BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2 (2022) 100062

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: https://www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

# An efficient encrypted deduplication scheme with security-enhanced proof of ownership in edge computing

Yukun Zhou [a,b,c], Zhibin Yu [a,*], Liang Gu [b], Dan Feng [c]

[a] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[b] Sangfor Technologies Inc, Shenzhen, China
[c] Wuhan National Laboratory for Optoelectronics, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China

ARTICLE INFO

*Keywords:*
Deduplication
Message-locked encryption
Proof of ownership
Edge computing

ABSTRACT

With the rapid expansion of Internet of Things (IoT), relevant files are stored and transmitted at the network edge by employing data deduplication to eliminate redundant data for the best accessibility. Although deduplication improves storage and network efficiency, it decreases security strength and performance. Existing schemes usually adopt message-locked encryption (MLE) to encrypt data, which is vulnerable to brute-force attacks. Meanwhile, these schemes utilize proof-of-ownership (PoW) to prevent duplicate-faking attacks, while they suffer from replay attacks or incur large computation overheads. This paper proposes SE-PoW, an efficient and location-aware hybrid encrypted deduplication scheme with a dual-level security-enhanced Proof-of-Ownership in edge computing. Specifically, SE-PoW firstly encrypts files with an inter-edge server-aided randomized convergent encryption (RCE) method and then protects blocks with an intra-edge edge-aided MLE method to balance security and system efficiency. To resist duplicate-faking attacks and replay attacks, SE-PoW performs the dual-level PoW algorithm. Then it combines the verification of a cuckoo filter and the homomorphism of algebraic signatures in sequence to enhance security and improve ownership checking efficiency. Security analysis demonstrates that SE-PoW ensures data security and resists the mentioned attacks. Evaluation results show that SE-PoW reduces up to 61.9% upload time overheads compared with the state-of-the-art schemes.

## 1. Introduction

With the high-speed development of 5G and edge computing, large amounts of data are collected in the core and edge devices, such as smartphones, wearables, automatic driving [1], and traffic flow detection [2]. In the big data era, IDC predicts that the digital universe will reach 175ZB in 2025 [3], and more than 44% of IoT-created data will be processed and analyzed at the network edge. Edge computing deploys computing and storage resources at the network edge to handle time-sensitive tasks while offering fast and convenient services to users [4]. Research analysis shows that there exist large amounts of redundant data (up to 76%) for workloads like VM images and car multimedia IoT data [5–7]. Data deduplication has been adopted in the modern central cloud (e.g., Dropbox [8], OneDrive [9]) and also pushed to the network edge for both optimized space and network efficiency. Fig. 1 describes a simple architecture of edge computing that deploys deduplication, for example, Ctera [10]. Edge computing can be seen as a three-tiered architecture. The central cloud stores and retrievals data from edge nodes and users. The edge nodes provide limited computing, indexing, storage, and other services [11,12]. Deduplication eliminates duplicate data on a file or block, which keeps only one physical copy and others refer to it. Deduplication can be classified into client-side or server-side approaches, while the former also saves network transmission. Edge computing deployed with deduplication has attracted lots of attention in both academia and industry [10–13], but it remains many security issues and potential threats [14,15].

Users usually encrypt data before outsourcing them to the edge and cloud for security and privacy concerns. However, encrypting the same data with different keys will result in different ciphertexts and makes deduplication impossible. Many researchers propose convergent encryption(CE) and message-locked encryption(MLE) [16–21] that adopt the hash value as the symmetric key to encrypt data, which users carry out deduplication over ciphertexts. Unfortunately, MLE suffers from resist brute-force attacks [18] that the attacker can recover the target file from a known set by offline encryption. To mitigate the attacks, researchers propose an oblivious pseudorandom function(OPRF) to generate MLE keys aided by secret messages of the server. However, client-side deduplication suffers from various attacks and privacy leakages, such as duplicate-faking attacks [15,22,23] and position attacks. That is, a malicious user can gain access to files belonging

* Corresponding author.
*E-mail address:* zb.yu@siat.ac.cn (Z. Yu).

https://doi.org/10.1016/j.tbench.2022.100062
Received 16 April 2022; Received in revised form 9 May 2022; Accepted 9 May 2022
Available online 24 May 2022

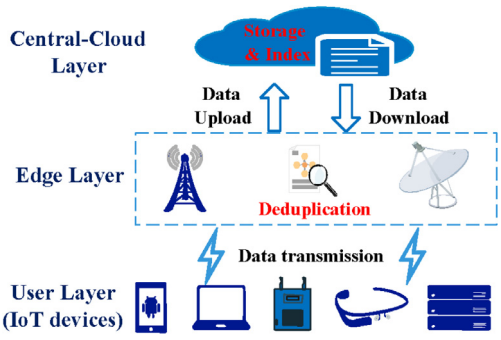



**Fig. 1.** An example of edge computing deployed with data deduplication.

to other users based on a hash value or upload corrupted data with valid hash values [24]. Some deduplication with Proof of ownership (PoW) schemes [22,25–27] are proposed to verify ownership of data users, such as MHT-PoW [22] or BF-PoW [21,26]. The user convinces the server that it owns the hash value and holds the file content. PoW is a protocol in which a server sends challenges, and the client returns the proofs as a response. Specifically, MHT-PoW encodes files into a fixed-size buffer and conducts a Merkle hash tree via a pairwise independent hash function [20,28]. BF-PoW divides files into fixed-size blocks, calculates the hash digests, and inserts them into a bloom filter [29]. However, existing encrypted deduplication schemes with PoW face new challenges.

First, existing schemes adopt MLE [17,30] or RCE [20,21] to protect data security, but they are vulnerable to brute-force attacks for the low-entropy files especially. Moreover, other encrypted deduplication schemes, such as OPRF [18], data re-encryption [20,21], and public encryption [31] bring a significant computational burden. They are not suitable for resource-constrained edge nodes and IoT devices.

Second, existing schemes introduce proof-of-ownership to resist duplicate-faking attacks. They nevertheless incur large computation overheads or suffer from replay attacks. Generally speaking, MHT-PoW brings a heavy computation burden because of the encoding of files and constructions of the Merkle hash tree. BF-PoW suffers from replay attacks and the privacy leakage of the false positive in a bloom filter. An attacker passes the verification of BF-PoW by generating valid proof from previous proofs without owning the original data. The replay attacks also have occurred in many scenes, such as provable data possession(PDP) and proof of retrievability (PoRs) [32].

To overcome these challenges, we propose an efficient encrypted deduplication with Security-Enhanced Proof-of-Ownership (SE-PoW). We observe that the capabilities and security risks for inter-/intra edge nodes are different [4], and duplicate files are mainly from multiple users [33,34]. The core idea behind SE-PoW is to employ different randomized MLE methods based on the location of deduplication. Specifically, SE-PoW first performs inter-edge encrypted deduplication for files via a server-aided RCE method. If the file is non-duplicate, SE-PoW further performs intra-edge encrypted deduplication for blocks via an edge-aided MLE method. Moreover, SE-PoW utilizes a dual-level proof-of-ownership to guarantee higher security. SE-PoW performs ownership checking based on a cuckoo filter to resist duplicate-faking attacks. SE-PoW adds unique labels and verifies the homomorphism of the algebraic signature [35] to resist replay attacks. Security analysis demonstrates that SE-PoW resists the above attacks from inside and outside attackers. Therefore, SE-PoW significantly reduces computation overheads compared with state-of-the-art schemes and ensures data security.

This paper makes the following contributions.

- We propose SE-PoW, a location-aware hybrid encrypted deduplication scheme in edge computing. SE-PoW performs inter-edge file-level and intra-edge block-level encrypted deduplication via server-aided RCE and edge-aided MLE algorithms, respectively. Thus SE-PoW balances data confidentiality and efficiency.
- SE-PoW proposes a dual-level security-enhanced proof-of-ownership by leveraging a cuckoo filter and algebraic signatures. SE-PoW achieves a higher security level and only increases little overheads, in which SE-PoW resists duplicate-faking attacks and replay attacks.
- We present a prototype of SE-PoW. Security analysis demonstrates that SE-PoW can ensure data confidentiality and resist duplicate-faking attacks and replay attacks under the proposed threat model. Experimental results based on real-world datasets show that SE-PoW reduces 21.9–61.9% upload time overheads compared with the state-of-the-art MHT-PoW.

The reset of our paper is organized as follows. Section 2 introduces the background and problems of SE-PoW in edge computing. In Section 3 the system model, threat model and security requirements are defined. Section 4 introduces the design and implementation details of SE-PoW. Section 5 discusses the security of SE-PoW. Section 6 presents the performance evaluation on real-world datasets. In Section 7, the related works on encrypted deduplication schemes are reviewed. Finally, Section 8 concludes this paper.

## 2. Background & problems

This section briefly introduces encrypted deduplication in edge computing and proof of ownership schemes. We further present the problems and motivation of SE-PoW.

### 2.1. Encrypted deduplication in edge computing

Many users store files at the network edge and respond to users' requests with low latency [4,34]. In Fig. 1, edge computing employs data deduplication at the network edge for space and network efficiency [11–13]. The user uploads/retrieves data and relevant information to the edge nodes. Then edge nodes could compute the tags of data via a hash function (i.e., SHA256) and encrypt data. Edge nodes maintain a deduplication index structure for local or cross-domain duplicate checking. Decentralized deduplication distributes data to multiple edge nodes for load balancing [13].

