\mathbb{G}_1| + No_f|AD_M|$ bits communication cost to the data owner and the CSP. Thus, data retrieval causes $No_f(|AD_M| + n|c| + 6n|\mathbb{G}_1|) + |AD_D| + |\mathbb{Z}_N| + |D| + |F|$ bits, $|AD_D| + |\mathbb{Z}_N| + |O| + |D|$ bits, and $No_f(|AD_M| + n|c| + 6n|\mathbb{G}_1|) + 2(|AD_D| + |\mathbb{Z}_N| + |D|) + |F| + |O|$ bits communication cost to the data owner, fog node, and CSP, respectively.

3) Storage Cost: We present the additional storage costs of both the fog nodes and CSP, excluding the cost of data ciphertexts. For a file with n blocks, the fog node stores the fog-level tags, block indices, encryption key shares, and data owner information, which causes $2n|\mathbb{G}_1| + n|bi| + n|O|$ bits storage cost. The CSP stores the short hash values, uploaders information, block indices, encryption key shares, cloud-level tags, owner information, link of metadata file and metadata file, which causes $n(\iota + 2|bi| + |\mathbb{G}_2|) + 5n|\mathbb{G}_1| + (n + 1)|F| + 2|O| + |D| + |Link| + |AD_M|$ bits storage cost. Compared to the scheme in [33], FCDedup causes $n(|\mathbb{G}_1| + |bi|)$ bits additional storage cost to the fog node and $n \cdot \iota + 5n|\mathbb{G}_1| + |Link| + |AD_M|$ bits additional storage cost to the cloud. However, these additional storage costs are negligible compared to the cost of ciphertexts. FCDedup achieves inter-deduplication across different data owners, and thus has a lower overall storage cost than the scheme in [33], which is illustrated in Section V-B2.

B. Simulation Evaluation

1) Prototype Implementation: We implement a prototype of FCDedup using Alibaba cloud object storage service (AlibabaCloud OSS) [27] as an implementation example.[2] Our implementation relies on the OpenSSL Library [34], GNU Multiple Precision Arithmetic (GMP) Library [35] and Pairing Based Cryptography (PBC) Library [36]. Since other commercial cloud storage systems such as Dropbox [37], Google Drive [38] provide APIs similar to the AlibabaCloud OSS, our FCDedup can also be compatible with them.

Fig. 4 shows the deploying structure of FCDedup and we use the AlibabaCloud OSS as the cloud storage backend. The cloud plugin program calls the APIs of AlibabaCloud OSS to perform the storage and retrieval of ciphertexts and metadata files. We utilize a windows laptop with 4 cores Intel Core i7-7560 U processor (1.80 GHz) and 8 GB of memory to perform the cloud plugin program. The fog plugins of FCDedup fog nodes are implemented in a windows laptop with 4 cores Intel Core i5-6200 processor and 8 GB of memory. The index structures of the cloud and fog nodes are constructed using MySQL database [39]. The programs of the endpoint devices and data owners are implemented in a MacOS laptop with 2.00 GHz Intel Core i5-1038NG7 processor and 16 GB of memory. All the laptops are deployed in the same wireless LAN with 20 Mbps bandwidth.

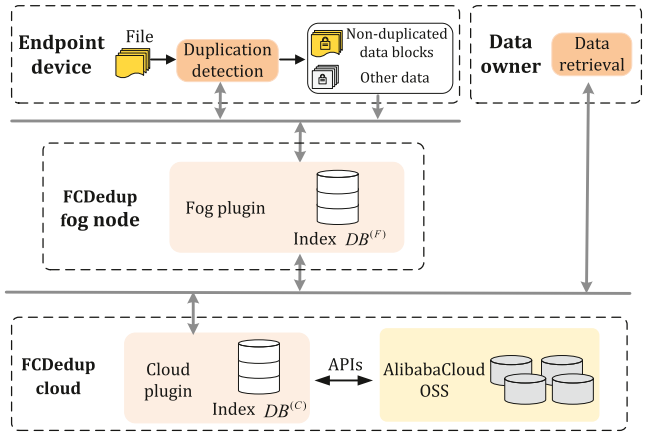


Fig. 4. Implementation of FCDedup using Alibaba cloud object storage service (AlibabaCloud OSS).

