


ORIGINAL RESEARCH

Blockchain-based deduplication with arbitration and incentives

Ke Huang¹  | Xiaosong Zhang^{1,2} | Yi Mu³ | Fatemeh Rezaeibagha⁴ |
Xiaoming Huang⁵ | Yongcheng Gong⁶

¹College for Cyber Security, University of Electronic Science and Technology of China (UESTC), Chengdu, China

²Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China

³Institute of Data Science, City University of Macau, Macau, China

⁴School of Engineering and Information Technology, Murdoch University, Perth, Australia

⁵CETC Cyberspace Security Research Institute Co., Ltd., Chengdu, China

⁶School of Optoelectronic Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China

Correspondence

Xiaosong Zhang, College for Cyber Security, University of Electronic Science and Technology of China (UESTC), Chengdu 611731, China.
Email: johnsonzxs@uestc.edu.cn

Funding information

National Natural Science Foundation of China, Grant/Award Numbers: 62002048, 61872087, U19A2066, 62072078; National Key R&D Program of China, Grant/Award Numbers: 2017YFB0802300, 2018YFB08040505, 2018YFB0804100; Research Initiation Fund (Central Universities), Grant/Award Number: Y030202059018061; Blockchain Research Lab of UESTC, Chengdu Jiaozhi Financial Holding Group Company Ltd

Abstract

Cloud storage is an ideal platform to accommodate massive data. However, with the increasing number of various devices and improved processing power, the amount of generated data is becoming gigantic. Therefore, this calls for a cost-effective way to outsource massively generated data to a remote server. Cloud service providers utilise deduplication technique which deduplicates redundant data by aborting identical uploading requests and deleting redundant files. However, current deduplication mechanisms mainly focus on the storage saving of the server, and ignore the sustainable and long-term financial interests of servers and users. This is not helpful to expand outsourcing and deduplication services. Blockchain is an ideal solution to achieve an economical and incentive-driven deduplication system. Though some current research studies have integrated deduplication with blockchain, they did not utilise blockchain as a financial tool. Meanwhile, it lacks an arbitration mechanism to settle disputes between the server and the user, especially in a Bitcoin payment where the payment is not confirmed immediately and a dispute may occur. This creates a burden to achieve fair and transparent incentive-based deduplication service. In this work, we construct a deduplication system with financial incentives for the server and the user based on Bitcoin. The data owner will pay money via Bitcoin to the server for outsourcing the file, but this fee can be compensated by charging deduplication users with some fees to acquire the deduplication service. The server and the user can receive revenues using deduplication service. Disputes on the fair distribution of incentives can be settled by our arbitration protocol with chameleon hashes as arbitration tags. We give concrete construction and security requirements for our proposed BDAI. The security analysis shows that our BDAI is theoretically secure. The performance evaluation shows that our proposed BDAI is acceptably efficient for the deduplication. Meanwhile, we evaluate and conclude that 1% of outsourcing fee (or less) is a reasonable and preferable price for each deduplication user to pay as compensation for data owner.

1 | INTRODUCTION

Cloud storage is an ideal platform to accommodate massive data [1] generated by various devices. The Cloud Service Providers (CSPs) offer outsourcing services to networked users to access their data whenever and wherever they want. However, with the booming numbers of data sources (e.g., personal computer, phone, pad, sensor, etc) and ever improved processing power, the amount of generated data is becoming gigantic [2]. Hence, it is necessary to improve the storage

efficiency of current cloud servers before being pushed to the limit of storage capacity.

Currently, cloud service providers are keen on using deduplication techniques [3] to reclaim space from storage. This works by deleting redundant data file and aborting redundant upload requests to save both storage and bandwidth. However, in deduplication-oriented storage, leakage of file contents [4] is witnessed more or less. To explain, a malicious user may attempt to retrieve a file owned by others via partial information of the file. As the simplest case, the

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2022 The Authors. *IET Information Security* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

supposed file is deduplicated via hash-as-a-key [4] where the file can be retrieved via a simple hash, and the attacker can easily obtain this key elsewhere and retrieve a file easily. This poses a significant threat to users' data. In addition, despite the efforts being made to enhance the security [5] and efficiency [6] of deduplication, the economics and fairness of the current deduplication mechanism have not been extensively studied yet. Basically, the philosophy of deduplication is to save storage and costs for CSP; otherwise, it is pointless to do so in that the server can trivially accommodate all outsourced data. This brings us to the very fundamental issue: financial gain and loss of the user. We also point out that in some scenarios, the infrastructures and resources devoted to enabling deduplication cost even more than the savings. For instance, as suggested by Jiang et al. [6], the equality test algorithm used to detect block-level deduplication by [7] induces large expenditures and makes the whole deduplication process impractical. On the other hand, as it is surveyed in our previous work [8], the current deduplication mechanism does not consider the financial interests of the server and the user. Therefore, it calls for a practical and economic mechanism for both the server and the user to expand deduplication services. Meanwhile, the current deduplication scheme or the system lacks an arbitration procedure, which leads to the abuses of file sharing without authorisation. As a result, it is difficult to protect the intellectual property (or copyright) of data file, because each user who holds an identical file with others can claim to be the file owner and share it with others. Financial disputes may also occur if we introduce financial incentives to the deduplication system, for instance, a dispute about whether a payment is correctly or timely proceeded. What's more, a public arbitration protocol is needed so that anyone can verify the validity of the arbitration procedure or decision.

Blockchain is a publicly distributed ledger used to record transactions in Bitcoin-like cryptocurrencies. The most successful blockchain application is cryptocurrency (e.g. Bitcoin), a digital currency without reliance on the central bank. Bitcoin [9] is the first successful cryptocurrency proposed by Nakamoto in 2008. Integrating blockchain with cloud computing (or cloud storage) is a natural and promising idea [10]. Current research mainly focuses on exploiting blockchain's immutable and traceable feature to design distributed database, or achieving decentralised service via using blockchain or its core element (e.g. smart contract [11]). These researches fail to consider blockchain as an important method to solve the financial problem in the deduplication system. To note, blockchain is initially proposed to solve the financial problem (decentralised banking system). As we discussed earlier, the application of deduplication is a financial problem to CSP. Thus, exploring and investigating how to achieve an economic deduplication system with blockchain is a reasonable and justifiable motivation.

When deploying blockchain as an incentive mechanism to boost the deduplication service, disputes may arise. Since deduplication only allows for one copy per file in the storage and avoids any redundant outsourcing requests, a malicious user may claim to be the file owner and cheat to get paid.

Meanwhile, who is supposed to pay (the server or the deduplication user) and how much should be paid are also questions. Besides, Bitcoin does not support fast payment [12] since it requires at least 10 min to confirm the payment (also resist double-spending [12]). This leads to potential disputes between the user and the server regarding the validity of payment.

To cope with the aforementioned problems, we introduce financial incentives and arbitration mechanism to design the deduplication system by exploiting Bitcoin [9] as a means of payment. In this work, we construct a deduplication system with financial incentives for the server and the user. Specifically, the server charges each data owner with some outsourcing fees for uploading the first distinct file (later, uploading requests will be deduplicated). The deduplication user who acquires deduplication service will pay the deduplication fee as a reward to the server and the data owner. To our knowledge, our work is the first blockchain-based deduplication focussing on the deduplication economy. Our contributions can be highlighted as below:

- (1) We propose a blockchain-based deduplication with arbitration and financial incentives (BDAI). The data owner pays the server via Bitcoin to gain the outsourcing service, but this fee can be compensated by the deduplication user later. Our arbitration protocol can settle disputes between users and servers for the authenticity of data source and validity of payment. We give concrete construction for our proposed BDAI as well as security requirements to capture potential attacks.
- (2) We briefly instantiate how our proposed BDAI works. Meanwhile, we discuss defining reasonable costs as outsourcing charge sum_m and deduplication rewards sum_m' . The discussion helps build an economically sound deduplication system based on Bitcoin.
- (3) Following previously defined security requirements, we give a comprehensive analysis of the security of our proposed BDAI. We also conduct experiments to test the performance of our proposal. Evaluations show that our proposed BDAI is theoretically and acceptably efficient for deduplication purposes.

2 | PRELIMINARY

We give an overview of some preliminary knowledge used in our work.

2.1 | Complexity assumptions

Let G be a cyclic multiplicative group generated by g with prime order q . We informally state the following assumptions:

Computational Diffie-Hellman Problem (CDHP): Given $g, g^a, g^b \in G$ where $a, b \in_{\mathbb{R}} \mathbb{Z}_q$, there is no Probabilistic Polynomial Time (PPT) adversary \mathcal{A} who can compute g^{ab} efficiently.

Decisional Diffie-Hellman Problem (DDHP): Given $g, g^a, g^b, g^c \in G$ where $a, b, c \in_R \mathbb{Z}_q$, there is no PPT adversary \mathcal{A} who can decide whether $c = ab \bmod q$ efficiently.

q-Strong Diffie-Hellman Problem (q-SDHP) [13]: Choose a random $x \leftarrow_R \mathbb{Z}_q$. Given a $(q + 1)$ -tuple $(g, g^x, \dots, g^{x^q}) \in G^{q+1}$ and g is the generator of q -order cyclic group G , there is no PPT adversary \mathcal{A} who can compute $(c, g^{\frac{1}{x+c}})$ for an unknown c .

2.2 | Bilinear pairing

Given two multiplicative groups G and G_T with same group order q and generator g , denote $\hat{e} : G \times G \rightarrow G_T$ as a symmetric bilinear map where $\hat{e}(x^a, y^b) = \hat{e}(x, y)^{ab}$ for all $x, y \in G$ and $a, b \in_R \mathbb{Z}_q$. Additionally, computing $\hat{e}(g, g) \neq 1$ is efficient.

2.3 | Proof of knowledge

Proof of knowledge (PoK) of a discrete logarithm which allows a prover \mathcal{P} to convince the verifier \mathcal{V} that he knows $X = \log_g Y$ without exposing X . By applying a Schnorr signature [14], a PoK for Y 's discrete logarithm is computed as: $\text{PoK}(Y) = (c, s) \in \{0, 1\}^k \times \mathbb{Z}_q$ where $c = H(g, Y, g^s Y^c; H : \{0, 1\}^* \rightarrow \{0, 1\}^k \text{ is a secure and collision-resilient hash function})$ and $s = r - cX \bmod q$.

To explain, the prover \mathcal{P} selects a randomness $r \leftarrow_R \mathbb{Z}_q$ to compute $c = H(g, Y, g^r)$ and $s = r - cX$. On receiving (c, s) , the verifier \mathcal{V} checks and accepts if $c = H(g, Y, g^s Y^c)$.

2.4 | Unpredictable block source

Our BDAI implements Block-Level Message-Locked Encryption (BL-MLE) [7]. Thus, we consider unpredictable block source in this paper to capture the best security possible.