To protect data confidentiality, Douceur et al. [16] propose convergent encryption(CE) and Bellare et al. [17] propose Message-locked Encryption(MLE) and random to enable deduplication over ciphertexts. Specifically, the client derives a key $K \leftarrow H(P, M)$ from message $M$, and $P$ is a public parameter and $H$ is a cryptographic hash function. And it encrypts the message as $C \leftarrow \text{Encry}(K, M)$, where $Encry/Decry$ is a pair of encryption and decryption functions. The tag $T$ derives $T \leftarrow H(P, C)$. In randomized convergent encryption(RCE), the client encrypts a message $C_1 \leftarrow Encry(L, M)$, where $L$ is a randomly chosen key. Then it encrypts the key $L$ and generate $C_2 \leftarrow L \oplus K$, where $K$ is derived from the message $K \leftarrow H(P, M)$. The client generates tag $T \leftarrow H(P, K)$. When any owner receives $C_1 \| C_2 \| T$ from the server, he computes $L \leftarrow C_2 \oplus K$, and obtains $M$ via $M \leftarrow Decry(L, C_1)$. However, MLE and RCE are vulnerable to brute-force attacks.

Bellare et al. [18] present server-aided MLE algorithms via an RSA-based oblivious PRF protocol to resist brute-force attacks. In edge computing, Ni et al. [36] put forward edge-based encrypted deduplication with BLS-OPRF and adopted proxy re-encryption on edge nodes. Yang et al. [15] propose a cross-domain deduplication scheme with server-aided MLE via HPS-OPRF in blockchain-enabled edge computing. In addition, Hur et al. [20] propose authorized encrypted deduplication with dynamic ownership management via proxy re-encryption [21]. In summary, encrypted deduplication has been widely used in edge computing.

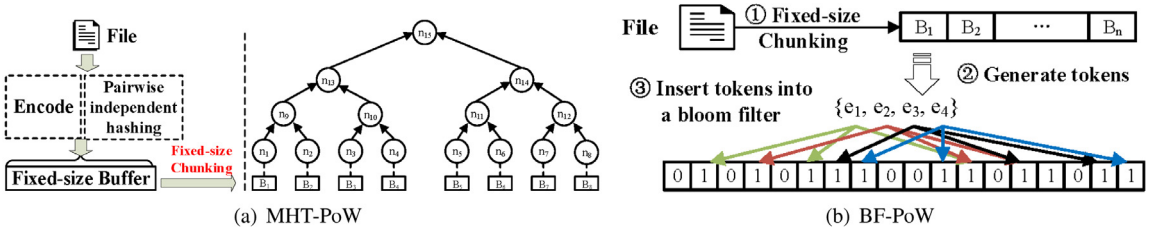



**Fig. 2.** We describe the procedure of MHT-PoW and BF-PoW. In figure (a), MHT-PoW encodes a file into a fixed-size buffer and constructs a Merkle hash tree. In figure (b), BF-PoW splits a file into blocks, generates tokens (e.g., $e_i$), and inserts them into a bloom filter. Finally, the verifier randomly selects $N$ challenged blocks for ownership checking.

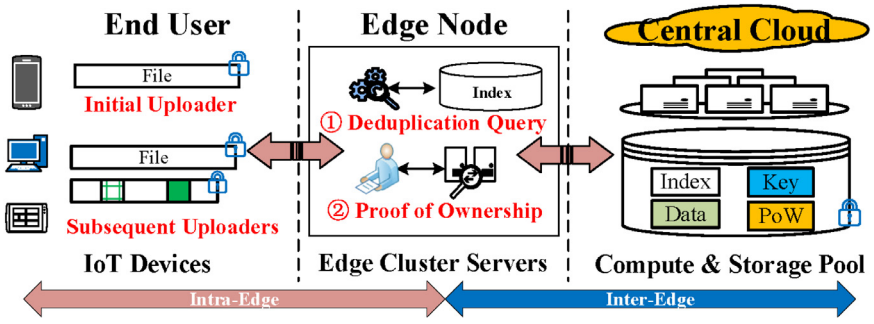



**Fig. 3.** The system model of SE-PoW in edge computing.

### 2.2. PoW Schemes & problems

Client-side encrypted deduplication schemes occur from the entities(i.e., IoT devices) and diminish bandwidth consumption significantly. However, the risks of privacy leakage arise in existing schemes, for example, duplicate-faking attacks [15,22]. In particular, an attacker uses a hash value to gain unauthorized access and download files in Dropbox [23]. Researchers propose proof-of-ownership to tackle the problem, which checks ownership and achieves authorized access. Existing schemes are classified into two categories: Merkle Hash Tree based PoW (MHT-PoW) [15,22,25] and Bloom Filter based PoW (BF-PoW) [21,37] in Fig. 2.

**MHT-PoW.** Halevi et al. [22] propose MHT-PoW to resist duplicate-faking attacks. In Fig. 2(a), the client and server simultaneously encode the file into a buffer via erasure coding and the pairwise independent hash function. The buffer is divided into fixed-size blocks as $B_i$ ($0 < i < n$), and computes the hash value $n_i$ for each data block $B_i$ as the leaf node. And the parent node is to calculate the hash value of the two child nodes. Finally, they get the root node $n_{15}$. During the verification of MHT-PoW, the server randomly selected $N$ leaf node indexes as the challenge information. The client returns the path information from the leaf node to the root node, and the server finally recalculates and compares the value of the root node. Similarly, ECC-based accumulators are adopted in [15]. Unfortunately, the encoding of files and the construction of structures in MHT-PoW will bring great computational and I/O overhead.

**BF-PoW.** To reduce the computation and I/O overheads, BF-PoW [21,26,30] uses a bloom filter to resist duplicate-faking attacks with a low error rate. In Fig. 2(b), BF-PoW divides the file into blocks and calculates the token $e_i$ ($1 \le i < 5$) of the corresponding block with a pseudo-random function, and inserts it into the bloom filter. For example, the server selects data blocks 1 and 3 as challenge blocks. The client calculates tokens $e_1$ and $e_3$ and queries whether they exist in the bloom filter to check the ownership. When the false positive occurs in a bloom filter or the attackers utilize the previous valid proofs, BF-PoW leads to privacy leakage.

According to our analysis, existing schemes face security and performance challenges. First, encrypted deduplication schemes suffer from brute-force attacks or a heavy computational burden. Second, MHT-PoW incurs extensive time and I/O overheads. BF-PoW is subjected to replay attacks. We analyze the redundant distribution and architectural features in edge computing to solve these problems. From previous work in [4,33,34,38], more than 90.5%–99% redundant data remains in cross-domain duplicate files and duplicate blocks within users. In edge computing, performing deduplication at edge nodes is highly efficient and prevents privacy risks and information leakage. Meanwhile, client-side deduplication between edge nodes and the central saves network bandwidth and achieves security guarantees [14]. These motivate us to propose SE-PoW, a hybrid encrypted deduplication scheme for intra- and inter-edge with proof-of-ownership to achieve higher security in edge computing.

## 3. System model & threat model

This section firstly describes the system model and threat model of SE-PoW. Next, the security requirements and design goals of SE-PoW are listed as follows.

### 3.1. System model

Fig. 3 describes our system model that consists of three entities: Central Cloud(CC), Edge Node(EN), and End User (EU). The CC cannot offer high-quality services for large-scale data in a restricted network environment. Edge nodes locate on the user side and provide computing and storage services with limited resources. In the cloud offloading applications, deduplication will be done at edge nodes and the central cloud to save storage space and network bandwidth.

- **Central Cloud(CC).** The CC provides centralized storage /retrieval services. When a user is connected to the CC, CC will verify his password and credential. CC maintains file-level indices for inter-edge data deduplication. CC also stores ciphertexts of blocks, keys, information of PoW, and metadata. It assigns tasks to multiple edge nodes to handle a large amount of data.
- **Edge Node(EN).** The EN is an entity located at the network edge, which provides computing and storage services with limited resources. EN connects to CC via inter-network (e.g., Wide Area Network(WAN)) but communicates with users in a restricted domain (via intra-network). EN acts as a proxy between CC and the

user, supporting duplicate checking, encryption, and challenge-and-response of PoW. A trusted EN assists the user in generating random keys.

- **End User(EU).** The EU is a client or outsourcing entity (e.g., Mobile and IoT devices) consisting of initial and subsequent uploaders. EU uploads data to and retrievals data from the CC through the EN. The EU connects to edge nodes via intra-network (e.g., Local Area Network(LAN)). Moreover, EN can generate keys and encrypt/decrypt data with limited computation and storage resources. The initial uploader transmits data to CC and initializes the PoW. The subsequent uploaders with duplicate files need to verify the PoW protocol.

### 3.2. Threat model

We assume that the CC is "honest-but-curious" in edge computing. The CC will not maliciously delete or modify users' data, but the CC tries to learn the sensitive information as much as possible, such as data, keys, tags(i.e., hash value), and proofs PoW. Without loss of generality, we assume that the malicious CC may collude with other adversaries. The EN will perform our proposed protocol honestly. We assume that the EN is hard to be compromised in the intra-network [14] and is protected by firewalls and access control systems. A trusted EN helps users to generate secure keys. In our threat model, the adversaries can be classified into two types: outside adversaries and inside adversaries.

- Outside adversaries may be malicious users or hackers. They obtain some sensitive data (e.g., a hash value, proofs of PoW) via a public network, such as a web crawler and artificial intelligence. Outside adversaries aim to get target users' sensitive data content and keys from CC and EN. They may disguise themselves as a legitimate user to interact with the CC or EN.
- Inside adversaries follow the prescribed protocols but try to obtain users' information, such as plaintexts of data, tags, and proofs of a specific file. The inside adversaries try to cheat the EN and CC by using previous proofs and make the verification of PoW successful.

### 3.3. Security requirements & design goals

We aim to achieve the following security requirements and design goals based on the above threat model.