We set the bit length κ of the secure parameter as 128, the bit length λ of the encryption key as 256, and the bit length ι of the short hash value as 10. The block size is set as 64 KB. The ID numbers of the fog nodes, endpoint devices, and data owners are encoded using 20 bits. The block indices in the CSP and fog nodes are encoded using 128 bits. Besides, we use the AES-256 to encrypt data. Although, many secure deduplication schemes have been developed for traditional cloud storage with a two-layer architecture, they are inapplicable to the fog-assisted cloud storage with three layers (i.e., cloud server, fog node and endpoint device). We compare our scheme with the scheme in [33] in experimental analysis, since only this scheme focuses on secure deduplication in fog-assisted cloud storage, which aligns with the context of our work. For easily presenting, we simply refer to the scheme of Zhang et al. [33] as Zhang22 in the subsequent description.

To evaluate the efficiency of our FCDedup at different deduplciaton rates, We conduct experiments using five datasets (i.e., DB1-DB5) derived from LCTSC [40]. The DB1-DB5 datasets were created by replicating the files in LCTSC dataset 1-5 times. Table II lists the key attributes of these datasets. In our simulation, we consider four data owners and four fog nodes, with each data owner deploying four endpoint devices under each fog node. In the following experiments, each dataset is sequentially divided into four subsets, ensuring an equal number of files in each subset. Each subset is assigned to a specific data owner and further divided into 16 parts, with each part containing

[2]The experimental result data and source code of FCDedup prototype are avaliable at https://github.com/MYSong6/FCDedup.

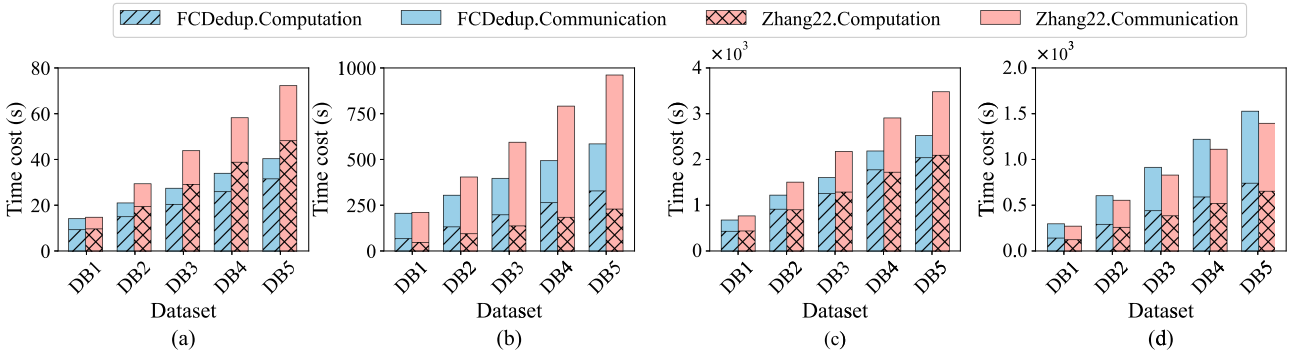


Fig. 5. Comparisons of actual time cost. (a) The average time cost per endpoint device during the data uploading process; (b) the average time cost per fog node during the data uploading process; (c) the time cost of the cloud during the data uploading process; (d) the average time cost per data owner during the data retrieval process.

TABLE II
EVALUATION DATASETS

Dataset	DB1	DB2	DB3	DB4	DB5
Size (GB)	5.11	10.22	15.33	20.44	25.55
Number of files	9569	19138	28707	38276	47845
Duplication rate	0.247	0.623	0.749	0.811	0.849

the same number of files. Each part is uploaded by an endpoint device belonging to the corresponding data owner.

2) Simulation Results: We evaluate the actual performance of our implemented prototype.

Overall time cost: We evaluate the average time cost of the FCDedup from the perspectives of the endpoint device, fog node, CSP, and data owner.

We first measure the average time cost of the endpoint devices during the data uploading process and Fig. 5(a) shows the results. The FCDedup has a lower time cost than Zhang22 in the five datasets. This is because FCDedup supports both intra-deduplication within a single data owner and inter-deduplication across different data owners, and the scheme Zhang22 supports only intra-deduplication. Then the duplicated data blocks across four data owners do not need to be encrypted and uploaded in our FCDedup.