Let M be a message vector over $\{0, 1\}^*$, $Z \in \{0, 1\}^*$ be some auxiliary information, $n(\lambda)$ denote the number of blocks in M . Therefore, for any $i \in [1, n(\lambda)]$, and $M[i]$ indicates the i th block of the message M . Denote block-source as a polynomial time algorithm \mathcal{M} that on input 1^λ , returns (M, Z) . \mathcal{M} is unpredictable if the guessing probability $\text{GP}_{\mathcal{M}} = \text{Max}_i \{ \text{GP}(M[i]|Z) \}$ is negligible. Here, $\text{GP}(M[i]|Z)$ is the conditional guessing probability of the i th block $M[i]$ on given auxiliary information Z .

2.5 | Block-level message-locked encryption

Chen et al. [7] proposed BL-MLE as an extension of Message-Locked Encryption (MLE) [5]. BL-MLE consists of 10 algorithms. Given an input λ , the parameter generation algorithm **Setup** returns the system parameters P . On input

of a message M , the key generation algorithm **KeyGen** returns a master key and block keys $(k_{mas}, \{k_i\}_{1 \leq i \leq n}) \leftarrow \text{KeyGen}$. Unlike MLE, BL-MLE encrypts per block to derive the ciphertext. The encryption algorithm **Enc**, on input of the block message $M[i]$ and the block key k_i , returns the block ciphertext $C[i]$. The decryption algorithm **Dec** is a reverse process of the **Enc**. On input of the file M , the tag generation algorithm **TagGen** returns the file tag T_0 and block tags $\{T_i\}_{1 \leq i \leq n}$.

On input of two block tags T, T' and two file tags T_0, T'_0 , **EqTest** returns *True* or *False*. As each block key is encapsulated in the block tag, decryption requires extracting each block key from the block tag. The block key retrieval algorithm **B-KeyRet**, on input of the master key k_{mas} and block tag T_i , returns a block key k_i or \perp .

In BL-MLE, block tags are also used to generate Proof-of-Ownership (PoW) tags. **PoWPrf** is an algorithm to produce a proof prf for a given file M and a challenge Q . **PoWPrf** is the corresponding algorithm to verify the validity of prf with Q and M .

2.6 | Proof-of-ownership

Proof-of-Ownership (PoW) [15] is an interactive protocol $\Pi = (P, V)$ between a prover P and a verifier V . It seeks to verify whether V owns a complete file. As early deduplication schemes relied on hash-as-a-proof to authenticate file ownership, the adversary can easily get such hash value elsewhere and cheat to pass the verification. By adopting PoW to authenticate the file ownership, the adversary cannot efficiently pass the verification without actually possessing the entire file. Merkle Hash Tree (MHT) is often used as a basic tool to achieve PoW. MHT works by inputting the hash of each file block as a leaf node and iteratively computing to derive the merkle root of MHT. In a MHT-based PoW protocol, V can randomly challenge some nodes, and P is supposed to return the challenged nodes as well as the corresponding sibling paths to V . Later, V can derive a merkle root R' based on the proof and compare it with the right one R . Due to the soundness of MHT, it is hard for P to forge a proof to pass the verification.

2.7 | Chameleon hash

Chameleon Hash $\text{CH} = (\text{Setup}_{\text{CH}}, \text{KeyGen}_{\text{CH}}, \text{Hash}_{\text{CH}}, \text{Forge}_{\text{CH}})$ is a trapdoor one-way hash function where it is hard to find a collision (s.t. $\text{Hash}_{\text{CH}}(m, r) = \text{Hash}_{\text{CH}}(m', r')$) without using the trapdoor key. Briefly, CH can be described as follows:

Setup_{CH}: This algorithm takes as input a security parameter λ and returns the system parameters P .

KeyGen_{CH}: This algorithm takes as input P and returns a hash/trapdoor (public/secret) key pair (hk, tk) .

Hash_{CH} : This algorithm takes as input a message m , hk , and a randomness r . It returns a hash value h .

Forge_{CH} : This algorithm takes as input a tuple (m, r, h) , a trapdoor key tk and a new message m' , and returns a new randomness r' .

Generally, a secure CH scheme is required to satisfy correctness, indistinguishability, collision-resistance and key-exposure freeness [16]. Here, indistinguishability is also known as semantic security. It is a preliminary security property for an insecure public-key encryption scheme. CH is a basic tool to build many cryptographic primitives. For example, one can build a practically secure (Indistinguishability under Adaptive Chosen Ciphertext Attack, IND-CCA2) encryption scheme based on CH [17].

3 | DEFINITIONS

In this section, we give the system model and security requirements of our proposed BDAI.

3.1 | System model

The framework of our BDAI is shown in Figure 1. It consists of three parties: the cloud server, data owner and deduplication user. We introduce each party as follows:

Data Owner (U_{own}): The one who first uploads the data file from the local-side to the cloud server for outsourcing. To acquire the outsourcing service, U_{own} pays the cloud server an outsourcing fee sum_m via Bitcoin. The sum_m is used to cover the expenditures for maintaining the outsourced files. We assume that the data owner will honestly and correctly perform outsourcing as defined in Section 4.3; otherwise, our scheme is required to achieve tag consistency security (see the discussion in Section 3.2).

Deduplication User (U_{dedup}): The one who attempts to upload a file that already exists on the server. Due to the deduplication, U_{dedup} only needs to transmit some metadata instead of the entire file. However, to acquire the deduplication, U_{dedup} needs to hold the entire file during the PoW protocol [15]. U_{dedup} also pays the data owner with deduplication reward sum_m' via Bitcoin. This reward sum_m' is used to compensate the sum_m paid by the data owner previously for outsourcing.

Cloud Server: The cloud server has unlimited space and computing power due to economies of scale. It offers an outsourcing service for the data owner to keep his file remotely at some costs. In addition, it offers deduplication service for the deduplication user who tries to upload an existing file on the server. The cloud server is honest but curious about the contents of stored files. It provides services as negotiated but tries to learn the contents of an encrypted file on the server. What's more, disputes may occur regarding the original source of the file or the validity of payment (because Bitcoin payment will not be confirmed immediately). To settle disputes, the server will host a dispute arbitration protocol for settlement.

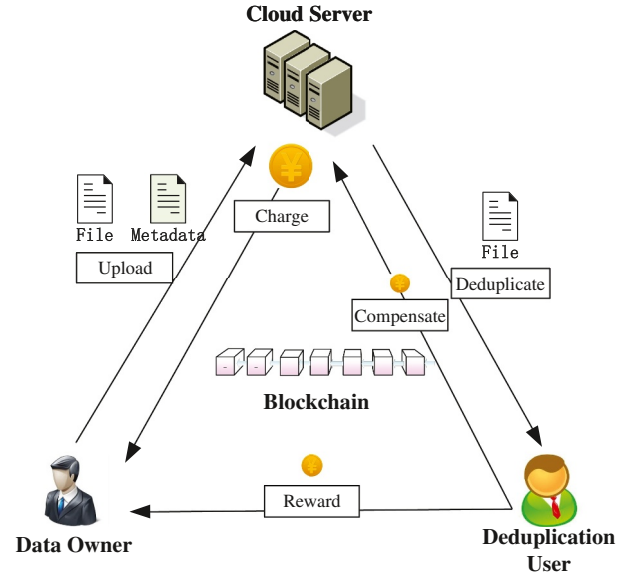


FIGURE 1 The framework of blockchain-based deduplication with arbitration and incentives (BDAI)

To emphasise, the server is assumed to honestly launch the protocol as negotiated because it is of its interests to settle disputes and prosper the deduplication business.

3.2 | Security requirements of BDAI

Definition 2 A secure BDAI should satisfy the following properties:

Correctness. Our proposed BDAI satisfies correctness for verification.

Confidentiality. Our BDAI implements BL-MLE [7] for encryption. Since BL-MLE cannot achieve any conventional semantic security, we ask for the best security model by assuming the file block to be unpredictable. Our proposed BDAI is secure if no PPT adversary could efficiently distinguish between the encryptions of an unpredictable block and a random string of the same length [7].

PoW Security. Our proposed BDAI is secure if no PPT adversary \mathcal{A} could efficiently cheat to pass our PoW protocol without storing the entire file.

Indistinguishability of Chameleon Hash. Our proposed BDAI is secure if no PPT adversary \mathcal{A} could efficiently distinguish between two randomly generated chameleon hash outputs [16].

Soundness of DU Protocol. Our proposed BDAI satisfies DU arbitration soundness if the dispute arbitration protocol among users is sound. It means no PPT adversary can pass the verification of the DU protocol efficiently without holding the private key of the data owner.

Soundness of DSU Protocol. Our proposed BDAI satisfies DSU arbitration soundness if the dispute arbitration protocol between the server and the user is sound. It means no PPT adversary can pass the verification of DSU protocol efficiently without holding the private key of the server.

Our proposal cannot satisfy the tag consistency property (or its stronger version) as suggested in Zhao and Chow's work [18]. The reason is: the master key and corresponding file ciphertext are generated secretly at the user side prior to outsourcing. No solution is yet found to test the consistency of ciphertext with its metadata, especially without breaking the confidentiality security in our setting. Though equality test (in Section 4.4) is given to detect the inconsistency of metadata, it cannot basically prevent the adversary from randomly forging ciphertext (and its metadata) to perform a duplicate faking attack as pointed out by Zhao and Chow's study [18]. So, in this work, we assume the data owner to be honest and the outsourcing (in Section 4.3) is executed correctly at all times. See countermeasures against the duplicate faking attack for our scheme in Section 4.10.

4 | THE PROPOSED BDAI SCHEME

The detailed construction of our proposed BDAI is given as follows.

4.1 | Setup

To setup the system, the system manager chooses a security parameter λ . Then, it chooses two groups G and G_T of prime order q and generator g . It sets bilinear map as $\hat{e}: G \times G \rightarrow G_T$, and hash functions as: $H_1: \{0,1\}^* \rightarrow Z_q$, $H_2: \{0,1\}^* \rightarrow G$, $H_3: \{Z_q\}^s \rightarrow G$, $H_4: G \rightarrow \{Z_q\}^s$. It randomly picks s elements $\{u_1, u_2, \dots, u_s\} \xleftarrow{R} G$, and outputs $\text{param}_{\text{BDAI}} = \{G, G_T, g, q, \hat{e}, H_1, H_2, H_3, H_4, s, u_1, u_2, \dots, u_s\}$ as system parameters.

4.2 | KeyGen

To generate the public and private key pair for cloud server, select a random number $x_s \xleftarrow{R} Z_q^*$ as the server's private key, and compute $y_s = g^{x_s}$ as the server's public key.

Similarly, for each user, select a random number $x \xleftarrow{R} Z_q^*$ as the private key, and compute $y = g^x$ as the user's public key. In the following, denote (x_o, y_o) as the data owner U_{own} 's key pair and (x_d, y_d) as the deduplication user U_{dedup} 's key pair.