- **Data confidentiality:** We require that the encrypted data and keys will be achieved semantically secure and resist brute-force attacks [17].
- **Tag consistency:** The deduplication scheme should allow the users to verify data integrity. It can resist poison attacks, in which a malicious attacker cannot upload a valid hash value but replaces a file with a poisoned one.
- **Backward privacy:** When a user uploads a duplicate file that exists in the CC, CC will check the ownership. Unauthorized data owners who cannot pass the verification of the PoW would not access files.
- **Resistance to duplicate-faking attacks:** An attacker who only has the data tag cannot download the corresponding ciphertexts of files.
- **Resistance to replay attacks:** An attacker cannot pass the verification of PoW, even if it generates valid proofs from the previous message without owning files.

**Design goals.** Our scheme should achieve the following design goals. First, SE-PoW should meet the mentioned security requirements. SE-PoW also realizes upload and download protocols using encrypted deduplication, key generation, and proof-of-ownership among the EU, EN, and CC. Second, SE-PoW ensures system efficiency, which reduces the cost of computation, transmission, and storage. At last, other problems, such as data reliability [39], updating, and ownership management, are beyond the scope of this paper.

**Table 1**
Notations used in the proposed scheme.

| Notation | Description |
|---|---|
| $u_i$ | An end user |
| $ID_{u_i}$ | The identity of $u_i$ |
| $F_i$ | A file |
| $B_i$ | A block |
| $n$ | Number of blocks |
| $C$ | Ciphertext of a block/key |
| $OList_{F_i}$ | An owner list of $F_i$ |
| $K_{u_i}/K_{F_i}/K_{B_i}$ | A user/file/block key |
| $CF_PoW[F_i]$ | A cuckoo filter based PoW |
| $Sig_g(B_i)$ | An algebraic signature |
| $V_i$ | A tag to resist replay attacks |

### 3.4. Preliminaries

Before introducing the design of SE-PoW, we describe two data structures: cuckoo filter and algebraic signature.

**Cuckoo Filter.** A cuckoo filter [40] is a data structure that is used to provide approximate set membership tests whether a given item is in a set or not. It is similar to Bloom filter [29]. A cuckoo filter is a compact variant of a cuckoo hash table that stores only fingerprints instead of key–value pairs. A set membership query for item $x$ searches the hash table for the fingerprint of $x$ and returns true if an identical fingerprint is found. A cuckoo filter can show false positives but not false negatives. It supports adding and removing items dynamically. It provides a higher lookup performance than Bloom filters. The cuckoo filter has various advantages over the Bloom filter. (1) It takes less time for lookups. (2) It has fewer false positives than the bloom filter for the same number of items stored. (3) It supports the deletion of items.

**Algebraic Signature.** Algebraic signature [35,41] is a hash function with homomorphic and algebraic properties. Algebraic signature has been widely used in remote data possession checking in distributed system [35] and cloud storage [42]. An algebraic signature consists of $n$ symbols to verify the uniqueness of data content. The basic feature of the algebraic signature method is that the sum of the algebraic signature of data blocks is equal to the signature result of the corresponding sum of data blocks. Concretely speaking, let $\lambda$ be a tuple in Galois Field, which $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_n)$ is a vector of distinct non-zero elements. The file $F$ is divided into $n$ blocks $f[1], f[2], \cdots, f[n]$, and the formula for calculating the algebraic signature of file $F$ is

$$S_\lambda(F) = \sum_{i=1}^{n} f[i] \cdot \lambda^{i-1} \tag{1}$$

The properties of an algebraic signature are as follows:

**Property 1.** *Concatenating two data blocks $f[i]$ and $f[j]$ of length $l$ and $m$, into a super block denoted $f[i] \parallel f[j]$. Then the signature $S_\lambda(f[i] \parallel f[j])$ is as follows.*

$$S_\lambda(f[i] \parallel f[j]) = S_\lambda(f[i]) + \lambda^l S_\lambda(f[j]) \tag{2}$$

**Property 2.** *The algebraic signature of the sum of all data blocks of file $F$ equals the sum of the algebraic signatures of each data block.*

$$\begin{aligned} &S_\lambda(f[1]) + S_\lambda(f[2]) + \cdots + S_\lambda(f[n]) \\ &\quad = S_\lambda(f[1] + f[2] + \cdots + f[n]) \end{aligned} \tag{3}$$

## 4. Design and implementation of SE-PoW

In this section, we first describe the overview of SE-PoW. Then we present the design and proof of ownership algorithms used in SE-PoW. Table 1 describes the notations.

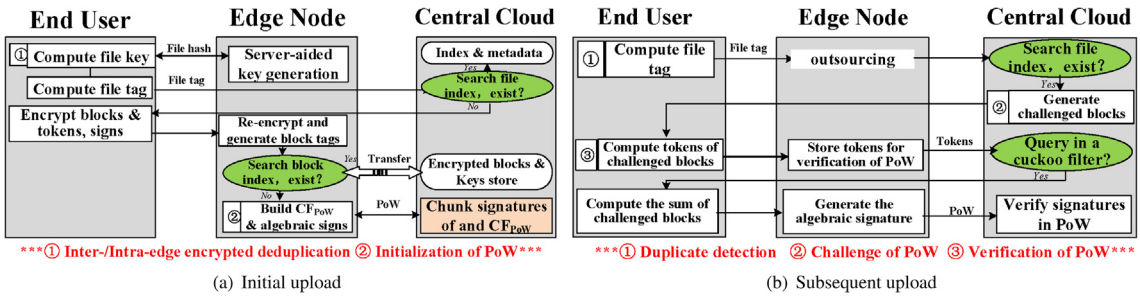



**Fig. 4.** The procedure of initial and subsequent data upload.

### 4.1. Overview of SE-PoW

We perform an inter-edge and intra-edge encrypted deduplication scheme for files and blocks in the upload phase. In Fig. 4, the EU is allowed to transfer files to the EN and CC and retrieves relevant files on demand. Data are encrypted via different MLE algorithms according to the location of deduplication to balance security and system efficiency. Specifically, the EU generates a file tag and encrypts the file before sending it to an EN. The EN outsources the file tag to CC for inter-edge(cross-domain) file-level deduplication via a server-aided RCE algorithm. If unique, EU encrypts blocks via an edge-aided MLE algorithm and initializes the tokens and algebraic signatures. EU outsources them to the EN for re-encryption and performs intra-edge block-level deduplication. Besides, the EN stores the data of SE-PoW based on a cuckoo filter and algebraic signatures for PoW verification. The EN transfers non-duplicated blocks and metadata to the CC and initializes a PoW protocol.

In the subsequent upload phase, the EU sends a tag of a duplicate file to the CC. Then the user performs a dual-level proof of ownership for duplicate files in edge computing to ensure data privacy. Concretely speaking, we first perform the challenge-and-response protocol over CF-PoW. If it passes, we will verify the homomorphism of algebraic signatures as the second-level PoW. Only verifying the ownership of SE-PoW, the end-user will send the file metadata without uploading data content.

### 4.2. Encrypted deduplication in SE-PoW

To resist brute-force attacks and minimize bandwidth overheads in edge computing, we proposed a location-aware hybrid encrypted deduplication in SE-PoW. SE-PoW combines server-aided RCE for inter-edge files and edge-aided MLE for intra-edge blocks. The encryption methods, such as MLE [17], RCE [17], and RSA-OPRF [18], are adopted from previous works. Details are shown as follows.

***System setup.*** We choose two hash functions $H_1$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$. And the uploader adopts the AES APIs [43], such as Encry() and Decry(). The public parameters $e$ and $N$ of RSA are initialized and $d$ is generated via $e \cdot d \equiv 1 mod \phi(N)$. Each edge node will initialize a master key $K_{e_i}$.

***Data Upload.*** As show in Fig. 4(a), an user $u_i$ uploads a file $F_i$ to the central cloud. The CC will verify the identity $ID_{u_i}$ and password. The following details are the file/block-level encrypted deduplication.

(1) $u_i$ computes the tag $T_{F_i} \leftarrow H_1(H_1(F_i))$. Then $u_i$ generates a server-aided RCE key $K_{F_i}$ via oblivious pseudorandom protocol [18]. Specifically, for $F_i$, $u_i$ chooses a random number $r \in N$, and sends $x = H_1(F_i) \cdot r^e \ mod \ N$ to a trusted edge node. The trusted edge node computes $y = x^d \ mod \ N$ and sends $y$ back. $u_i$ calculates $z = y \cdot r^{-1} \ mod \ N$. $u_i$ could verify whether or not $H_1(F_i) \equiv z^e \ mod \ N$. Thus, $u_i$ chooses a random key via $L_{F_i} \leftarrow \{0,1\}^{k(\lambda)}$, and denotes as $K_{F_i} = (L_{F_i}, z)$.

(2) $u_i$ sends $T_{F_i}$ to the edge nodes(EN) and forwards it to the central cloud(CC) for inter-edge file-level deduplication. The CC will check whether $T_{F_i}$ exists in the inter-edge global file index. If no, $u_i$ performs block-level deduplication and jumps to (3). Otherwise, the CC will check the ownership, and details are in Section 4.3.

(3) $u_i$ performs intra-edge block-level encrypted deduplication via an edge-aided MLE. In particular, $u_i$ divides $F_i$ into $n$ blocks via $\{B_i\} \leftarrow$ Chunking $(F_i)$. For a block $B_i$ $(0 \le i < n)$, $u_i$ generates a MLE key $K_{B_i}$ via $K_{B_i} \leftarrow H_1(B_i)$. Then $u_i$ encrypts the block into ciphertexts $C_i || C_i^2 || C_i^3$ via $C_i \leftarrow Encry(K_{B_i}, B_i)$ and $C_i^2 \leftarrow Encry(L_{F_i}, K_{B_i})$, and $C_i^3 \leftarrow z \oplus L_{F_i}$.