We then test the average time cost of the fog nodes during the data uploading process and Fig. 5(b) shows the results. Our FCDedup requires more computation time to achieve inter-deduplication across different data owners compared to the scheme Zhang22. However, it requires less communication time since the duplicated blocks across four data owners do not need to be uploaded. Besides, our FCDedup has a lower total time cost than the scheme Zhang22.

Fig. 5(c) shows the time cost of the CSP during the data uploading process. Our FCDedup has a very similar computation time and a lower communication time compared to the scheme Zhang22. Because the duplicated blocks across four data owners do not need to be uploaded.

Finally, we test the average time cost of the data owners during the data retrieval process, and Fig. 5(d) shows the results. Our FCDedup has a slightly higher time cost than the scheme Zhang22. This is because our FCDedup includes a data reliability verification mechanism for the data owner, which is not considered in the scheme Zhang22. The data owner in FCDedup needs to download additional auxiliary data and verify data reliability using these data, which causes additional time cost.

Bandwidth cost: We evaluate the bandwidth cost of our FCDedup from the perspectives of the endpoint device, fog node, and CSP, and Fig. 6 shows the results during the data uploading process. The duplication detection and transmission of ciphertexts and other auxiliary data cause bandwidth cost. Compared to the scheme Zhang22, our FCDedup achieves much lower bandwidth costs. This is due to the fact that our FCDedup supports both intra-deduplication and inter-deduplication, while the scheme Zhang22 only supports intra-deduplication. As a result, duplicated data blocks across different data owners do not need to be uploaded to the cloud in our FCDedup.

Storage cost: We evaluate the storage efficiency from the perspectives of the fog nodes and CSP. As shown in Fig. 7(b), FCDedup has a significantly lower cloud storage overhead than the scheme Zhang22. Because FCDedup supports inter-deduplication across different data owners and duplicate blocks across four data owners do not need to be stored. As shown in Figs. 7(a) and (b), both our FCDedup and the scheme Zhang22 require additional storage to achieve deduplication in both the fog nodes and CSP. Our FCDedup has a slightly higher storage cost for the additional data compared to the scheme Zhang22, because the CSP and fog nodes store some specific data to achieve data reliability verification, which is not considered in the scheme Zhang22. However, the additional storage is acceptable, since it only takes a small part of storage space compared to the storage cost of ciphertexts.

VI. RELATED WORK

Secure data deduplication aims to improve cloud storage efficiency while protecting data confidentiality through deduplication over encrypted data. Existing standard symmetric encryption algorithms cannot be used in secure deduplication, since they encrypt an identical file into different ciphertexts by users with different secret keys. To address this issue, Douceur et al. [41] first proposed the principle of convergent encryption

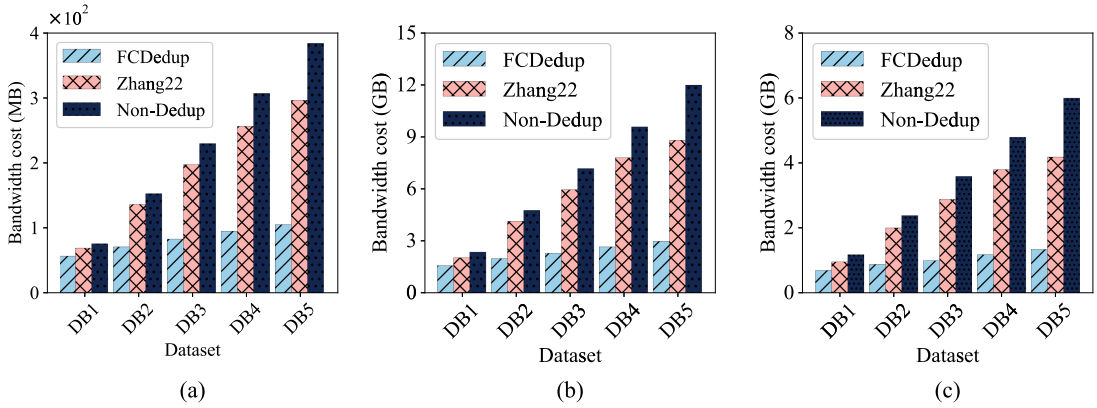


Fig. 6. Comparisons of bandwidth cost during the data uploading process. (a) The average bandwidth cost per endpoint device; (b) the average bandwidth cost per fog node; (c) the bandwidth cost of the cloud.