4.3 | Pre-Processing for outsourcing

Before outsourcing, the data owner U_{own} pre-processes file m as follows:

1. U_{own} splits the file m into n blocks to derive $\tilde{m} = m_1 || \dots || m_n$ where each $m_i \in \{Z_q\}^s$. Then, U_{own} computes the master key $k_0 = H_1(\tilde{m})$ and file tag $\sigma_0 = g^{k_0}$. Then, compute a proof of knowledge $\text{PoK}(\sigma_0)$.
2. U_{own} transfers an outsourcing fee to the server via Bitcoin. Denote $\text{tran}_{\tilde{m}}$ as the transaction identifier; $\text{sum}_{\tilde{m}}$ as the

transferred money; $t_{\tilde{m}}$ as the time point for confirmation of the transaction $\text{tran}_{\tilde{m}}$; $\mathcal{B}_{\text{tran}_{\tilde{m}}}$ as the hash of the block which contains $\text{tran}_{\tilde{m}}$.

3. U_{own} selects a customised identity $\text{CID} \in \{0,1\}^*$ and computes coefficients $e = H_1(\text{CID})$ and $h = g^e$. Then, U_{own} computes $r_1 = (r_{1,0}, r_{1,1}) = (\sigma_0, y_o^{k_0})$ and chameleon hash $\tilde{h}_1 = \hat{e}(\sigma_0, g) \cdot \hat{e}(h \cdot y_o, \sigma_0)$. To explain, the tuple (\tilde{h}_1, r_1) is an arbitration tag used to settle disputes between deduplication users regarding the originality of a file (as will be discussed in Section 4.8).
4. U_{own} sets $\alpha = H_1(\mathcal{B}_{\text{tran}_{\tilde{m}}} || t_{\tilde{m}})$. Then, U_{own} computes $r_2 = (r_{2,0}, r_{2,1}) = (g^\alpha, y_s^\alpha)$ and chameleon hash $\tilde{h}_2 = \hat{e}(r_{2,0}, g) \cdot \hat{e}(h \cdot y_s, r_{2,0})$. To explain, the tuple (\tilde{h}_2, r_2) is an arbitration tag used to settle disputes between the server and deduplication user regarding the payment of deduplication reward (as will be discussed in Section 4.9).
5. For each i , compute each block key $k_i = H_3(m_i)$ and block ciphertext $c_i = H_4(k_i) \oplus m_i$. Split each c_i into s sectors such that $c_i = \{c_{ij}\}_{1 \leq j \leq s}$ and $c_{ij} \in Z_q$. For each i , compute block tag $\sigma_i = (k_i \prod_{j=1}^s u_j^{c_{ij}})^{k_0}$ and block auxiliary information $\text{aux}_i = \hat{e}(k_i, \sigma_0)$. Denote $c = c_1 || \dots || c_n$ as file ciphertext; $\sigma = \{\sigma_i\}_{0 \leq i \leq n}$ as tag set; $\text{aux} = \{\text{aux}_i\}_{1 \leq i \leq n}$.

The data owner U_{own} outsources the encrypted file together with metadata $C = \{c || \text{meta}\} = \{c, \sigma, \tilde{h}_1 || \tilde{h}_2 || \text{aux} || \text{PoK}(\sigma_0)\}$ to the server. After confirming the Bitcoin payment $\text{tran}_{\tilde{m}}$, the server keeps $\{c, \sigma, (\tilde{h}_1, r_1) || (\tilde{h}_2, r_2), \text{tran}_{\tilde{m}}, \mathcal{B}_{\text{tran}_{\tilde{m}}}, \text{sum}_{\tilde{m}}, t_{\tilde{m}}\}$ at local storage. $\{(\tilde{h}_1, r_1) || (\tilde{h}_2, r_2), \text{tran}_{\tilde{m}}, \mathcal{B}_{\text{tran}_{\tilde{m}}}, \text{sum}_{\tilde{m}}, t_{\tilde{m}}\}$ will be used to settle potential disputes in the future. After passing the Deduplication&Equality Test (in Section 4.5) and Proof of Ownership (in Section 4.7), $\{\mathcal{B}_{\text{tran}_{\tilde{m}}}, \text{sum}_{\tilde{m}}, t_{\tilde{m}}\}$ (already recorded on blockchain) and $\{\text{aux}\}$ (only for consistency check) will be discarded from the server storage to reclaim space.

4.4 | Consistency check

To ensure the consistency of block tag with block ciphertext, for each $i \in [1, n]$, the server checks whether

$$\hat{e}(\sigma_i, g) \stackrel{?}{=} \text{aux}_i \cdot \hat{e}\left(\prod_{j=1}^s u_j^{c_{ij}}, \sigma_0\right)$$

If not hold, block tag inconsistency is found, reject and terminate deduplication; if all hold, proceed to the next stage.

4.5 | Deduplication & Equality Test

For file-level deduplication, the equality test is based on the direct search of file tag. For example, for a given file m_0

with file tag $\sigma_0 = g^{k_0} = g^{H_1(\tilde{m}_0)}$, the user simply uploads $\sigma_0 \parallel \text{PoK}(\sigma_0)$ to the server as the deduplication request. The server checks the validity of $\text{PoK}(\sigma_0)$ first. To avoid reuse of old $\text{PoK}(\sigma_0)$, a table is required to record all $\text{PoK}(\sigma_0)$ sent to server to resist reply attack. After passing the check, the server proceeds to search the file corresponding to σ_0 in its storage. If it found, it would jump to proof-of-ownership in Section 4.7; otherwise, it aborts and terminates.

For block-level deduplication, equality test is based on pairing computation. For two distinct files m and m' with corresponding file tags σ_0 and σ'_0 respectively, denote m_i and m'_i as blocks from m and m' respectively. Run the following equation to check the block equality:

$$\widehat{e}(\sigma_0, \sigma'_i) = \widehat{e}(\sigma'_0, \sigma_i)$$

If it holds, two blocks are identical; otherwise, they are not.

4.6 | Decryption

For each $i \in [1, n]$, given block tag σ_i and master key k_0 , compute block key as: $k_i = \sigma_i^{k_0^{-1}} \cdot \left(\prod_{j=1}^s u_j^{c_{ij}} \right)^{-1}$. Then, for each i , compute block plaintext $m_i = H_4(k_i) \oplus c_i$. Finally, derive file plaintext as: $m = m_1 \parallel \dots \parallel m_n$.

4.7 | Proof-of-ownership (PoW)

To authenticate the file ownership instead of hash-as-a-proof [15], the deduplication user U_{dedup} and the server engage in an interactive protocol called: Proof-of-Ownership (PoW) as follows:

Dedup.Challenge: Upon receiving σ_0 from U_{dedup} , the server picks a subset of E out of $[1, n]$, that is, $E \subseteq [1, n]$ and returns a deduplication challenge $\text{Pow.chal} = (E)$ to U_{dedup} .

Dedup.Prove: On receiving Pow.chal , the user processes file m to derive encrypted file c with Pow.chal as defined in Step 1-4 of Section 4.3. Then, U_{dedup} computes $H_2(c_i)$ for each $i \in E$ to construct a merkle hash tree Ω' . Denote $L = \{H_2(c_i)\}_{i \in E}$ as a set of leaf nodes where each node is the hash of ciphertext c_i . Denote $S = \{s_i\}_{i \in E}$ as a set of sibling paths from the challenged leaf nodes L to the root of Ω' . In addition, U_{dedup} transfers a deduplication fee $\text{sum}_m^{-'}$ (e.g. 1% of the outsourcing fee sum_m^{-}) to the server. Denote $\text{tran}_m^{-'}$ as the transaction of payment $\text{sum}_m^{-'}$. Then, U_{dedup} sends the deduplication proof $\text{Pow.prf} = \{L, S, \text{tran}_m^{-'}\}$ to the user.

Dedup.Verify: On receiving Pow.prf , the server uses L and S to construct a merkle hash tree Ω'' and computes the root R'' . Then, the server checks whether $R = R''$ where R is the root of the original file. If it holds, the server proceeds to check the validity of $\text{tran}_m^{-'}$ on the Bitcoin blockchain. If the payment is valid, the server shall grant U_{dedup} with direct access to the file c ; otherwise, it aborts and terminates.

4.8 | Dispute arbitration protocol between deduplication users (DU)

Suppose two users (one is the data owner U_{own} , the other is the deduplication user U_{dedup}) claim to be the owner of the file c^* (i.e. one who first outsourced c^* to the server). Denote the outsourced file with metadata on the server as $C = \{c^* \parallel \text{meta}^*\} = \{c^*, \text{CID}^*, h_1^*, h_2^*, R^*, \sigma_0^*, \text{tran}_m^*, \mathcal{B}_{\text{tran}_m^*}^*, \text{sum}_m^*, t_m^*\}$. The dispute protocol is defined as follows:

DU.Challenge: To arbitrate the original ownership of file c^* between U_{own} and U_{dedup} , the server accesses its storage for $C = \{c^* \parallel \text{meta}^*\}$ and generates a dispute challenge as:

$$\text{DU.chal} = \{\text{CID}^*, h_1^*, r_1^*, \mathcal{B}_{\text{tran}_m^*}^*, t_m^*\}.$$

DU.Prove: On receiving DU.chal , parse $r_1^* = (r_{1,0}^*, r_{1,1}^*) = (\sigma_0^*, y_o^{k_0^*})$ where k_0^* is the master key of c^* . The data owner U_{own} first computes $e^* = H_1(\text{CID}^*)$. Then, U_{own} uses his private key x_o to compute new chameleon randomness $r_1'^* = (r_{1,0}'^*, r_{1,1}'^*) = r_{1,0}^* \cdot g^{(x_o + e)^{-1} H_1(m^*) - H_1(h_1^* \parallel r_1^* \parallel \mathcal{B}_{\text{tran}_m^*}^* \parallel t_m^*)}$. Output $\text{DU.prf} = r_1'^*$ as dispute proof.

DU.Verify: On receiving DU.prf , compute $e^* = H_1(\text{CID}^*)$ and $b^* = g^{e^*}$. Then, check whether Equation (1) holds:

$$h_1^* \stackrel{?}{=} \widehat{e}(g, g)^{H_1(h_1^* \parallel r_1'^* \parallel \mathcal{B}_{\text{tran}_m^*}^* \parallel t_m^*)} \cdot \widehat{e}(b^* \cdot y_o, r_{1,0}'^*) \quad (1)$$

Then, check whether $\widehat{e}(r_{1,0}'^*, y_o) = \widehat{e}(r_{1,1}'^*, g)$. If both hold, output 1; otherwise, output 0.

The user who returns a valid proof to pass DAU.Verify wins the dispute arbitration and is deemed as the data owner of the file c^* . To convince the public, the server can publish $(\text{DU.chal}, \text{DU.prf})$. Since anyone can verify it by running Equation (1) and checking with $(\mathcal{B}_{\text{tran}_m^*}^*, t_m^*)$ on Bitcoin blockchain, the above arbitration is publicly verifiable.