(4) $u_i$ transmits all block ciphertexts $(C_i, C_i^2, C_i^3)$ to the EN. Then EN re-encrypts $C_i$ via $C_i^1 \leftarrow$ Encry$(K_{e_i}, C_i)$. The tag of block $B_i$ is generated via $T_{B_i} \leftarrow H_1(C_i^1)$. The EN performs intra-edge block-level encrypted deduplication by checking $T_{B_i}$ in the local block-level index. Then, the EN will upload all the ciphertext of non-duplicated blocks $C_i^1 \ || \ C_i^2 \ || \ C_i^3$ and metadata information $T_{B_i} \ || \ ID_{u_i}$ to the central cloud. The CC receives and stores ciphertexts and metadata. Then the central cloud adds the $ID_{u_i}$ to the owner list $OList_{F_i}$.

(5) The EN has to generate tokens and algebraic signatures to initialize a dual-level PoW protocol. Details are present in the initialization of SE-PoW in Section 4.3.

***Subsequent Upload.*** In Fig. 4(b), an subsequent uploader $u_j$ sends the file $F_i$ to the edge nodes and central cloud. First, $u_j$ generates the file tag $T_{F_i}$ as described in the upload phase. $u_j$ outsources them to the central cloud. The central cloud finds that $T_{F_i}$ exists in the file index via duplicate checking. Second, the central cloud performs a challenge-and-response phase to verify the ownership of $u_i$. (1) The central cloud randomly generates $c$ challenged blocks and returns the position of blocks. (2) The edge node computes tokens and algebraic signatures of challenged blocks based on the position of blocks and transfers them to the central cloud. (3) The central cloud firstly checks the tokens whether or not they exist in the cuckoo filter. If it passes, the CC will verify the homomorphism of algebraic signatures. Otherwise, the central cloud returns failed results. Details are present in the Algorithm 2. Third, if $u_j$ passes the verification of PoW, his identify $ID_{u_j}$ will be added to the owner list $OList_{F_i}$. And the CC returns results to the EN and $u_j$.

***Data Download.*** If a user $u_i$ wants to download a file $F_i$, $u_i$ will firstly send the identity $ID_{u_i}$ and file tag $T_{F_i}$ to the edge nodes and central cloud. The central cloud will firstly verify his identity $ID_{u_i}$ whether or not in the owner list. If no, the download request will be rejected. Otherwise, the CC will read the metadata of $F_i$ to return the ciphertexts of all blocks and keys $C_i^1 \ || \ C_i^2 \ || \ C_i^3$ $(0 \le i < n)$ to the EN. After receiving the ciphertexts, EN decrypts $C_i^1$ with $K_{e_i}$ via $C_i \leftarrow$ Decry$(K_{e_i}, C_i^1)$ And the EN forwards $C_i \ || \ C_i^2 \ || \ C_i^3$ $(0 \le i < n)$ to $u_i$. Next, $u_i$ decrypts the block key via $L_{F_i} \leftarrow C_i^3 \oplus z$ and $K_{B_i} \leftarrow$ Decry$(L_{F_i}, C_i^2)$. Thus, $u_i$ decrypts the ciphertext of block to get $B_i$ via $B_i \leftarrow$ Decry$(K_{B_i}, C_i)$. At last, $u_i$ creates a new file $F_i$ and writes each block $B_i (0 \le i < n)$ sequentially to recover the file $F_i$.

**Algorithm 1** The initialization of SE-PoW

**Input:** File $F_i$ & parameter $m$ of CF.
**Output:** $CF_{PoW}[F_i]$ & $Sig_g(B_i)$.
1: $u_i$ divides $F_i$ into blocks $B_i(0 \le i < n)$.
2: $u_i$ generates tokens $t_{B_i} \leftarrow H_2(B_i)$.
3: $u_i$ generates algebraic signatures of blocks $Sig_g(B_i) \leftarrow S_\lambda(B_i||ID_{F_i}||i)$.
4: $u_i$ generates $V_i \leftarrow S_\lambda(ID_{F_i}||i)$
5: $u_i$ outsources $t_{B_i}||Sig_g(B_i)||V_i$ to the EN and CC.
6: CC initializes $CF_{PoW}$=InitCF($m$) and PRF.
7: **for** $i = 0 \rightarrow n-1$ **do**
8: $e_i \leftarrow$ PRF($t_{B_i}$, $i$)
9: $CF_{PoW}[F_i] \leftarrow$ AddCF($e_i$)
10: **end for**
11: CC stores all $Sig_g(B_i)||V_i$ and $CF_{PoW}[F_i]$.

### 4.3. Proof of ownership in SE-PoW

We propose a dual-level PoW algorithm over encrypted deduplication with a cuckoo filter and algebraic signatures to resist duplicate-faking attacks and replay attacks. The cuckoo filter [40] and algebraic signature have been used in storage systems [41]. SE-PoW is a challenge-and-response protocol between two entities on a file $F$: $\Pi = (P, V)$. $P$ is the end-user and the edge node, and $V$ indicates the central cloud. In addition, protocol $\Pi$ consists of three phases: Initialization(Data upload), challenge, and verification (Subsequent upload). The details are present in Algorithm 1 and 2.

***Initialization of SE-PoW.*** An initial uploader $u_i$ generates tokens and algebraic signatures of each block in file $F_i$ for the ownership verification. As shown in algorithm 1, $u_i$ firstly divides $F_i$ into blocks $B_i(0 \le i < n)$. $u_i$ generates tokens via $t_{B_i} \leftarrow H_2(B_i)$ and algebraic signatures $Sig_g(B_i) \leftarrow S_\lambda(B_i \| ID_{F_i} \| i) \leftarrow \sum_{j=1}^{n}(B_{i,j} \| ID_{F_i} \| i) \cdot \lambda^{j-1}$. $ID_{F_i}$ is the identity of file $F_i$ and $i$ is the index of block $B_i$. $u_i$ also computes $V_i \leftarrow S_\lambda(ID_{F_i} \| i)$. Then SE-PoW resists replay attacks via unique labels $ID_{F_i} \| i$. $u_i$ sends tokens $t_{B_i}$ and algebraic signatures $Sig_g(B_i)$ to the EN. Next, the EN outsources tokens and signatures to the CC. Then the central cloud constructs a cuckoo filter $CF_{PoW}[F_i] \leftarrow$ InitCF($m$) with the parameter of total items $m$. For each token $t_{B_i}(0 \leqslant i < n)$, the CC computes $e_i \leftarrow$ PRF($t_{B_i}$, $i$) with a pseudorandom function PRF. The CC inserts all tokens into a cuckoo filter $CF_{PoW}[F_i] \leftarrow AddCF(e_i)$. Finally, the CC stores all the algebraic signatures $Sig_g(B_i) \,||\, V_i$ and $CF_{PoW}[F_i]$.

If a subsequent uploader $u_j$ uploads a file $F_i$, $u_j$ will perform the challenge and verification of SE-PoW to resist duplicate-faking attacks and replay attacks in Algorithm 2.

***Challenge of SE-PoW.*** An subsequent uploader $u_j$ generates the file tag and outsources it to the CC to perform inter-edge file-level deduplication. Specifically, $u_j$ computes the file tag $T_{F_i} \leftarrow H_1(H_1(F_i))$ and outsources $T_{F_i}$ to the central cloud. The CC searches $T_{F_i}$ in the global file index. If it does not exist, the CC will return the result to $u_j$. Otherwise, CC randomly selects $c$ indices of blocks $I[k]$ $(0 \leqslant k < c)$ as the challenge, and returns it to the edge node and $u_j$.

***Verification of SE-PoW.*** Then, CC will perform verification of SE-PoW among CC, EN, and $u_j$ via a dual-level PoW, including a cuckoo filter and algebraic signatures. Specifically, details are described in Algorithm 2. (1) $u_j$ divides file $F_i'$ into blocks and generates tokens of the challenged position belong to $I[k]$ $(0 \leqslant k < c)$ via $t_{B_k'} \leftarrow H_2(B_k')$. Then $u_j$ sends $t_{B_k'}$ to EN and CC for the first-level verification of PoW. CC receives $t_{B_k'}$ and computes $e_k' \leftarrow$ PRF($t_{B_k'}$, $k$). Then CC executes $\eta =$ ContainCF($CF_{PoW}[F_i]$, $e_k'$) for all tokens. If any token does not exist in $CF_{PoW}$, $u_j$ does not pass the first-level verification of SE-PoW. (2) If $u_j$ passes the first-level verification, CC will request $u_j$ to verify the homomorphism of algebraic signatures. $u_j$ reads the challenged blocks $B_k'(k \in I[k])$ and computes the sum of challenged blocks via $\gamma \leftarrow \sum_{k=0}^{c-1} B_k'$. Then, $u_j$ sends $\gamma$ to the EN. EN computes the algebraic signature $\sigma \leftarrow Sig_g(\gamma)$ and sends $\sigma$ to the CC. (3) CC reads the block signature $Sig_g(B_k)$ of $F_i$ and executes $\mu \leftarrow \sum_{k=0}^{c-1} Sig_g(B_k) \oplus V_k$ $(k \in I[k])$. Finally, CC verifies whether $\sigma$ equals $\mu$ or not. If no, CC will return that $u_j$ does not pass the verification. Otherwise, CC will add $ID_{u_j}$ to the owner list $OList_{F_i}$. $u_j$ just updates the metadata of $F_i$ and does not upload the content of $F_i$.