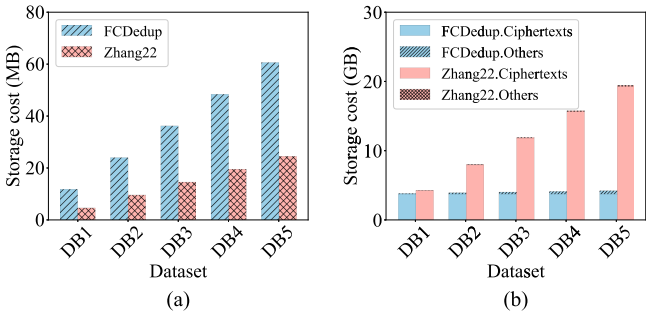


Fig. 7. Comparisons of storage cost. (a) The actual storage cost per fog node; (b) the actual storage cost of the cloud.

TABLE III
FUNCTIONALITY COMPARISONS

Scheme	Intra-deduplication within a single data owner	Inter-deduplication across different data owners	Data reliability verification
Zhang22 [33]	√	×	×
FCDedup	√	√	√

(CE), which derives the encryption key from the plaintext itself. Then different users can encrypt an identical file to the same ciphertexts for deduplication. After that, Bellare et al. [17] formalized CE as the MLE that has been widely used in many secure deduplication schemes [18], [19], [20], [42], [43], [44], [45], [46]. However, the deterministic MLE inherently suffers from offline BFA [21]. To improve the data confidentiality, Bellare et al. [21] proposed DupLESS, which uses a key server to participate in generating the encryption key. Then the cloud cannot launch offline BFA. Later, some subsequent schemes [23], [24], [25] use multiple key servers to mitigate the issues of a single point of failure and efficiency bottleneck in single key server system. Besides, the schemes in [2], [22] develop a new multi-domain deduplication architecture over encrypted data. The users under the same agent have the same secret parameters and an agent first detects duplicated encrypted data among users under the same domain. Then the CSP detects duplicated encrypted data across multiple agents.

Most previous secure deduplicaiton schemes are designed for traditional two-layer cloud storage and cannot be applied to the emerging fog-assisted cloud storage with a three-layer architecture. It is also almost impossible to deploy trusted key servers in fog computing since an endpoint device only communicates with the near fog node in practice. The schemes in [2], [22] perform secure deduplication on multiple domains and can be regarded as three-layer deduplicaiton schemes. However, they are also designed for traditional cloud storage and all the users in the same domain use a same secret parameter. In fog-assisted cloud storage, the endpoint devices under the same fog node may belong to different data owners and thus use different secret parameters. Thus, these previous schemes are inapplicable to fog-assisted cloud storage that has a different system architecture from traditional cloud storage.

Recently, Zhang et al. [33] proposed the first secure deduplication scheme for fog-assisted cloud storage. This scheme can achieve intra-deduplication within a single data owner at both the fog node and cloud levels. However, there exists considerable duplicated data across multiple data owners in practical applications [1] and this scheme cannot perform deduplication across different data owners. Table III summarizes the functionalities of our FCDedup and the scheme Zhang22 in [33]. Our FCDedup is the first secure deduplication scheme that supports both intra-deduplication and inter-deduplication while providing encrypted data with the ability to resist BFA. Besides, FCDedup is also designed to allow a data owner to verify the file content and file number of the files downloaded from the cloud and thus can guarantee the data reliability, which is not considered in the scheme [33].

VII. CONCLUSION

In this paper, we propose the FCDedup to enable encrypted data deduplication in fog-assisted cloud storage. A two-level duplication detection mechanism is designed to enable each fog node to detect duplicated encrypted data uploaded by the endpoint devices of the same data owner and also enable cloud server to detect duplicated encrypted data belonging to the same

or different data owners. Thus, FCDedup can achieve both intra-deduplication within a single data owner and inter-deduplication across different data owners. During the data storage, FCDedup can provide the encrypted data with a high ability to resist BFA launched by the fog node and CSP. Besides, a data verification mechanism is provided to allow a data owner to verify the file content and file number between the downloaded files with the files uploaded by endpoint devices, which ensures the reliability of the downloaded files. We theoretically prove the deduplication correctness and security of FCDedup. Besides, we implement a prototype of FCDedup using AlibabaCloud OSS as backend cloud storage and the comprehensive evaluation results demonstrate the efficiency of our design. Since it is common for the endpoint devices of a data owner to upload a large number of files to the cloud, it is time-consuming and inconvenient for the data owner to verify these files individually, Our future work aims to extend the functionality of our scheme by implementing batch verification capabilities for the data owner.