4.9 | Dispute arbitration protocol between the server and the user (DSU)

To arbitrate whether the server rewards data owner U_{own} with compensation (i.e. $\text{sum}_m^{-'}$) as negotiated, the server and the data owner engage in a DSU protocol as follows. Here, we assume each deduplication user U_{dedup} shall send deduplication reward $\text{sum}_m^{-'}$ to the server, then the server relays it to data owner U_{own} . This prevents leaking the identity of the data owner to deduplication user directly.

DSU.Challenge: To arbitrate the validity of a payment made by the server to the user, the server accesses its storage

for $C = \{c^* \| meta^*\}$ and generates a dispute challenge as: $DSU.chal = \{CID^*, h_1^*, r_1^*, B_{tran_m}^*, t_m^*\}$.

DSU.Prove: On receiving $DSU.chal$, parse $r_2^* = (r_{2,0}^*, r_{2,1}^*) = (g^{\alpha^*}, \gamma_s^{\alpha^*})$. The server first computes $e^* = H_1(CID^*)$, $\alpha^* = H_1(B_{tran_m}^* \| t_m^*)$ and $\beta^* = H_1(h_2^* \| r_2^* \| B_{tran_m}^* \| t_m^*)$. Denote $B_{tran_m}^*$ as the block hash of the corresponding transaction; t_m^* as the confirmation time of the payment. Then, the server uses its private key x_s to compute new chameleon randomness $r_2'^* = (r_{2,0}'^*, r_{2,1}'^*) = (r_{2,0}^* \cdot g^{(x_s+e^*)^{-1}(\alpha^*-\beta^*)}, r_{2,1}^* \cdot \gamma_o^{(x_s+e^*)^{-1}(\alpha^*-\beta^*)})$. Output a dispute proof $DSU.prf = r_1'^*$.

DSU.Verify: On receiving $DSU.prf$, compute $e^* = H_1(CID^*)$ and $h^* = g^{e^*}$. Then, check whether Equation (2) holds:

$$h_2^* = ? \widehat{e}(g, g)^{\beta^*} \cdot \widehat{e}(h^* \cdot \gamma_s, r_{2,0}'^*) \quad (2)$$

Then, check whether $\widehat{e}(r_{2,0}'^*, \gamma_o) = \widehat{e}(r_{2,1}'^*, g)$. If both hold, output 1; otherwise, output 0.

If the server successfully passes **DASU.Verify**, it means the server has compensated data owner U_{own} as negotiated. To convince the public, the server can publish $(DSU.chal, DSU.prf)$. Since anyone can verify it by running Equation (2) and checking with $(B_{tran_m}^*, t_m^*)$ recorded on Bitcoin blockchain, arbitration is publicly verifiable.

4.10 | Further discussion

According to Zhao and Chow's work [18], our scheme cannot prevent adversary from performing random forgery on the ciphertext and metadata to be outsourced, that is, does not achieve tag consistency. The use of equality test (defined in Section 4.4 and in Chen et al's BL-MLE [7]) only guarantees the consistency of file tag, block tag and auxiliary information (instead of the file ciphertext). However, it is still possible to detect and prevent duplicate faking attack. For example, the deduplication user who failed to decrypt a forged file resulted from the duplicate faking attack [18] could report to a trusted third party (or employ our DU protocol defined in Section 4.8 for arbitration). Once the number of reports reach a certain number (or our DU protocol is won by a deduplication user instead of the owner), the server should revoke the outsourcing (or deduplication) of designated file accordingly. Also, from the economic perspective, it is not beneficial for the data owner to outsource the forged file as no rewards will be gained if being reported (or even suffer from punishment). Similarly, the server will honestly execute the deduplication procedures since it generally reduces the costs of outsourcing services.

5 | THE SECURITY ANALYSIS OF BDAI

We give security analysis based on security requirements defined in section 3.2. Due to the space limitation, we omit discussing the security of equality test and tag consistency. Refer to [7] for more details.

5.1 | Correctness

The correctness of Equation (1) is elaborated as follows.

$$\begin{aligned} h_1^* &= \widehat{e}(\sigma_0^*, g) \cdot \widehat{e}(h^* \cdot \gamma_o, \sigma_0^*) \\ &= \widehat{e}(g, g)^{k_0^*} \cdot \widehat{e}(h^* \cdot \gamma_o, \sigma_0^*) \\ &= \widehat{e}(g, g)^{H_1(m^*) + H_1(h_1^* \| \dots \| t_m^*) - H_1(h_1^* \| \dots \| t_m'^*)} \\ &\quad \cdot \widehat{e}(h^* \cdot \gamma_o, r_{1,0}^*) \\ &= \widehat{e}(g, g)^{H_1(h_1^* \| r_1^* \| B_{tran_m}^* \| t_m^*)} \cdot \widehat{e}(h^* \cdot \gamma_o, r_{1,0}^*) \\ &\quad g^{(x_o+e^*)^{-1}H_1(m^*) - H_1(h_1^* \| r_1^* \| B_{tran_m}^* \| t_m^*)} \\ &= \widehat{e}(g, g)^{H_1(h_1^* \| r_1^* \| B_{tran_m}^* \| t_m^*)} \cdot \widehat{e}(h^* \cdot \gamma_o, r_{1,0}'^*) \end{aligned}$$

The correctness of Equation (2) is elaborated as follows:

$$\begin{aligned} h_2^* &= \widehat{e}(r_{2,0}^*, g) \cdot \widehat{e}(h^* \cdot \gamma_s, r_{2,0}^*) \\ &= \widehat{e}(g^{\alpha^*}, g) \cdot \widehat{e}(h^* \cdot \gamma_s, g^{\alpha^*}) \\ &= \widehat{e}(g, g)^{\beta^*} \cdot \widehat{e}(g^{(x_s+e^*)}, g^{\alpha^*}) \\ &\quad \widehat{e}(g, g)^{(\alpha^*-\beta^*)(x_s+e^*)^{-1}(x_s+e^*)} \\ &= \widehat{e}(g, g)^{\beta^*} \cdot \widehat{e}(h^* \cdot \gamma_s, r_{2,0}^* \cdot g^{(x_s+e^*)^{-1}(\alpha^*-\beta^*)}) \\ &= \widehat{e}(g, g)^{\beta^*} \cdot \widehat{e}(h^* \cdot \gamma_s, r_{2,0}'^*) \end{aligned}$$

5.2 | Confidentiality of BDAI

Similar to MLE and BL-MLE, our BDAI cannot achieve any semantic security. However, following Bellare et al.'s [5] and Chen et al.'s [7] ideas, we can prove that our encryption is indistinguishable from a random string in equal size under the assumption of unpredictable block source (defined in Section 2.4). We denote this security as **PRV\$-CDA-B** [7]. Following the model of BL-MLE given in part B of Section II in ref. [7], we give security analysis as follows.

Assume \mathcal{A} is an adversary who can efficiently break our **PRV\$-CDA-B** security. Then, we can construct an algorithm \mathcal{B} to efficiently solve CDHP. We give a proof sketch as follows: On given a random instance of CDHP $g_0, g_0^a, g_0^b \in G$, the

algorithm \mathcal{B} is supposed to output $g_0^{ab} \in G$. \mathcal{B} simulates the challenger and interacts with the adversary \mathcal{A} as follows.

First, \mathcal{A} sends the description of an unpredictable block-source \mathcal{M} to \mathcal{A} . To generate system parameters $\text{param}_{\text{BDAI}}$, \mathcal{B} sets $g = g_0^a$. For all $k \in \{1, s\}$, \mathcal{B} randomly chooses $r_k \leftarrow^R Z_q$ and computes $u_k = g_0^{a \cdot r_k}$. Then, \mathcal{B} returns $\text{param}_{\text{BDAI}} = \{G, G_T, g, q, \hat{e}, H_1, H_2, H_3, H_4, s, u_1, u_2, \dots, u_s\}$ to \mathcal{A} . \mathcal{B} controls H_1, H_2, H_3, H_4 as random oracles and answers to each distinct hash query with a different response. \mathcal{B} always returns the same response to the same query (by maintaining and checking a query history). During the Challenge stage, given the challenged file M , adversary \mathcal{A} would not query the hash function about it due to the unpredictable block-source \mathcal{M} .

\mathcal{B} sets the master key as $H_1(M) = a^{-1}$ and file tag as $\sigma_0 = g^{a^{-1}} = g_0$. Then, \mathcal{B} picks $\alpha_j \leftarrow Z_q$, and $c_j \leftarrow^R \{Z_q\}^s$ randomly for each M_j . Partition c_j into s sectors $c_{j1}, c_{j2}, \dots, c_{js} \in Z_q$ and compute the block tag as $\sigma_j = g_0^{\alpha_j b} \prod_{k=1}^s g_0^{r_k c_{jk}} = (g_0^{\alpha_j ab} \prod_{k=1}^s u_k^{c_{jk}})^{-1}$. As observed that the block key $k_j = H_3(M_j) = g_0^{\alpha_j ab}$, σ_j is thus consistent with the block ciphertext c_j . Finally, after \mathcal{A} outputs its guess, \mathcal{B} picks a random tuple $\langle m_i, k_i, h_i \rangle$ from H_3 query list, chooses $j \leftarrow^R [1, n(\lambda)]$ and outputs $k_j^{\alpha_j^{-1}}$ as the solution to the given instance of CDHP.

Accordingly, the adversary \mathcal{A} 's view is identical to the original PRV\$-CDA-B game. We can apply the min-entropy of the block-source \mathcal{M} to bound the probability of \mathcal{A} in solving the CDHP. Refer to ref. [7] for more details.

5.3 | PoW security

Unlike BL-MLE, which utilises block tag as PoW tag, our BDAI utilises hash of ciphertext as PoW tag. Suppose \mathcal{A} is an efficient PPT adversary against our PoW security, we briefly show that it is reducible to breaking the soundness of underlying MHT, collision resistance of cryptographic hash function, or double-spending resistance of Bitcoin.

On a given challenge $\text{Pow.chal} = (E)$, adversary \mathcal{A} is supposed to output a proof $\text{Pow.prf} = \{L, S, \text{tran}_m^{\sim}\}$ to pass the verification of the **Dedup.Verify**. Denote $\text{Pow.prf}^* = \{L^*, S^*, \text{tran}_m^{\sim*}\}$ as the valid proof. If $\text{Pow.prf} \neq \text{Pow.prf}^*$ and both proofs can pass the verification, we do a case analysis as follows:

Case 1 $L \neq L^*$.

Parse $L^* = \{H_2(c_i^*)\}_{i \in E}$ as the correct set of challenged block ciphertext. Parse $L = \{H_2(c_i)\}_{i \in E}$ as a set forged by \mathcal{A} . Here, \mathcal{A} does not possess a complete file m (e.g. missing one block). However, due to unpredictable block source \mathcal{M} , \mathcal{A} 's probability in guessing the missing block(s) is negligible. If $S \neq S^*$, this implies breaking the collision resistance of the underlying cryptographic hash function [15]. If $S = S^*$, as discussed in [15], the probability of adversary \mathcal{A} in passing MHT verification in the **Dedup.Verify** is defined as: $\Pr(\text{succ}_{\text{MHT}}) = 1 - \epsilon$

$(1 - p)^c$. This can be bounded by $\lceil \frac{\lambda \ln 2}{\epsilon(1-p)} \rceil$ with particular choice on ϵ (denoted as the success probability of guessing a random block). However, ϵ is negligible under the assumption of the unpredictable block source in our case. Hence, $L = L^*$.