**Algorithm 2** The challenge and verification of SE-PoW

**Input:** $CF_{PoW}[F_i]$ & $Sig_g(B_k)$.
**Output:** The result of SE-PoW verification.
1: CC generates the index of challenged blocks $I[k]$ $(0 \leqslant k < c)$ and sends to $u_j$.
2: EN requests $u_j$ to divide $F_i'$ into $B_i'$ $((0 \le i < n))$ and selects challenged blocks $B_i'$ according to $I[k]$
3: **while** $k \in I[k]$ **do**
4: $u_j$ executes $t_{B_k'} \leftarrow H_2(B_k')$ and sends to EN.
5: CC executes $e_k' \leftarrow$ PRF($t_{B_i'}$, $k$).
6: CC executes $\eta =$ ContainCF($e_k'$) (First-level PoW)
7: **if** $\eta = 0$ **then**
8: **return** $\perp$
9: **end if**
10: **end while**
11: CC verifies the second-level PoW.
12: **while** $k \in I[k]$ **do**
13: $u_j$ reads the blocks $B_k'$.
14: $u_j$ executes $\gamma \leftarrow \sum_{k=0}^{c-1} B_k'$.
15: **end while**
16: $u_j$ and EN compute $\sigma \leftarrow Sig_g(\gamma)$ and send $\sigma$ to CC.
17: **while** $k \in I[k]$ **do**
18: CC reads the signature $Sig_g(B_k)$ and $V_k$ of $F_i$.
19: CC executes $\mu \leftarrow \sum_{k=0}^{c-1} Sig_g(B_k) \oplus V_k$
20: **end while**
21: **if** $\sigma = \mu$ **then**
22: **return** 1
23: **else**
24: **return** 0
25: **end if**

### 4.4. Implementation detail of SE-PoW

We propose a prototype based on the design of SE-PoW. To achieve the balance between security and efficiency, SE-PoW implements a dual-level hybrid encrypted deduplication in edge computing. Thus, SE-PoW lessens the pressure on network bandwidth and improves data security and privacy. Specifically, SE-PoW adopts a global file index in the central cloud and block indices in the edge nodes. The index is a key–value storage structure for tags and data storage locations, for example, hash tables. The key is the block's tag, and the value is the physical address of the data block (such as block offset and length). Furthermore, the hash and encryption function in SE-PoW is the CTR mode of SHA-256 and AES-256 [43], and the token calculation uses the SHA-1 function. The secure network transmission between the edge nodes and the central cloud uses SSL/TLS [43]. A trusted edge node is used to compute server-aided keys, and RSA-OPRF [18] is implemented for evaluation.

To realize the dual-level PoW, SE-PoW uses an efficient cuckoo filter [40] with better performance and lower false positive rate than a bloom filter. The cuckoo filter supports InitCF(), AddCF(), ContainCF() and DeleteCF(). In addition, the overall collision probability of an algebraic signature used in SE-PoW is very low [35].

## 5. Security analysis

SE-PoW is designed to ensure data confidentiality and backward privacy and resist attacks for encrypted deduplication in edge computing. We consider two types of adversaries: inside and outside adversaries. We assume that the following technologies are secure, such as symmetric encryption [43] and OPRF protocols [18]. In worst cases, the adversaries may compromise the CC and collude with users.

### 5.1. Data confidentiality

In this case, the adversary gets the ciphertexts of blocks by compromising the CC or EU. SE-PoW resists brute-force attacks in the hybrid deduplication scheme and ensures data confidentiality and tag consistency.

In general, the adversary obtains the ciphertext of target block $C_i^1 \| C_i^2 \| C_i^3$ $(0 \le i < n)$ from a specific file $F_i$. The adversary knows that the blocks $\{B_i'\}(0 \le i < n)$ are from a specific set $|S|$. For each block $B_i'$, the adversary first gets the hash to get the key via $K_{B_i}$. The adversary gets the ciphertext via $C_i^{1'} \leftarrow Encry(K_{B_i'}, B_i')$ and compares it with $C_i^1$. However, $C_i^1$ is protected by the master key $K_{e_i}$ of each EN. SE-PoW generates a random file key $K_{F_i}$ via an oblivious pseudorandom function. All block keys $K_{B_i}$ are protected securely by random key $L_i \,||\, K_{F_i}$. Thus the adversary cannot get the plaintext of file $F_i$. As a result, SE-PoW can resist brute-force attacks to ensure data confidentiality. In addition, the adversary compromises the data integrity by colluding with users. It uploads the valid tags but replaces the blocks with poisoned data. SE-PoW computes the hash value of $C_i^1$ via $T_{B_i'} \leftarrow H_1(C_i^1)$ and compares whether or not $T_{B_i'}$ equals $T_{B_i}$. Thus, SE-PoW ensures tag consistency.

We discuss the security of SE-PoW under different situations. In the best case, the adversary compromises the CC but cannot access the EN. All data and metadata stored in the CC are encrypted with random keys. The adversary cannot obtain the plaintext of files even if it performs brute-force attacks. In the worst case, the adversary may get the master key of a specific EN and collude with malicious users. SE-PoW can still ensure security for unpredictable data that are not falling into a known set. The users access the EN through an intra-network, which naturally faces fewer security threats than inter-network. SE-PoW makes the worst-case rarely occur by further protecting the EN and file metadata with access control policies.

### 5.2. Security of proof of ownership

For a file $F_i$, the adversary's goal is to pass the verification of SE-PoW by leveraging replay attacks or the false positive in a cuckoo filter. The adversary knows parts of the file, but he does not own the entire content of the file.

We define that the event $v_i$ is the adversary could pass the verification of SE-PoW when he gets a *token*. It happens in the following two cases: (1) The adversary receives the correct proof. (2) When the cuckoo filter checks the element, a false positive occurs. We define the false positive of the CF as $p_f$. According to the above analysis, the probability of event $v_i$ can be described as:

$$\begin{aligned} P(v_i) &= P(v_i \cap (token_i \cup \overline{token_i})) \\ &= P(v_i|token_i)P(token_i) + P(v_i|\overline{token_i})P(\overline{token_i}) \\ &= P(token_i) + p_f P(\overline{token_i}) \end{aligned} \tag{4}$$

The adversary performs replay attacks by leveraging the previous proofs and the false positive of the cuckoo filter. After receiving the proofs, the CC will verify the ownership. To resist these attacks, SE-PoW adopts algebraic signatures as the second verification of PoW. It satisfies the property that the sum of algebraic signatures of challenged blocks equals the signature of the sum of challenged blocks. That is whether or not $\sigma = \mu$.

$$\begin{aligned} \sigma &= Sig_g(\gamma) \\ &= S_\lambda(\sum_{k=0}^{c-1} B_k') \\ &= S_\lambda(B_0' + B_1' + \cdots + B_{c-1}') \\ &= S_\lambda(B_0') + S_\lambda(B_1') + S_\lambda(B_{c-1}') \\ &= \sum_{k=0}^{c-1} S_\lambda(B_k') \end{aligned} \tag{5}$$

After receiving the proofs from the EN and end user, the CC could verify the ownership.

$$\begin{aligned} \mu &= \sum_{k=0}^{c-1} Sig_g(B_k) \oplus V_i \\ &= \sum_{k=0}^{c-1} S_\lambda(B_k \| ID_{F_i} \| i) \oplus S_\lambda(ID_{F_i} \parallel i) \\ &= \sum_{k=0}^{c-1} S_\lambda(B_k) \oplus \lambda^l S_\lambda(ID_{F_i} \parallel i) \oplus S_\lambda(ID_{F_i} \parallel i) \\ &= \sum_{k=0}^{c-1} S_\lambda(B_k) \\ &= \sigma \ (B_k' = B_k) \end{aligned} \tag{6}$$

Then, SE-PoW prevents the attacks of the false positive of CF via a dual-level PoW. Moreover, SE-PoW can resist replay attacks because the adversary does not know $V_i$. Thus we denote:

$$p_f P(\overline{token_i}) = 0, \quad and \quad P(v_i) = P(token_i) \tag{7}$$

We define event $g_i$, the adversary gets tokens of the challenged block $B_i$, and the probability is $p$. The token is the output of the hash function of $H_2$ with the length $l$. Based on the random oracle model, the probability of guessing the correct token is $2^{-l}$. Thus, the probability of event $token_i$ is:

$$\begin{aligned} P(token_i) &= P(token_i \cap (g_i \cup \overline{g_i})) \\ &= P(token_i|g_i)P(g_i) + P(token_i|\overline{g_i})P(\overline{g_i}) \\ &= p + (1-p) \cdot 2^{-l} \end{aligned} \tag{8}$$

The adversary needs to get at least $c$ tokens of challenged blocks. Thus, the probability $P(succ)$ is defined as the adversary can pass the verification of SE-PoW.

$$P(succ) = (p + (1-p) \cdot 2^{-l})^c \tag{9}$$

We set up a security parameter $k$ to derive a lower bound for $c$, that is $P(succ) \leqslant 2^k$. To ensure the security of SE-PoW, the number of challenged blocks is:

$$c \ge \frac{k \ln 2}{p + (1-p) \cdot 2^{-l}} \tag{10}$$

The probability of running a successful SE-PoW should be negligible under the security parameter $k$ and the number of tokens $c$. SE-PoW can resist duplicate-faking attacks, and it also prevents replay attacks and the false positive of CF.

### 5.3. Security discussion of SE-PoW

Table 2 shows the comparison results of encrypted deduplication schemes. Halevi [22] and Xu [25] refer to the encrypted deduplication schemes that implement with MHT-PoW and the variants of CE. Yang [15] encrypts data with server-aided MLE and achieves MHT-PoW via ECC-based accumulators. In addition, Lorena [37] and Jiang [21] realize an encrypted deduplication via BF-PoW. The difference is that

**Table 2**
Comparison of encrypted schemes with PoW.

| Scheme | Brute-force attack | Duplicate-faking attack | Replay attack | Perf.[a] |
|---|---|---|---|---|
| Halevi [22]/ Xu [25] | × | √ | × | *L* |
| Yang [15] | √ | √ | × | *L* |
| Lorena [37] | × | √ | × | *H* |
| Jiang [21] | × | √ | × | *H* |
| SE-PoW | √ | √ | √ | *H* |

[a]"L" means Low and "H" refers to High.