REFERENCES

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

[2] X. Yang, R. Lu, J. Shao, X. Tang, and A. A. Ghorbani, "Achieving efficient and privacy-preserving multi-domain Big Data deduplication in cloud," *IEEE Trans. Serv. Comput.*, vol. 14, no. 5, pp. 1292–1305, Sep./Oct. 2021.

[3] Y. Fu, N. Xiao, T. Chen, and J. Wang, "Fog-to-multicloud cooperative eHealth data management with application-aware secure deduplication," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 5, pp. 3136–3148, Sep./Oct. 2022, doi: 10.1109/TDSC.2021.3086089.

[4] "Medical data," 2020. [Online]. Available: https://academictorrents.com/collection/medical

[5] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Proc. IEEE Australas. Telecommun. Netw. Appl. Conf.*, 2014, pp. 117–122.

[6] D. Koo and J. Hur, "Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing," *Future Gener. Comput. Syst.*, vol. 78, pp. 739–752, 2018.

[7] C. Zhu et al., "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019.

[8] T. N. Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fog computing in healthcare Internet of Things: A case study on ECG feature extraction," in *Proc. IEEE Int. Conf. Comput. Inf. Technol.; Ubiquitous Comput. Commun.; Dependable, Autonomic Secure Comput.; Pervasive Intell. Comput.*, 2015, pp. 356–363.

[9] H. Dubey, J. Yang, N. Constant, A. M. Amiri, Q. Yang, and K. Makodiya, "Fog data: Enhancing telehealth Big Data through fog computing," in *Proc. ASE Big Data Soc. Inform.*, 2015, pp. 1–6.

[10] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. IEEE Federated Conf. Comput. Sci. Inf. Syst.*, 2014, pp. 1–8.

[11] S. Jiang, J. Liu, Y. Zhou, and Y. Fang, "FVC-Dedup: A secure report deduplication scheme in a fog-assisted vehicular crowdsensing system," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2727–2740, Jul./Aug. 2022, doi: 10.1109/TDSC.2021.3069944.

[12] A. Yang, J. Weng, K. Yang, C. Huang, and X. Shen, "Delegating authentication to edge: A decentralized authentication architecture for vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1284–1298, Feb. 2022.

[13] J.-S. Fu, Y. Liu, H.-C. Chao, B. K. Bhargava, and Z.-J. Zhang, "Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4519–4528, Oct. 2018.

[14] S. He, B. Cheng, H. Wang, X. Xiao, Y. Cao, and J. Chen, "Data security storage model for fog computing in large-scale IoT application," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2018, pp. 39–44.

[15] T. Wang, J. Zhou, X. Chen, G. Wang, A. Liu, and Y. Liu, "A three-layer privacy preserving cloud storage scheme based on computational intelligence in fog computing," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 3–12, Feb. 2018.

[16] T. S. Nikoui, A. M. Rahmani, and H. Tabarsaied, "Data management in fog computing," in *Fog and Edge Computing: Principles and Paradigms*, Hoboken, NJ, USA: Wiley, 2019, pp. 171–190.

[17] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 2013, pp. 296–312.

[18] 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 Secur.*, vol. 12, no. 3, pp. 532–543, Mar. 2017.

[19] S. Jiang, T. Jiang, and L. Wang, "Secure and efficient cloud data deduplication with ownership management," *IEEE Trans. Serv. Comput.*, vol. 13, no. 6, pp. 1152–1165, Nov./Dec. 2020.

[20] L. Liu, Y. Zhang, and X. Li, "KeyD: Secure key-deduplication with identity-based broadcast encryption," *IEEE Trans. Cloud Comput.*, vol. 9, no. 2, pp. 670–681, Second Quarter 2021.

[21] S. Keelveedhi, M. Bellare, and T. Ristenpart, "DupLESS: Server-aided encryption for deduplicated storage," in *Proc. 22nd USENIX Secur. Symp.*, 2013, pp. 179–194.

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

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

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

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

[26] G. Anthoine et al., "Dynamic proofs of retrievability with low server storage," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 537–554.