Case 2 $S \neq S^*$.

Parse $S^* = \{S_i^*\}_{i \in E}$ as the set of sibling paths from challenged leaf node to the root of Ω'' (i.e. R''). Denote Ω'' as the correct merkle hash tree and R'' as the corresponding root. If $S \neq S^*$ and both Pow.prf and Pow.prf^* could pass the verification, this implies breaking the soundness of MHT and the collision resistance of the underlying cryptographic hash function [15]. Hence, $S = S^*$.

Case 3 $\text{tran}_m^{\sim'} \neq \text{tran}_m^{\sim*}$.

If the same set of coins have been spent in two different transactions, this implies a double-spending case. As generally concluded, double-spending is resisted by the Proof-of-Work consensus [19] mechanism employed by the Bitcoin system. This is against the soundness of Bitcoin's Proof-of-Work consensus protocol. Hence, $\text{tran}_m^{\sim'} = \text{tran}_m^{\sim*}$. Therefore, we can conclude that no PPT adversary can efficiently forge a valid $\text{Pow.prf} \neq \text{Pow.prf}^*$ to pass the PoW verification without possessing the entire file m .

5.4 | Indistinguishability of Chameleon Hash

Indistinguishability (semantic) security is a basic requirement for any CH to satisfy. Based on this property, other security properties (e.g. collision-resistance, key-exposure freeness) can be derived [16]. Due to space limitation, we only give the security analysis of indistinguishability for our CH in this work. The proof of other security properties for our CH follows similar proof techniques given in Section 6 of ref. [20].

To analyse, two chameleon hash and randomness pairs are generated as arbitration tags during our Pre-Processing stage: (h_1, r_1) and (h_2, r_2) . They are kept by the server to settle disputes between users (as defined in Section 4.8), or the server and user (as defined in Section 4.9). To explain:

- h_1 is committed to the file message m to be deduplicated (with file tag $\sigma_0 = g^{H_1(\tilde{m})}$) and owner's public key y_o . Only server (who holds corresponding private key x_o) can efficiently find a collision of h_1 such that $h_1 = \text{Hash}_{\text{CH}}(m, r_1) = \text{Hash}_{\text{CH}}(m', r'_1)$.
- h_2 is committed to $\alpha = H_1(\mathcal{B}_{\text{tran}_m^{\sim}} \| t_m^{\sim})$ (the payment of deduplication fee) and server's public key y_o . Only the server (who holds corresponding private key x_s) can efficiently find a collision for h_2 .

The indistinguishability property asks that no efficient adversary \mathcal{A} could distinguish between (h_1, r_1) and (h_2, r_2) with non-negligible advantage under the assumption of

unpredictable block source. The collision-resistance property further asks that no efficient adversary \mathcal{A} could forge a fresh collision for a given tuple (h_b, r_b) where $b = \{1, 2\}$ under the assumption of the unpredictable block source without holding the corresponding private key (i.e. x_o and x_s respectively). The key-exposure freeness specifically asks that no efficient adversary \mathcal{A} could efficiently forge a fresh collision of his choice.

Following the indistinguishability model given in Section 2.2 in ref. [16], we prove the indistinguishability between (h_1, r_1) and (h_2, r_2) via game hopping as follows. To emphasise, our security analysis is given under the assumption of unpredictable source.

Suppose \mathcal{A} is a PPT adversary who can efficiently distinguish between (h_1, r_1) and (h_2, r_2) where $h_1 = \widehat{e}(r_{1,0}, g) \cdot \widehat{e}(h \cdot y_o, r_{1,1})$ and $h_2 = \widehat{e}(r_{2,0}, g) \cdot \widehat{e}(h \cdot y_s, g^{H_1(\mathcal{B}_{tran_m} || t_m)})$. We prove by game hopping as follows:

- **Game 0:** This is the original indistinguishability game for our BDAI.

1. **Setup:** The adversary \mathcal{A} first generates an unpredictable file source \mathcal{M} and sends it to the simulator \mathcal{S} . Upon receiving \mathcal{M} , the simulator \mathcal{S} runs \mathcal{M} to generate message M and auxiliary information Z . Then, it runs the **Setup** algorithm to derive $\text{param}_{\text{BDAI}} = \{G, G_T, g, q, \widehat{e}, H_1, H_2, H_3, H_4, s, u_1, u_2, \dots, u_s\}$. Run the **KeyGen** algorithm to derive (x_o, y_o) and (x_s, y_s) for the data owner U_{own} and the server, respectively. Then, \mathcal{S} selects a $\text{CID} \leftarrow^R \{0, 1\}^*$ and relays $(\mathcal{M}, \{G, G_T, g, q, \widehat{e}, H_1, H_2, H_3, H_4, s, u_1, u_2, \dots, u_s\}, \text{CID}, y_o, y_s)$ to \mathcal{A} .

2. **Challenge:** \mathcal{S} flips a coin $b \leftarrow^R \{0, 1\}$ and derives h_b differently.

For $b = 0$, compute:

$$r_1 = (r_{1,0}, r_{1,1}) = (g^{H_1(M)}, y_o^{H_1(M)}), \quad h = g^{H_1(\text{CID})},$$

$$h_1 = \widehat{e}(g, g)^{H_1(M)} \cdot \widehat{e}((h \cdot y_o)^{H_1(M)}, g).$$

For $b = 1$, compute:

$$\alpha = H_1(\mathcal{B}_{tran_m} || t_m), \quad r_2 = (r_{2,0}, r_{2,1}) = (g^\alpha, y_s^\alpha),$$

$$h_2 = \widehat{e}(g, g)^\alpha \cdot \widehat{e}((h \cdot y_s)^\alpha, g).$$

Then, \mathcal{S} relays (h_b, r_b, Z) to adversary \mathcal{A} . Due to the unpredictability of source \mathcal{M} , \mathcal{A} is not allowed to query the challenged file M . Suppose \mathcal{A} issues at most q_s distinct queries to oracle H_1 (controlled by \mathcal{S}) on e_i and M_i , \mathcal{S} will return the answer to each distinct query accordingly.

3. **Output:** The adversary \mathcal{A} outputs $a \in \{0, 1\}$ and wins if $b = a$.

- **Game 1:** The same as **Game 0** except that we randomly sample $R_1 \in G$ and use it to compute h_1 . Thus, (h_1, r_1) is computed as follows: (Note: adaptations are highlighted in boxes.)

$$r_1 = (g^{H_1(M)}, y_o^{H_1(M)}), \quad R_1 \in_R G,$$

$$h_1 = \widehat{e}(g, g)^{H_1(M)} \cdot \widehat{e}(b^{H_1(M)} \cdot R_1, g).$$

(h_2, r_2) is computed as follows:

$$\alpha = H_1(\mathcal{B}_{tran_m} || t_m), \quad r_2 = (r_{2,0}, r_{2,1}) = (g^\alpha, y_s^\alpha),$$

$$h_2 = \widehat{e}(g, g)^\alpha \cdot \widehat{e}((h \cdot y_s)^\alpha, g).$$

- **Game 2:** The same as **Game 1** except that we randomly sample $R_2 \in G$ and use it to compute h_1 . So, (h_0, r_0) is computed as follows: (Note: adaptations are highlighted in boxes.)

$$r_1 = (g^{H_1(M)}, y_o^{H_1(M)}), \quad R_1 \in_R G,$$

$$h_1 = \widehat{e}(g, g)^{H_1(M)} \cdot \widehat{e}(b^{H_1(M)} \cdot R_1, g).$$

(h_2, r_2) is computed as follows:

$$\alpha = H_1(\mathcal{B}_{tran_m} || t_m), \quad r_2 = (g^\alpha, y_s^\alpha), \quad R_2 \in_R G,$$

$$h_2 = \widehat{e}(g, g)^{H_1(M)} \cdot \widehat{e}(b^\alpha \cdot R_2, g).$$

Since there are only negligible changes to each game hop, the adversary \mathcal{A} cannot distinguish between the games; otherwise, we can construct an algorithm \mathcal{B} to solve the DDHP assuming adversary \mathcal{A} can distinguish between **Game 0** and **Game 1**, or **Game 1** and **Game 2**. We briefly prove it as follows:

Take **Game 0** and **Game 1** as an example:

On given a DDHP instance $g, g^a, g^b, R \in G$, the algorithm \mathcal{B} serves as a simulator during the above game and sets the public key as $y_o = g^a$ where a is unknown. Then, \mathcal{B} generates (h_1, r_1) as follows:

$$r_1 = (g^{H_1(M)}, y_o^{H_1(M)}), \quad R_1 \in_R G,$$

$$h_1 = \widehat{e}(g, g)^{H_1(M)} \cdot \widehat{e}(b^{H_1(M)} \cdot Q, g)$$

If $Q = g^{ab}$, this implies **Game 0**; else, $Q \in_R G$, which implies **Game 1**. Thus, we can bound the \mathcal{B} 's advantage in solving the DDHP by $\text{Adv}_{\mathcal{B}}^{\text{DDHP}} = |\Pr[E_0] - \Pr[E_1]|$ via distinguishing between **Game 0** and **Game 1**. Here, we denote E_i as the event for \mathcal{A} winning in **Game i**. Analogously, we have

$\text{Adv}_{\mathcal{B}}^{\text{DDHP}} = |\Pr[E_1] - \Pr[E_2]|$ by distinguishing between **Game 1** and **Game 2**.

Due to unpredictability of source \mathcal{M} , **Game 2** generates (h_b, r_b) as a one-time pad, we have $\text{Adv}_{\mathcal{A}}^{\text{Game2}} = \frac{1}{2}$. Based on the above, we can bound the \mathcal{B} 's advantage in solving the DDHP by: $\text{Adv}_{\mathcal{A}}^{\text{IND-CH}} \leq 2 \cdot \text{Adv}_{\mathcal{B}}^{\text{DDHP}}$. Therefore, (h_1, r_1) and (h_2, r_2) are indistinguishable.