**Table 3**
Description of three real-world datasets.

| Name | Size (GB) | Num. | Description |
|---|---|---|---|
| LNX-set | 111.32 | 258 | 258 tar files of linux source code |
| VMA-set | 58.67 | 135 | Virtual machine images, including Fedroa & Ubuntu. |
| WEB-set | 43.31 | 16 | 16 days of snapshot files, retrieval depth is 3 by wget |

they use CE and RCE, respectively. We discuss them regarding resistance to brute-force attacks, duplicate-faking attacks and replay attacks, and performance.

Since all the schemes allow users to encrypt data and realize deduplication over ciphertexts, they can guarantee data confidentiality. On the one hand, the method of Halevi et al. suffers from brute-force attacks because of the utilization of CE. The scheme of Halevi et al. [22] and Yang et al. [15] are both vulnerable to replay attacks and incur large time overheads due to the Merkle Hash Tree. On the other hand, Lorena et al. [37], and Jiang et al. [21] cannot prevent brute-force attacks and replay attacks, but they achieve high performance. As mentioned above, SE-PoW can resist brute-force attacks and ensure data confidentiality and tag consistency. Furthermore, SE-PoW also resists duplicate-faking attacks and replay attacks to ensure backward privacy, only adding little overheads compared with the scheme of Jiang [21].

## 6. Performance evaluation

### 6.1. Experimental setup

**Platform:** We conduct experiments to evaluate the performance of SE-PoW. These machines are equipped with an Intel(R) Core(TM) i7-4770@3.40 GHZ 8-core CPU, 96 GB memory and 2 TB hard disk. They are installed with an Ubuntu 20.04 LTS 64-bit operation system. These machines are connected with 100 Mbps and 1000 Mbps ethernet network.

**Methodology:** To evaluate the performance of SE-PoW, we implement a research prototype to compare the related schemes, including MHT-PoW [22,25] and BF-PoW [21,26,30,37]. MHT-PoW is a file-level encrypted deduplication scheme based on the MHT and variants of CE [22,25]. BF-PoW refers to Jiang et al. [21] scheme that is a block-level encrypted deduplication scheme with BF-PoW and hybrid RCE algorithms. Meanwhile, the encryption schemes consist of convergent encryption(CE), server-aided Message-locked Encryption(MLE), and SE-PoW. We mainly use quantitative metrics for encryption time, the cumulative time of SE-PoW, initial and subsequent upload time, metadata, and storage overheads. The time of SE-PoW consists of phases: initialization, challenge, and verification. We also observe the impacts of varying block size, number of tokens, and file size.

Finally, the security parameters are set according to MHT-PoW [22] and BF-PoW [21,37]. Where security parameters, the number $k$ is 66, and the token length is set to 16 bytes. According to formula (10) in the security analysis, the number of challenge blocks is set {102, 204, 509, 1017}. Note that our evaluation results should be interpreted as an approximate assessment of other schemes.

**Datasets:** There are two types of datasets used in SE-PoW for performance evaluation, including synthetic datasets and real-world datasets. (1) Synthetic datasets: artificial files with random content of different sizes or different average block size, and each file is divided into fixed-size blocks. (2) Table 3 describes the real-world datasets, which contain three different types, namely LINUX-set, VMA-set and WEB-set. Linux-set contains the tar package file of the 258 version of the Linux source code. VMA-set [44] is collected images of different operating systems of virtual machines, including Fedora and Ubuntu. WEB-set is a snapshot of 15-day web pages downloaded from *news.sina.com* using the tool *wget*, and the maximum retrieval depth is 3.

### 6.2. A sensitivity study on encryption & PoW

This subsection evaluates the performance of encryption and proof-of-ownership algorithms varying different block sizes and file sizes with synthetic datasets. First, to assess time overheads of encryption, we upload a 1024 MB unique file repeatedly with varying block sizes, i.e., 2 kB, 4 kB, and 8 kB. Second, to evaluate time overheads of related pow schemes, we use files that are generated with random contents of size $2^i$ kB for $i \in \{5, \ldots, 21\}$, which ranging from 16 kB to 2048 MB. Third, to evaluate the performance of SE-PoW, we use a 2 GB file varying different average block sizes, file sizes, and the number of tokens.

Fig. 5(a) shows that the location-aware hybrid encryption scheme used in SE-PoW reduces more time overheads than server-aided MLE, and it is similar to CE and RCE as discussed in Section 4.2. In addition, the larger the average block size, the shorter time overhead. It is because the OPRF protocol costs a lot and the time of key generation decreases, as discussed in Section 2. As described in other papers [30], encrypted deduplication schemes based on proxy re-encryption [21] also incur high computation cost.

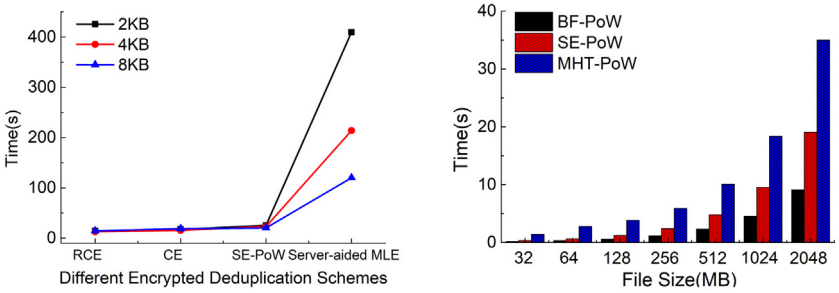
We also evaluate the overheads on the edge side of SE-PoW. SE-PoW mainly adds the time overheads of data re-encryption, tag generation, and duplication checking in the block-level index. For example, we use a 2 GB unique file with random content, and the average block size is 8 kB. We evaluate the server-side overhead of SE-PoW. The re-encryption time is 10.639 s. The time of tag generation and duplication checking are 0.841 s and 8.576 s, respectively.

Fig. 5(b) shows the results that SE-PoW significantly reduces the cumulative time compared with MHT-PoW and only increases little overheads than BF-PoW as discussed in Section 4.3. Specifically, for an individual file of 1 GB, SE-PoW reduces 70–86.7% time overheads relative to MHT-PoW. The erasure coding and construction of the Merkle hash tree used in MHT-PoW incur large time overheads. Compared with BF-PoW, SE-PoW only increases 13.2–14.9% the cumulative time overheads because of the calculation of algebraic signatures.

Fig. 6(a), (b) and (c) evaluate the time overheads of the proof of ownership protocol in SE-PoW, including initialization, challenge and verification phases. Fig. 6(a) shows that the cumulative time increases with the file size, and the initialization phase accounts for more than 60%. The time overhead of PoW is low. For example, SE-PoW costs 0.88 s for a file with 2 GB. Fig. 6(b) evaluates the performance of varying different average block sizes. The result shows that the cumulative time of SE-PoW decreases with the increase of the average block size. Fig. 6(c) shows that only the verification phase costs more time for a larger number of tokens, as discussed in Section 4.3.

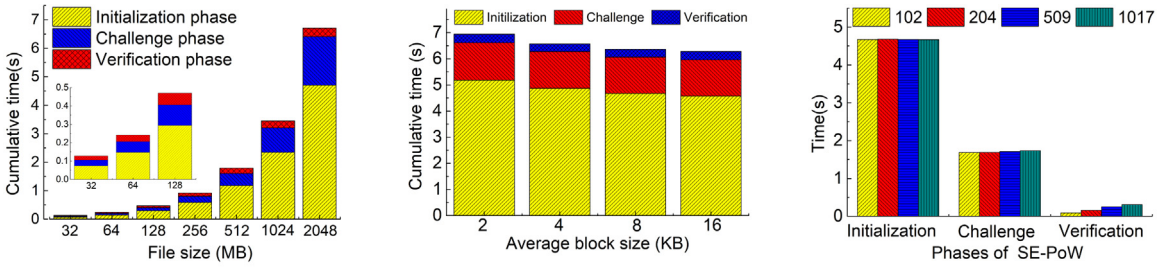
### 6.3. Evaluating SE-PoW on real-world datasets

In this subsection, we evaluate the overall performance of SE-PoW compared with MHT-PoW and BF-PoW on three real-world datasets. First, we assess the storage and metadata overheads. Second, the user



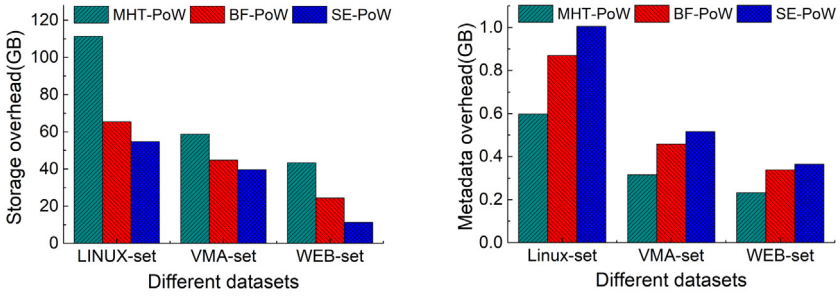

(a) Time of encrypted deduplication schemes (b) The cumulative time of three PoWs

**Fig. 5.** The encryption time on different average block size and the cumulative time of three PoW methods.



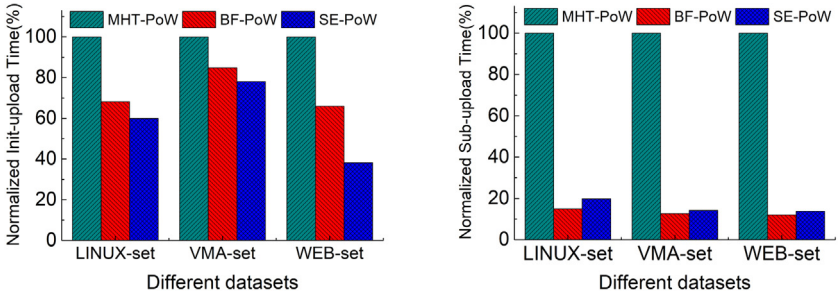

(a) File Size (b) Average block size (c) Number of tokens

**Fig. 6.** He cumulative time overheadT of SE-PoW varying on file size, average block length and the number of tokens.




(a) Storage overhead (b) Metadata overhead

**Fig. 7.** Comparison of storage overhead and metadata overhead under datasets of MHT-PoW, BF-PoW and SE-PoW.




(a) Initial upload time of the file (b) Subsequent upload time of the file

**Fig. 8.** The relative time of the initial and subsequent uploads of files under datasets of MHT-PoW, BF-PoW and SE-PoW.

performs file-level and block-level deduplication in the first upload, uploads non-duplicated data blocks, and initializes the PoW protocol.