[27] "Alicloud," 2019. [Online]. Available: https://www.aliyun.com/product/oss

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

[29] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography Conference*. Springer, 2007, pp. 535–554.

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

[31] F. Bao, R. H. Deng, and H. Zhu, "Variations of Diffie-Hellman problem," in *Proc. Int. Conf. Inf. Commun. Secur.*, Springer, 2003, pp. 301–312.

[32] J. Liu, N. Asokan, and B. Pinkas, "Secure deduplication of encrypted data without additional independent servers," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 874–885.

[33] C. Zhang, Y. Miao, Q. Xie, Y. Guo, H. Du, and X. Jia, "Privacy-preserving deduplication of sensor compressed data in distributed fog computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4176–4191, Dec. 2022, doi: 10.1109/TPDS.2022.3179992.

[34] "OpenSSL," 2019. [Online]. Available: https://www/openssl.org/

[35] T. Granlund, "GNU multiple precision arithmetic library," 2010. [Online]. Available: http://gmplib.org/

[36] "PBC library," 2010. [Online]. Available: https://crypto.stanford.edu/pbc/

[37] "Dropbox," 2007. [Online]. Available: https://www.dropbox.com

[38] "Google drive," 2014. [Online]. Available: http://drive.google.com

[39] "Mysql," 2008. [Online]. Available: https://dev.mysql.com

[40] "LCTSC dataset," 2017. [Online]. Available: https://academictorrents.com/details/0a3611528c9172383656cb1b6a07cfb7f095eb82

[41] 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. IEEE 22nd Int. Conf. Distrib. Comput. Syst.*, 2002, pp. 617–624.

[42] G. Tian et al., "Blockchain-based secure deduplication and shared auditing in decentralized storage," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 3941–3954, Nov./Dec. 2022, doi: 10.1109/TDSC.2021.3114160.

[43] 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 Conf. Comput. Commun.*, 2017, pp. 1–9.

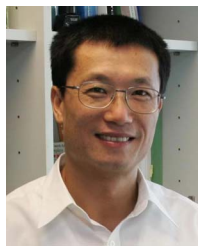
[44] J. Li, Y. K. Li, X. Chen, P. P. Lee, and W. Lou, "A hybrid cloud approach for secure authorized deduplication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1206–1216, May 2015.
[45] H. Yuan, X. Chen, J. Wang, J. Yuan, H. Yan, and W. Susilo, "Blockchain-based public auditing and secure deduplication with fair arbitration," *Inf. Sci.*, vol. 541, pp. 409–425, 2020.
[46] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2386–2396, Aug. 2016.

Mingyang Song received the BE and ME degrees in software engineering from Sun Yat-sen University, Guangzhou, China, in 2019 and 2021, respectively. He is currently working toward the EngD 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 BS degree in software engineering from Chongqing University, Chongqing, China, in 2011, and the MS and PhD 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 Multimedia*, and *ACM Multimedia*. He has been recognized as a 'Highly Cited researcher 2022'. His current research interests are focused on chaotic system, multimedia security, and secure cloud computing. He has published about seventy papers on the subject, receiving more than 5300 citations.

Yifeng Zheng (Member, IEEE) received the PhD degree in computer science from the City University of Hong Kong, Hong Kong, in 2019. He is an assistant professor with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. He worked as a postdoc with the Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia and City University of Hong Kong. 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*. He received the Best Paper Award in the European Symposium on Research in Computer Security (ESORICS) 2021. His current research interests are focused on security and privacy related to cloud computing, IoT, machine learning, and multimedia.

Tao Xiang (Senior Member, IEEE) received the BEng, MS and PhD 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. His research interests include multimedia security, cloud security, data privacy and cryptography. He has published more than 150 papers on international journals and conferences. He also served as a referee for numerous international journals and conferences.



Xiaohua Jia (Fellow, IEEE) received the BSc and MEng degrees from the University of Science and Technology of China, in 1984 and 1987, respectively, and the DSc 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 in 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 the general chair of ACM MobiHoc 2008, area-chair of IEEE INFOCOM 2010, TPC co-chair of IEEE GlobeCom 2010-Ad Hoc and Sensor Networking Symposium, and Panel co-chair of IEEE INFOCOM 2011. He is an editor of the *Journal of World Wide Web, IEEE Transactions on Computers, etc.*