5.5 | Soundness of DU protocol

We briefly show that the soundness of our DU is reducible to the q-SDHP. Suppose \mathcal{A} is a PPT adversary against our DU. On given a challenge $\text{DU.chal} = \{\text{CID}^*, h_1^*, r_1^*, \mathcal{B}_{\text{tran}_m}^*, t_m^*\}$, \mathcal{A} forges a proof $\text{DU.prf}^{**} = r_1^{**}$ to pass the verification of **DU.Verify**. Denote $\text{DU.prf}^{*'} = r_1^{*'}$ as the valid proof. Then, we have the following equations:

$$\begin{aligned} & \widehat{e}(\sigma_0^*, g) \cdot \widehat{e}(h^* \cdot \gamma_o, \sigma_0^*) \\ &= \widehat{e}(g, g)^{H_1(h_1^* \parallel \dots \parallel t_m^*)} \cdot \widehat{e}(h^* \cdot \gamma_o, r_{1,0}^{*'}), \end{aligned} \quad (5)$$

$$\begin{aligned} & \widehat{e}(\sigma_0^*, g) \cdot \widehat{e}(h^* \cdot \gamma_o, \sigma_0^*) \\ &= \widehat{e}(g, g)^{H_1(h_1^* \parallel \dots \parallel t_m^*)} \cdot \widehat{e}(h^* \cdot \gamma_o, r_{1,0}^{**}). \end{aligned} \quad (6)$$

Next, we do a case analysis:

Case 1 $\text{DU.prf}^{*'} \neq \text{DU.prf}^{**}$.

Based on the above equations, we have $r_{1,0}^{*'} = r_{1,0}^{**}$ and $r_{1,1}^{*'} \neq r_{1,1}^{**}$. Thus, we have $\widehat{e}(r_{1,0}^{*'}, \gamma_o) = \widehat{e}(r_{1,1}^{*'}, g)$ and $\widehat{e}(r_{1,0}^{**}, \gamma_o) = \widehat{e}(r_{1,1}^{**}, g)$. Since both DU.prf^{**} and $\text{DU.prf}^{*'}$ could pass the verification, we have: $r_{1,1}^{*'} = r_{1,0}^{*'} \gamma_o$ and $r_{1,1}^{**} = r_{1,0}^{**} \gamma_o$. Following this, we have $r_{1,1}^{*'} = r_{1,1}^{**}$. However, this contradicts the conclusion derived from Equation (5) and Equation (6). Therefore, $\text{DU.prf}^{*'} = \text{DU.prf}^{**}$.

Case 2 $\text{DU.prf}^{*'} = \text{DU.prf}^{**}$.

Following this, \mathcal{A} could forge a valid DU.prf^{**} to pass the **DU.Verify** without using data owner U_{dedup} 's private key x_o . This brings us to discuss the collision-resistance of our underlying chameleon hash (i.e. $h = \widehat{e}(g, g)^{k_0} \cdot \widehat{e}(h \cdot \gamma_o, r_{1,0})$). Following the security model sketched in Section 2.2 of ref. [16] for the collision-resistance, we show that the collision-resistance of our CH is reducible to the breaking of the q-SDHP. By assuming \mathcal{A} as a PPT adversary who can break our collision-resistance, we briefly show how to construct a simulator \mathcal{B} that uses \mathcal{A} to solve the q-SDHP as follows.

On given a q-SDHP instance (g, g^x, \dots, g^{x^q}) (parse it by: (A_0, A_1, \dots, A_q) where $x \in \mathbb{Z}_p^*$). Here, $x \in \mathbb{Z}_p^*$ is unknown. We can

construct an algorithm \mathcal{B} that interacts with \mathcal{A} to derive $(c, g^{\frac{1}{x+c}})$ for some $c \in \mathbb{Z}_q^*$ as an answer to the q-SDHP as follows:

Setup: The adversary \mathcal{A} first generates an unpredictable source \mathcal{M} and sends it to \mathcal{B} . Upon receiving \mathcal{M} , \mathcal{B} runs \mathcal{M} to generate message \tilde{M} and auxiliary information Z . Then, it runs the **Setup** algorithm to generate system parameters $\text{param}_{\text{BDAI}} = \{G, G_T, g, q, \widehat{e}, H_1, H_2, H_3, H_4, s, u_1, u_2, \dots, u_s\}$ and the **KeyGen** algorithm to generate private and public key pair (x_o, y_o) for the data owner U_{own} . Then, \mathcal{S} relays $(\tilde{M}, \{G, G_T, g, q, \widehat{e}, H_1, H_2, H_3, H_4, s, u_1, u_2, \dots, u_s\}, y_o)$ to \mathcal{A} where \tilde{M} is the challenged file.

Query: Adversary \mathcal{A} issues q_s distinct queries $\{\text{CID}_i, M_i', (M_i, h, r_i)\}_{i \in [1, q_s]}$ to the H_1 oracle controlled by \mathcal{B} (suppose $q_s = q - 1$). Due to the unpredictability of the file source \mathcal{M} , \mathcal{A} is not allowed to query the challenged file \tilde{M} .

Response: For each M_i where $1 \leq i \leq q_s$, \mathcal{B} generates responses as follows: Set polynomial $f(z) = \prod_{i=1}^{q_s} (z + e_i) = \sum_{i=0}^{q_s} a_i z^i$ where a_0, \dots, a_{q_s} are randomness of polynomial $f(z)$ and $e_i = H_1(\text{CID}_i)$. Define:

$$g' = \prod_{i=0}^{q_s} (A_i)^{a_i} = g^{f(z)},$$

$$\tilde{h} = \prod_{i=1}^{q_s} (A_i)^{a_{i-1}} = g^{zf(z)} = g'^{f(z)}.$$

Next, define polynomial $f_i(z) = f(z)/(z + e_i) = \prod_{j=1, j \neq i}^{q_s} (z + e_j)$ and $f_i(z) = \sum_{j=0}^{q_s-1} (b_j z^j)$. Compute:

$$s_i = \prod_{j=0}^{q_s-1} (A_j)^{b_j} = (g')^{1/(x+e_i)} \text{ where } e_i = H_1(\text{CID}_i, y).$$

Then, \mathcal{B} can compute each r'_i for each $i \in [1, q_s]$:

$$r'_i = \left(g^{a_i} \cdot s_i^{H_1(M_i) - H_1(M_i')}, \gamma^{a_i} \cdot s_i^{x[H_1(M_i) - H_1(M_i')]} \right).$$

Since the equation $g^{H_1(M_i)}(h_i \cdot \gamma)^{a_i} = g^{H_1(M_i')}(h_i \cdot \gamma)^{a_i}$ holds for each i where $h_i = g^{e_i} = g^{H_1(\text{CID}_i, y)}$, r' is the correct response to hold each collision. \mathcal{B} replies \mathcal{A} with (r'_1, \dots, r'_{q_s}) as a response.

Output: Adversary \mathcal{A} wins by outputting $(\text{CID}, M, r, M^*, r^*, \tilde{h})$ such that:

$$\widehat{e}(g, g)^{H_1(M)} \cdot \widehat{e}(r_{1,0}^{(e+x)}, g) = \widehat{e}(g, g)^{H_1(M^*)} \cdot \widehat{e}(r_{1,0}^{*(e+x)}, g),$$

where $h = g^e = g^{H_1(\text{CID})}$, $r = (r_{1,0}, r_{1,1})$, $r^* = (r_{1,0}^*, r_{1,1}^*)$.

Next, we can parse r^* by:

$$\begin{aligned} r^* &= (g^{\alpha^*}, \gamma^{\alpha^*}) \\ &= (r_{1,0} \cdot s^{H_1(M)-H_1(M^*)}, r_{1,1} \cdot s^{x(H_1(M)-H_1(M^*))}). \end{aligned}$$

where:

$$s = \left(\frac{r_{1,0}^*}{r_{1,0}} \right)^{\frac{1}{H_1(M)-H_1(M^*)}} = (g')^{1/(x+e)} = g^{f(x)/(x+e)}.$$

Next, we can parse f by $f(z) = \gamma(z) (z + e) + \gamma_{-1}$ for some $\gamma(y) = \sum_{i=0}^{q_s-1} \gamma_i z^i$ and $\gamma_{-1} \in \mathbb{Z}_q$. Then, we can deduce by:

$$f(z)/(z+e) = \frac{\gamma_{-1}}{z+e} + \sum_{i=0}^{q_s-1} \gamma_i z^i.$$

Since $\gamma_{-1} \neq 0$ and CID has never been queried before (i.e. $\text{CID} \notin \{\text{CID}_1, \dots, \text{CID}_{q_s}\}$), $(z+e)$ cannot divide $f(z)$. Then, the algorithm \mathcal{B} derives a q-SDHP answer as follows:

$$\left(e, g^{\frac{1}{z+e}} = \left(s \cdot \sum_{i=1}^{q_s} (A_i)^{-\gamma_i} \right)^{\frac{1}{\gamma_{-1}}} \right).$$

However, q-SDHP is hard. Therefore, the soundness of our DU protocol holds.

5.6 | Soundness of DSU protocol

The soundness of our DSU is reducible to q-SDHP. Suppose \mathcal{A} is a PPT adversary against our DSU. On given a challenge $\text{DSU.chal} = \{\text{CID}^*, h_1^*, r_1^*, \mathcal{B}_{\text{trans}_m}^*, t_m^*\}$, suppose \mathcal{A} forged a proof DSU.prf to pass the verification DASU.Prove . Denote DSU.prf^{**} as the valid proof. Then, we have the following equations:

$$h_2^* \stackrel{?}{=} \widehat{e}(g, g)^{\beta^{*'}} \cdot \widehat{e}(h^* \cdot \gamma_s, r_{2,0}^{*'}). \quad (7)$$

$$h_2^* \stackrel{?}{=} \widehat{e}(g, g)^{\beta^{**}} \cdot \widehat{e}(h^* \cdot \gamma_s, r_{2,0}^{**}). \quad (8)$$

Based on Equation (7) and Equation (8), we can also do a similar case analysis as given in Section 5.5. We omit the details here.

6 | INSTANTIATION & PRICING

This section gives instantiation of our BDAI and discusses how to set a reasonable deduplication reward.

6.1 | Instantiation

In this section, we briefly explain how our proposed BDAI works. In the **Setup** stage, the system manager selects a

security parameter λ and generates system parameters $\text{param}_{\text{BDAI}}$. The system manager publishes system parameters and goes off-line afterwards. During the **KeyGen** stage, the server and users (data owner and deduplication user) acquire the public key and the private key pair. During the **Pre-Processing and Outsourcing** stage, the data owner U_{own} pre-processes the file m to derive file ciphertext and metadata: $C = \{c \parallel \text{meta}\}$. Meanwhile, U_{own} transfers an outsourcing fee sum_m^{\sim} to the server via blockchain to acquire outsourcing service. On receiving $C = \{c \parallel \text{meta}\}$, the server runs the **Deduplication & Equality Test** to check redundancy and consistency of the metadata.

Suppose U_{dedup} is a deduplication user who holds the file m , then U_{dedup} pre-processes the file m to derive $C = \{c \parallel \text{meta}\}$ as defined in the **Pre-Processing and Outsourcing** stage. To grant U_{dedup} the access to the file c , the server and U_{dedup} engage in a **Proof-of-Ownership** protocol to check the file ownership. In the PoW protocol, a MHT (Ω) is constructed to authenticate the file structure. The server will refuse to grant the deduplication user the file access if the user fails the **Dedup.verify** step. Meanwhile, no refund will be processed as a consequence.