Fig. 7(a), compared with MHT-PoW, SE-PoW reduces storage overhead by 31.7–73.3%. SE-PoW reduces storage overhead by 10.8–52.9% relative to BF-PoW. In Fig. 7(b), the growth trend of metadata overhead is exactly the opposite of storage overhead. SE-PoW increases 56.6–68.2% and 7.7–15.6% metadata overheads compared with MHT-PoW and BF-PoW. MHT-PoW has less metadata. BF-PoW needs to store tokens of blocks, while SE-PoW stores extra algebraic signatures of all blocks. Compared to MHT-PoW and BF-PoW, SE-PoW reduces the overall data and metadata storage overhead by 31.7–73.3% and 10.8–52.9%, respectively. It is because that SE-PoW combines them for inter-edge and intra-edge and utilizes a content-defined chunking algorithm to balance efficiency and storage overheads.

As shown in Fig. 8(a), in the initial upload, SE-PoW reduces 21.9–61.9% and 6.8–27.7% upload time compared with MHT-PoW and BF-PoW under the real-world datasets. The encoding and construction of the Merkle Hash tree bring large computation overheads, as

discussed in Section 2.2. Fig. 8(b) shows that SE-PoW reduces the subsequent upload time by over 80% compared with MHT-PoW. And SE-PoW increases about 14.4% subsequent upload time related to BF-PoW. Compared with BF-PoW, SE-PoW adds the calculation overheads of algebraic signatures (See Section 4.3).

## 7. Related work

Edge computing has been gaining much popularity in recent years. Data deduplication at the network edge can exploit the geographic distribution and low latency to achieve high performance and optimized storage cost. Li et al. [11] partition the resource-constrained edge nodes into disjoint clusters. They perform decentralized deduplication within these clusters to improve the deduplication ratio. They also present HotDedup [13] to maximize edge service rate and storage efficiency with deduplication at the network edge by exploiting data popularity and similarity. Cheng et al. [12] proposed LOFS, a file storage strategy via a three-layer hash mapping scheme to allocate files to the proper edge servers for data deduplication. However, they do not solve the problem of data confidentiality and proof-of-ownership.

**Encrypted Deduplication.** To protect data confidentiality of deduplication, randomized convergent encryption(CE) and message-locked encryption(MLE) and their variants have been proposed in [16,17]. To resist brute-force attacks, DupLESS [18] and ClearBox [45] leverage server-aided MLE via an oblivious pseudorandom protocol (e.g., RSA-OPRF, BLS-OPRF). Liu et al. [46] present a secure deduplication scheme without additional independent servers by using a PAKE protocol. The convergent key management [38,47] are studied to ensure key reliability and reduce space overheads. Moreover, encrypted deduplication has gained much attention in fog and edge computing. Koo et al. [14] combine server-side deduplication and client-side deduplication in fog computing. Fo-SSD [48] leverages BLS-OPRF to support encrypted deduplication and enables fog nodes to remove replicate data. Yang et al. [15] use a hash proof system-based OPRF to resist brute-force attacks and provide dynamic cross-domain deduplication in blockchain-enabled edge computing. However, they do not address the problem of PoW or suffer from potential attacks and time overheads.

**Proof-of-Ownership.** To solve the problem that attackers can access files with a small hash value, Halevi et al. [22] present MHT-PoW, using erasure coding to build a Merkle Hash Tree(MHT) for ownership verification. Ng et al. [49] proposed a private PoW scheme over encrypted data. Xu et al. [25] firstly encrypt data and generate a hash digest to construct a Merkle Hash Tree, which enhances data security of client-side deduplication under a bounded leakage setting. Yang et al. [15] adopt ECC-based accumulators for MHT-PoW and achieve better performance. However, these schemes require high computation and I/O overheads, which are not suitable for edge computing. Pietro et al. [50] propose s-PoW to reduce computation and I/O overheads, which outputs a proof with each bit that is selected at a random position of the file. BF-PoW [21,26,30,37] generates a token for each block and inserts tokens into a bloom filter for ownership checking under the bounded leakage setting. In addition, access control and user revocation have been studied. REED [39] encrypts data with a deterministic version of the all-or-nothing transform. It achieves deduplication with dynamic access control. Nevertheless, they suffer from privacy leakage of false positives in a bloom filter and replay attacks.

## 8. Conclusion

Nowadays, edge computing employs data deduplication to reduce storage and computation overheads. However, the state-of-the-art schemes face some security and performance problems, including data confidentiality and security of proof-of-ownership. We design SE-PoW, which employs a location-aware hybrid encrypted deduplication method and a dual-level security-enhanced proof-of-ownership algorithm.

To resist brute-force attacks, SE-PoW exploits server-aided RCE for inter-edge file-level encrypted deduplication. For non-duplicate files, SE-PoW utilizes edge-aided MLE for intra-edge block-level encrypted deduplication. To resist duplicate-faking attacks, we further exploit a cuckoo filter as the first-level PoW to verify the ownership. Then we prove the homomorphism of algebraic signatures to enhance the security of SE-PoW and resist replay attacks. Finally, the security analysis demonstrates that SE-PoW achieves higher security. And the performance evaluation makes it clear that SE-PoW is efficient compared with the state-of-the-art schemes. The problems of data reliability, updating, and dynamic user management are our future work.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

We want to thank the reviewers and editors for their constructive comments and suggestions. This research is partly supported by ZDSY20200811143600002. We also thank anyone who helped us improve this work.

## References

[1] Fengmin Tang, Feng Gao, Zilong Wang, Driving capability-based transition strategy for cooperative driving: From manual to automatic, IEEE Access 8 (2020) 139013–139022.
[2] Wang Xiaoyang, Ma Yao, Wang Yiqi, Jin Wei, Wang Xin, Tang Jiliang, Jia Caiyan, Jian Yu, Traffic flow prediction via spatial temporal graph neural network, in: Proceedings of the Web Conference 2020(WWW'20), 2020, pp. 1082–1092.
[3] The future of data: Data age 2025, 2019, https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf.
[4] Rethik data: Put more of your business data to work - from edge to cloud, 2021, https://www.seagate.com/files/www-content/our-story/rethink-data/files/Rethink_Data_Report_2020.pdf.
[5] Tim Süß, Tunahan Kaya, Markus Mäsker, Andre Brinkmann, Deduplication analyses of multimedia system images, in: USENIX Workshop on Hot Topics in Edge Computing (HotEdge'18), Boston, MA, 2018.
[6] Jianbing Ni, Kuan Zhang, Yong Yu, Xiaodong Lin, Xuemin Sherman Shen, Providing task allocation and secure deduplication for mobile crowdsensing via fog computing, IEEE Transactions on Dependable and Secure Computing(TDSC) 17 (3) (2020) 581–594.
[7] Yu Wang Hongyang Yan, Chunfu Jia, Centralized duplicate removal video storage system with privacy preservation in IoT, Sensors 18 (6) (2018) 1814.
[8] Dropbox, 2022, https://www.dropbox.com/.
[9] Microsoft OneDrive, 2022, https://drive.google.com/.
[10] Ctera edge X series, 2022, https://www.ctera.com/x-series/.
[11] Shijing Li, Tian Lan, Bharath Balasubramanian, Moo-Ryong Ra, Hee Won Lee, Rajesh Panta, EF-Dedup: Enabling collaborative data deduplication at the network edge, in: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019, pp. 986–996.
[12] Geyao Cheng, Deke Guo, Lailong Luo, Junxu Xia, Siyuan Gu, LOFS: A Lightweight online file storage strategy for effective data deduplication at network edge, IEEE Trans. Parallel Distrib. Syst. (TPDS) (01) (2021) 1.
[13] Shijing Li, Tian Lan, Hotdedup: managing hot data storage at network edge through optimal distributed deduplication, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 247–256.
[14] Dongyoung Koo, Youngjoo Shin, Joobeom Yun, Junbeom Hur, A hybrid deduplication for secure and efficient data outsourcing in fog computing, in: IEEE CloudCom'2016, 2016, pp. 285–293.
[15] Yang Ming, Chenhao Wang, Hang Liu, Yi Zhao, Jie Feng, Ning Zhang, Weisong Shi, Blockchain-enabled efficient dynamic cross-domain deduplication in edge computing, IEEE Internet Things J. (2022) 1.
[16] J. Douceur, A. Adya, W.J Bolosky, et al., Reclaiming space from duplicate files in a serverless distributed file system, in: Proceedings of IEEE ICDCS, 2002, pp. 617–624.
[17] M. Bellare, S. Keelveedhi, T. Ristenpart, Message-locked encryption and secure deduplication, in: Proceedings of EUROCRYPT, 2013, pp. 296–312.
[18] S. Keelveedhi, M. Bellare, T. Ristenpart, DupLESS: server-aided encryption for deduplicated storage, in: Proceedings of Usenix Security, 2013, pp. 1–16.