When two users dispute the original file ownership, they can summon the server to host the **DU** protocol. When the user is in dispute with the server about the payment, they can engage in the **DSU** protocol. Since all protocols are publicly verifiable and the server is assumed to be honest in performing the protocol as negotiated, the dispute can be settled accordingly. Although the server is curious about the contents of the file, the confidentiality is achieved to prevent the server from learning the stored files with the best possible security.

6.2 | Pricing

Here, we discuss how to set up a reasonable price as a deduplication reward to compensate for the outsourcing costs.

We give the meaning of the symbols in Table 1. Our goal is to calculate an ideal n_f which compensates the outsourcing costs from the deduplication rewards. Concretely, we set n_f as the number of files taken to compensate the outsourcing fee (sum_m^{\sim}) via the deduplication reward ($\text{sum}_m^{\sim'}$). Denote cost_s as the storage cost of the server for keeping a file; cost_o as the outsourcing cost charged by the server; and cost_d as the downloading cost charged by the server. Since it is the user's privilege to download a file, we do not investigate the connection between the downloading cost cost_d with other costs. For a file uploaded by one data owner and shared with many deduplication users, suppose the server charges the data owner U_{own} with sum_m^{\sim} for outsourcing and charges each deduplication user U_{dedup} with $\text{sum}_m^{\sim'}$ to enjoy the deduplication. To compensate sum_m^{\sim} with $\text{sum}_m^{\sim'}$, we have the following equation:

$$\text{cost}_o + \text{sum}_m^{\sim} = n_f \cdot \text{sum}_m^{\sim'} \quad (9)$$

For the simplicity, we assume the data owner should pay the server at least $cost_s$ to store the file m . This is reasonable since the user will pay for the costs to maintain any unpopular files at once, and the server will always carry out its duty to store the files after getting paid. Thereby, we set $sum_m = cost_s$ and we have:

$$sum_m' = \frac{sum_m + cost_s}{n_f} \quad (10)$$

For quantitative analysis, we release the relevant costs associated with the cloud storage in Table 2. To explain, as accessed on fourth.July.2020, the lowest outsourcing charge in the US market was \$0.005/GB/Month (by BACKBLAZE), the highest charge is \$0.020/GB/Month (by Google Cloud). To download files from the server, the lowest charge is \$0.010/GB (by BACKBLAZE), the highest charge is \$0.080/GB/Month (by Google Cloud). Additionally, a minimum bandwidth charge is applied (\$0.087/GB/Month) to each user. For simplicity, we do not consider the expiration of the paid storage service. For analysis, we range the file size from 0.1TGB to 10TGB, and give the total costs of 100 files for different price items in Table 2. To evaluate the saved time, we

assume the user with standard bandwidth (100MBps), which offers approximately 3 MB/s uploading speed and 12.5 MB/s downloading speed. We also assume the user with fast bandwidth (100GBps, e.g. 5G network), which offers approximately 20 MB/s uploading speed and 128 MB/s downloading speed.

As given in Table 2, for 100 files of 0.1 TGB size, we have: $0.97 * 10^{-3} \leq sum_m' \leq 1.27 * 10^{-3}$. For 100 files of 10 TGB size, we have: $0.097 \leq sum_m' \leq 0.127$. We can observe that: by setting $n_f = 100$, sum_m' is always below $1\%cost_o$. Hence, $sum_m' = 1\%sum_m$ is a reasonable charge as a deduplication reward to compensate for the outsourcing cost. In addition, it takes Bitcoin at most 0.5 or 1 h to confirm the payment. For a popular file smaller than 0.1 TGB, there is no need for the data owner to pay an outsourcing fee in advance in that this fee will be quickly compensated by the deduplication rewards before the payment is confirmed (if the file is popular and is uploaded via 5G network which triggers the deduplication). In other words, under such circumstance, data owners are encouraged to upload and share files with others because they do not have to pay and even can gain some extra rewards in return.

7 | PERFORMANCE EVALUATIONS

In this section, we give performance evaluation of our BDAI.

7.1 | Experimental environment

Simulations are coded in C language, and cryptographic operations are based on the PBC-0.5.13 library. We chose a supersingular curve $y^2 = x^3 + x$ with an embedding degree of 2. We set that curve group G is of the 160 bits group order. We set that $|G| \approx 160$ bits and $|G_T| \approx 1024$ bits as the binary sizes of groups $|G|$ and $|G_T|$, respectively. All computational experiments are run on a laptop with a 3.5 GHz 4-cores CPU, 8 GB RAM and 256 SSD for the storage. The operating system is 32 bits Windows 7 SP1. We use OpenSSL (version-1.1.1) as a means of secure communication. For large data storage, we utilise TS-932PX as a RAID device to store a large data set. We compute the hash value using SHA256. The bandwidth environment is 100 Mbps and 100 Gbps (using 5G network for fast transmission). Each result is computed by a mean of 10 consecutive trials.

TABLE 1 Meaning of symbol

Symbol	Type	Meaning
n_f	?	The number of deduplication Rewards taken to compensate Outsourcing fee sum_m
sum_m	?	Outsourcing fee charged With the user by the server.
sum_m'	?	Deduplication reward charged With deduplication user U_{dedup} By data owner U_{own} .
$cost_s$	◦	Storage cost of the server
$cost_d$	◦	Downloading cost charged With user by the server
$cost_o$	◦	Outsourcing cost charged With user by the server

Denote ? as a variable to be determined; ◦ as a constant which is given by survey.

TABLE 2 Comparison of costs for outsourcing and deduplication

File size (TGB)	Saved storage ($cost_s$) cost (\$)/per file	Outsourcing cost (\$)/per file ($cost_o$)	Downloading ($cost_d$) Cost (\$)/per file	Saved time (hour)/per file (100MBps)	Saved time (hour)/per file (100GBps)
0.1×10^2	Min.\$0.05, Max.\$0.2	Min.\$0.92, Max.\$1.07	Min.\$0.97, Max.\$1.67	0.95 h	0.14 h
1×10^2	Min.\$0.5, Max.\$2	Min.\$9.2, Max.\$1.7	Min.\$9.7, Max.\$16.7	9.5 h	1.4 h
10×10^2	Min.\$5, Max.\$20	Min.\$92, Max.\$107	Min.\$97, Max.\$167	95 h	14 h

Denote T_m as group multiplication; T_e as group exponentiation; T_i as group inversion; T_p as bilinear pairing operation; n/a as not applicable; Q as the challenged set for PoW; n as number of blocks per file; $n' > n$ as a positive number; T_h as hash operation; T_{sym} as the symmetric encryption; $\ell = \log_{|m_i|}^n$ as the discrete logarithm to n with block size $|m_i|$ as the base; T_{mht} as the computation of a merkle hash tree; T_{xor} as the bit-wise XOR operation.

7.2 | Theoretical analysis of BDAI

We compare the complexities of different schemes in Table 3. For our BDAI and BL-MLE, we omit counting expenditures caused by hashing. For UMLE, hashing is counted since it is one of the few notable complexities (together with symmetric encryption). As it is shown in Table 3, our BDAI introduces some constant costs to generate arbitration tags (chameleon hashes) than BL-MLE. In addition, the costs of our PoW protocol are linear with the file size and computations of Merkle Hash Tree (MHT). The use of MHT allows to enumerate each block for authentication. This presents the server or user to only store the file tags in order to pass the verification of PoW. The UMLE scheme is efficient in almost every stage since it mainly involves deterministic hashing and symmetric encryption which are generally fast to execute. However, unlike our BDAI and BL-MLE, diverse functions (e.g. proof-of-storage, arbitration) based on bilinear pairing or other cryptographic primitives are not supported.

7.3 | Experimental analysis of BDAI

We set the encoded file M as 0.1T GB using the (255,223,32)-Reed-Solomon (RS) code where blocksize is 1024 bits and the number of blocks is $n = 2^{33}$. Suppose the probability of the adversary in passing the MHT verification via random guessing in **Dedup.Verify** is less than 1.5%, which requires $(1 - p)^c < 1.5\%$ where $a \approx 2^{-3}$. Consequently, $c = 2^5$. In other words, it is required to upload 2^5 leaf nodes and corresponding siblings to derive the root of MHT. When considering the worst case, one needs to produce no more than $2^5 \times (\log n - \lfloor \log c \rfloor) = 2^5 \times 18$ siblings and $2^5 \times 19$ blocks in order to derive the root of MHT [21]. The final communication cost is 76 KB. To achieve the higher detection probability, for example, $1 - 10^{-15}$, only 500 KB data is required to be transmitted as PoW proof for the verification.

For quantitative analysis, we conduct experiments to test the computational cost of our proposed BDAI. For the ease of simulations, we use synthetic data to test the deduplication performance. Before the encryption, we partition the file into blocks and apply RS codes to derive the file \tilde{m} . We construct MHT for each encrypted file for authenticating the entire file structure.

To evaluate the impact of the block size on metadata size, we range the block size from 15 KB to 40 KB, and test the metadata of 0.1 TGB file. For comparison, we also apply metadata generation based on the BL-MLE scheme introduced by Chen et al. [7]. As it is shown in Figure 2, our BDAI generates less metadata than BL-MLE. The gap between the two schemes in producing metadata narrows when the block size increases. Here, we set the similarity of these files as 50%. As it is shown in Figure 3, when the redundancy rate varies from 0% to 100%, the deduplicated data processed based on both BDAI and BL-MLE drop. Although BL-MLE achieves the best saving in comparison with our BDAI, according to Table 3, BL-MLE produces more metadata to achieve the deduplication.