[19] Jingwei Li, Zuoru Yang, Yanjing Ren, Patrick P.C. Lee, Xiaosong Zhang, Balancing Storage Efficiency and Data Confidentiality with Tunable Encrypted Deduplication, Association for Computing Machinery, New York, NY, USA, 2020.
[20] Junbeom Hur, Dongyoung Koo, Youngjoo Shin, Kyungtae Kang, Secure data deduplication with dynamic ownership management in cloud storage, IEEE Transactions on Knowledge and Data Engineering(TKDE) 28 (11) (2016) 3113–3125.
[21] Shunrong Jiang, Tao Jiang, Liangmin Wang, Secure and efficient cloud data deduplication with ownership management, IEEE Transactions on Services Computing (TSC) 13 (6) (2018) 1152–1165.
[22] Shai Halevi, Danny Harnik, Benny Pinkas, Alexandra Shulman-Peleg, Proofs of ownership in remote storage systems, in: Proceedings of ACM CCS, 2011.
[23] Dropship: dropbox aip utilities, 2012, https://github.com/driverdan/dropship.
[24] Martin Mulazzani, Sebastian Schrittwieser, Manuel Leithner, Markus Huber, Edgar Weippl, Dark clouds on the horizon: Using cloud storage as attack vector and online slack space, in: The 20th USENIX Security Symposium (Security'11), 2011, pp. 363–370.
[25] Jia Xu, Ee-Chien Chang, Jianying Zhou, Weak leakage-resilient client-side deduplication of encrypted data in cloud storage, in: Proceedings of ACM AsiaCCS, 2013, pp. 195–206.
[26] J. Blasco, R. DiPietro, A. Orfila, A. Sorniotti, A tunable proof of ownership scheme for deduplication using Bloom filter, in: Proceedings of IEEE CNS, 2014, pp. 481–489.
[27] Dongyoung Koo, Junbeom Hur, Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing, Future Generation Computer Systems(FGCS) 78 (2018) 739–752.
[28] Haoran Yuan, Xiaofeng Chen, Jianfeng Wang, Jiaming Yuan, Hongyang Yan, Willy Susilo, Blockchain-based public auditing and secure deduplication with fair arbitration, Inform. Sci. 541 (2020) 409–425.
[29] B.H. Bloom, Spacetime trade-offs in hash coding with allowable errors, Commun. ACM 13 (7) (1970) 422–426.
[30] Jinbo Xiong, Yuanyuan Zhang, Shaohua Tang, Ximeng Liu, Zhiqiang Yao, Secure encrypted data with authorized deduplication in cloud, IEEE Access 7 (2019) 75090–75104.
[31] Xue Yang, Rongxing Lu, Kim Kwang Raymond Choo, Fan Yin, Xiaohu Tang, Achieving efficient and privacy-preserving cross-domain big data deduplication in cloud, IEEE Trans. Big Data 8 (1) (2022) 73–84.
[32] Yong Yu, Yafang Zhang, Jianbing Ni, Man Ho Au, Lanxiang Chen, Hongyu Liu, Remote data possession checking with enhanced security for cloud storage, 52 (C), November 2015.
[33] Dutch T. Meyer, William J. Bolosky, A study of practical deduplication, in: The 9th USENIX Conference on File and Storage Technologies (FAST'11), USENIX Association, San Jose, CA, USA, 2011, pp. 229–241.
[34] The digitization of the world - from edge to core, 2019, https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf.
[35] Thomas J.E. Schwarz, Ethan L. Miller, Store, forget and check: Using algebraic signatures to check remotely administered storage, in: Proceedings of IEEE ICDCS, 2006, pp. 1–12.
[36] Jianbing Ni, Xiaodong Lin, Kuan Zhang, Yong Yu, Secure and deduplicated spatial crowdsourcing: A fog-based approach, in: IEEE GLOBECOM, 2016, pp. 1–6.
[37] Lorena Gonz'alez-Manzano, Agusti'n Orfila, An efficient confidentiality-preserving proof of ownership for deduplication, J. Netw. Comput. Appl. 50 (2015) 49–59.
[38] Yukun Zhou, Dan Feng, Wen Xia, Min Fu, Fangting Huang, Yucheng Zhang, Chunguang Li, SecDep: A user-aware efficient fine-grained secure deduplication scheme with multi-level key management, in: Proceedings of IEEE MSST, 2015, pp. 1–14.
[39] Jingwei Li Chuan Qin, Patrick P.C. Lee, The design and implementation of a rekeying-aware encrypted deduplication storage system, ACM Trans. Storage (TOS) 13 (1) (2017) 9:1–9:30.
[40] B. Fan, D. G. Andersen, M. Kaminsky, M. D. Mitzenmacher, Cuckoo filter: Practically better than bloom, in: Proceedings of ACM CoNEXT, 2014, pp. 75–88.
[41] W. Litwin, T. Schwarz, Algebraic signatures for scalable distributed data structures, in: Proceedings. 20th IEEE ICDE, 2004, pp. 412–423.
[42] Jian Shen, Dengzhi Liu, Debiao He, Xinyi Huang, Yang Xiang, Algebraic signatures-based data integrity auditing for efficient data dynamics in cloud computing, IEEE Trans. Sustain. Comput. (ISSN: 2377-3782) 5 (02) (2020) 161–173.
[43] OpenSSL Project, 2022, https://www.openssl.org/.
[44] W. Xia, Y. Zhou, H. Jiang, D. Feng, Y. Hua, Y. Hu, Q. Liu, Y. Zhang, FastCDC: A fast and efficient content-defined chunking approach for data deduplication, in: Proceedings of USENIX ATC'16, 2016, pp. 101–114.
[45] A. Frederik, B. Jens-Matthias, K. Ghassan O., Y. Franck, Transparent data deduplication in the cloud, in: Proceedings of ACM CCS'15, 2015, pp. 886–900.
[46] Jian Liu, N. Asokan, Benny Pinkas, Secure deduplication of encrypted data without additional independent servers, in: Proceedings of ACM CCS, 2015, pp. 874–885.
[47] Jin Li, Xiaofeng Chen, Mingqiang Li, Jingwei Li, Patrick PC Lee, Wenjing Lou, Secure deduplication with efficient and reliable convergent key management, IEEE TPDS 25 (6) (2014) 1615–1625.
[48] Jianbing Ni, Kuan Zhang, Yong Yu, Xiaodong Lin, Xuemin Sherman Shen, Providing task allocation and secure deduplication for mobile crowdsensing via fog computing, IEEE Transactions on Dependable and Secure Computing (TDSC) (2018).
[49] Wee Keong Ng, Yonggang Wen, Huafei Zhu, Private data deduplication protocols in cloud storage, in: Proceedings of ACM SAC, 2012, pp. 441–446.
[50] Roberto Di Pietro, Alessandro Sorniotti, Boosting efficiency and security in proof of ownership for deduplication, in: Proceedings of the 7th ACM AsiaCCS, 2012, pp. 81–82.

**Yukun Zhou** received the B.E. and Ph.D. degrees in computer science and technology from Huazhong University of Science and Technology in 2013, and 2019, respectively. He is an expert at Sangfor Inc. His research interests include storage security and edge computing. His research works have been published in Usenix ATC, IEEE TC, TPDS, INFOCOM, MSST, FGCS, etc.

**Zhibin Yu** received his Ph.D. degree in computer science from Huazhong University of Science and Technology (HUST) in 2008. He visited the Laboratory of Computer Architecture (LCA) of ECE of the University of Texas at Austin for one year and he worked in Ghent University as a postdoctoral researcher for half of a year. Now he is a professor in SIAT. His research interests are micro-architecture simulation, computer architecture, workload characterization and generation, performance evaluation, multi-core architecture, GPGPU architecture, virtualization technologies, big data processing and so forth. He won the outstanding technical talent program of Chinese Academy of Science (CAS) in 2014 and the 'peacock talent' program of Shenzhen City in 2013. He is a member of IEEE and ACM. He serves for ISCA 2013, 2015, 2020, 2021, 2022, MICRO 2014, HPCA 2015, 2018, 2020, PACT 2016, and ICS2018.

**Liang Gu** received the Ph.D. degree in Computer Software and Theory from Peking University in 2010. He worked as an associate research fellow at Yale University from 2010 to 2015. He is currently the chief scientist and the director of Sangfor Research Institute at Sangfor Technology Inc. As the person in charge of R&D technology at Sangfor, he is responsible for the technical framework improvement of a series of core products, including NGAF, AC, a Cloud HCI, aSAN, etc.

**Dan Feng** received the B.E., M.E., and Ph.D. degrees in computer science and technology from Huazhong University of Science and Technology (HUST), Wuhan, China, in 1991, 1994, and 1997, respectively. She is a Professor and the Dean of the School of Computer Science and Technology, HUST. Her research interests include computer architecture, non-volatile memory technology, distributed and parallel file system, and massive storage system. She published more than 100 papers in IEEE TC, TPDS, TCAD, ACM TOS, FAST, USENIX ATC, EuroSys, ICDCS, SC, ICS, IPDPS, and DAC. She is a member of IEEE and ACM, chair of Information Storage Technology Committee of Chinese Computer Academy.