To test the impact of the number of files on the storage or time saving, we fix the file size by 0.1TGB and range the file number from 1 to 100. We set $n_f = \text{sum}_{\tilde{m}}' = 1\% \text{sum}_{\tilde{m}}$ and record the saved costs and time in Figure 4 and Figure 5, respectively. As is shown in Figure 4, although the outsourcing costs surge with the number of files, it is finally compensated by the deduplication rewards. As depicted in Figure 5, for slow bandwidth with a relatively small file data set (under 0.4TGB as

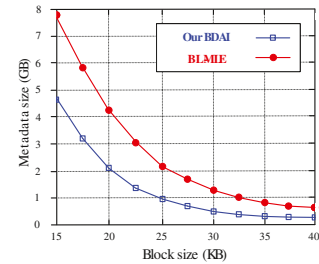


FIGURE 2 Impact of block size on metadata size

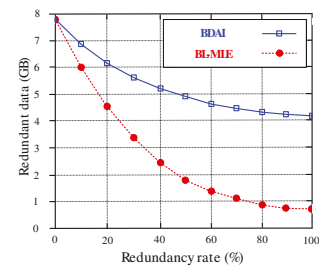


FIGURE 3 Impact of re-deduplication rate on metadata size

TABLE 3 Complexities of different schemes

Scheme	MetaGen	Enc	PoW protocol	TC Check	Dec	Arbitration
Our BDAI	$(8 + s)T_e + (4 + n)T_m + (4 + n)T_p$	nT_{xor}	$\mathcal{O}(m)T_{\text{mht}}$	$2T_e + \mathcal{O}(m)T_p$	$(s + n + 1)T_e + nT_m + (ns + n)T_i + nT_{\text{xor}}$	$8T_e + 8T_m + 4T_p + 1T_i$
BL-MLE [7]	$(1 + s)T_e + nT_m + nT_p + nT_m + nT_p$	nT_{xor}	$ Q (T_e + T_m)$	$\mathcal{O}(m)(2T_p)$	$(s + n + 1)T_e + nT_m + (ns + n)T_i + nT_{\text{xor}}$	n/a
UMLE [18]	nT_b	$\mathcal{O}(n)(T_{\text{sym}} + (\mathcal{O}(l)T_h + T_{\text{sym}}))$	$ Q T_{\text{sym}}$	$\mathcal{O}(1)T_h$	$\mathcal{O}(n')\mathcal{O}(l)T_{\text{sym}}$	n/a

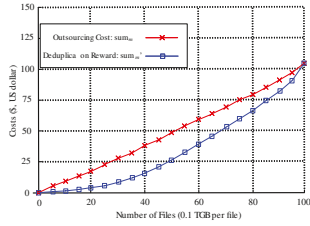


FIGURE 4 Impact of number of files on saved costs

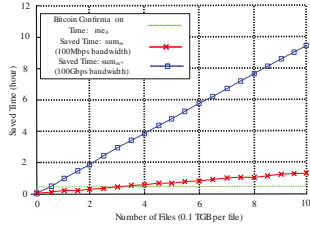


FIGURE 5 Impact of number of files on saved time

total size), the deduplication process is faster than Bitcoin payment. This means the server has to wait until the Bitcoin payment is confirmed before granting the deduplication user direct access to the file. However, this can be overcome by adopting cryptocurrencies with fast payment (e.g. Ethereum) [12] or adopting fast bandwidth (such as 5G network).

8 | RELATED WORK

Deduplication is a technique to abort the redundant uploading request and eliminates the identical files from the server to save costs for a cloud server. From the deduplication policy perspective, there are target-based [5, 21–23] and source-based deduplication [4, 15]. The former tends to delete redundant data after data is uploaded, while the latter can detect and abort redundant uploading requests.

Regarding the confidentiality of deduplication, symmetric encryption is often applied to data before it is outsourced. However, this raises problems of key derivation and key management [24]. To cope with, Douceur et al. [3] first proposed the notion of convergent encryption (CE), which encrypts a file by hash-as-a-key manner such that the key is generated from the file itself. Later on, Bellare et al. [5] formally defined the security of convergent encryption and proposed Message-Locked Encryption (MLE). Since then, the notion of MLE becomes a hot research topic. To note, Chen et al. extended MLE to BL-MLE [7] to support fine-grained savings at block-level. As another notable work about block-level deduplication, Zhao and Chow et al. [18] proposed the notion of updatable block-level message-locked encryption (UMLE). They achieved the desirable security by formalising block-tag consistency and pointing out the security flaws in [7]. They also achieved computation cost logarithmic in the file size. As other notable developments of MLE, notion of MLE-2 [25] and interactive MLE [26] were proposed as strong

security, and randomised the proposed encryption and meta-data generation.

For blockchain-based deduplication schemes, current works either exploit blockchain as an open trust-layer (e.g. distributed database) or utilise it to achieve decentralised services (e.g. smart contract-enabled service). Ghazouani et al. [27] proposed to adopt blockchain as a database for storing metadata of client files. This database serves as a logging database to grant data integrity auditing. Yuan et al. [28] proposed to exploit smart contract to enable automatic penalisation and fair arbitration of malicious behaviours. Li et al. [29] proposed a deduplication scheme to distribute files to multi-servers and record storage information on blockchain. They also exploited smart contract to provide secure deduplication service. Wang et al. [30] proposed a decentralised fair payment protocol for deduplication. The proposed protocol takes advantage of the Ethereum blockchain, which guarantees fair payment by pre-storing penalty money in the smart contract. Xu et al. [31] proposed a blockchain-based data auditing scheme with deduplication where blockchain is used to record all behaviours for auditing and traceability purposes. Huang et al. [32] proposed to utilise the traceability of the blockchain to resist duplicate faking attack. By forming a tamper-proof data chain, the identities of the malicious users who have launched attacks could be revealed.

To generalise, the above deduplication schemes mainly build extra functions based on blockchain. One notable use is to build dispute arbitration based on traceable and immutable features of blockchain. However, the above schemes did not use blockchain to distribute financial incentives to boost the deduplication services. To note, blockchain (such as Bitcoin) has been successfully implemented in the financial sector in the first place. Meanwhile, cloud servers adopted deduplication out of financial concerns. Thereby, exploring and investigating how to solve financial problems in a deduplication system with blockchain is an interesting research gap.

9 | CONCLUSION

In this work, we proposed a blockchain-based deduplication with arbitration and financial incentives (BDAl). The data owner pays money to the server to gain outsourcing service, but deduplication services can compensate this fee. Both the data owner and the server can receive financial incomes to encourage the outsourcing actions and boost the deduplication services. Our work is an extension of BL-MLE, which integrates chameleon hash to achieve dispute arbitration. We give the concrete construction and security requirements for our BDAl. The security analysis shows that our BDAl is theoretically secure. The performance evaluation shows that our proposed BDAl is acceptably efficient for the deduplication. Meanwhile, we conclude that 1% of the outsourcing fee (or less) is a preferable and reasonable price for each deduplication user to pay as the compensation for the data owner.

In the future, we would like to investigate an elegant design of the deduplication scheme to achieve tag consistency security

or its stronger version. In addition, we also would like to study the deduplication scheme under a stronger security model, such as leakage-resistance and key-exposure freeness. We would also seek more diverse application scenarios. A deduplication mechanism with more practical incentives and accuracy is also an interesting topic.

ACKNOWLEDGEMENT

This work was supported in part by the National Key R&D Program of China under Grant No.2017YFB0802300, Grant No.2018YFB08040505 and Grant No.2018YFB0804100; National Natural Science Foundation of China under Grants 62002048, 61872087, U19A2066, 62072078; University Startup Grant under Grant No.Y030202059018061; Blockchain Research Lab of UESTC, Chengdu Jiaozhi Financial Holding Group Company Ltd; China Mobile Information Communication Technology Co., Ltd (Chengdu) 2020 UAV Operation Management Platform Phase II (Package 2: Safety Subsystem; No. CMCMI-202001245).

CONFLICT OF INTEREST

No.

PERMISSION TO REPRODUCE MATERIALS FROM OTHER SOURCES

None.

DATA AVAILABILITY STATEMENT

Research data are not shared.

ORCID

Ke Huang  <https://orcid.org/0000-0002-5102-610X>

REFERENCES

- Kamara, S., Lauter, K.: Cryptographic cloud storage. In: International Conference on Financial Cryptography and Data Security, pp. 136–149. Springer, Tenerife, Canary Islands, Spain (2010)
- Gantz, J., Reinsel, D.: The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far east. IDC iView: IDC Analyze the future. 2007(2012), 1–16 (2012)
- Douceur, J.R., et al.: Reclaiming space from duplicate files in a serverless distributed file system. In: Proceedings of the 22nd International Conference on Distributed Computing Systems, 617–624. IEEE, Vienna, Austria (2002)
- Harnik, D., Pinkas, B., Shulman-Peleg, A.: Side channels in cloud services: deduplication in cloud storage. *IEEE Secur. Priv.*, vol. 8, 40–47, Oakland, California, USA (2010)
- Bellare, M., Sriram, K., Ristenpart, T.: Message-locked encryption and secure deduplication. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 296–312. Springer, Athens, Greece (2013)
- Jiang, T., et al.: Secure and efficient cloud data deduplication with randomized tag. *IEEE Trans. Inf. Forensics Secur.* 12(3), 532–543 (2017). <https://doi.org/10.1109/tifs.2016.2622013>
- Chen, R., et al.: Block-level message-locked encryption for secure large file deduplication. *IEEE Trans. Inf. Forensics Secur.* 10(12), 2643–2652 (2015). <https://doi.org/10.1109/tifs.2015.2470221>
- Huang, K., et al.: HUCDO: a hybrid user-centric data outsourcing scheme. *ACM Trans. Cyber-Phys. Syst.* 4(3), 1–23 (2020). <https://doi.org/10.1145/3379464>
- Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. Technical report, Manubot (2019)
- Gai, K., et al.: Blockchain meets cloud computing: a survey. *IEEE Commun. Surv. Tutorial* 22(3), 2009–2030 (2020). <https://doi.org/10.1109/comst.2020.2989392>
- Wood, G., et al.: Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum project yellow paper, vol. 151(2014), pp. 1–32 (2014)
- O Karame, G., Androulaki, E., Capkun, S.: Double-spending fast payments in bitcoin. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 906–917. ACM, Raleigh, North Carolina, USA (2012)
- Boneh, D., Boyen, X.: Short signatures without random oracles. In: International Conference on the Theory and Applications of Cryptographic Techniques, pp. 56–73. Springer, Interlaken, Switzerland (2004)
- Schnorr, C.P.: Efficient signature generation by smart cards. *J. Cryptol.* 4(3), 161–174 (1991). <https://doi.org/10.1007/bf00196725>
- Halevi, S., et al.: Proofs of ownership in remote storage systems. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 491–500. ACM, Chicago Illinois, USA (2011)
- Jan, C., et al.: Chameleon-hashes with ephemeral trapdoors. In: IACR International Workshop on Public Key Cryptography, pp. 152–182. Springer, Amsterdam, Netherlands (2017)
- Huang, K., et al.: Design and Analysis of Cryptographic Algorithms in Blockchain. CRC Press (2021)
- Zhao, Y., ShermanChow, S.M.: Updatable block-level message-locked encryption. *IEEE Trans. Dependable Secure Comput.* 18(4), 1620–1631 (2019). <https://doi.org/10.1109/tdsc.2019.2922403>
- Conti, M., et al.: A survey on security and privacy issues of Bitcoin. *IEEE Commun. Surv. Tutorial*. 20(4), 3416–3452 (2018). <https://doi.org/10.1109/comst.2018.2842460>
- Huang, K., et al.: Bidirectional and malleable proof-of-ownership for large file in cloud storage. *IEEE Trans. Cloud Comput.*, 1 (2021). <https://doi.org/10.1109/tcc.2021.3054751>
- Jin, X., et al.: Anonymous deduplication of encrypted data with proof of ownership in cloud storage. In: 2013 IEEE/CIC International Conference on Communications in China (ICCC), pp. 224–229. IEEE, Xi'an, China (2013)
- Storer, M.W., et al.: Secure data deduplication. In: Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, pp. 1–10. ACM, Alexandria, Virginia, USA (2008)
- Ng, W.K., Wen, Y., Zhu, H.: Private data deduplication protocols in cloud storage. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 441–446. ACM, Trento, Italy (2012)
- Jin, L., et al.: Secure deduplication with efficient and reliable convergent key management. *IEEE Trans. Parallel Distr. Syst.* 25(6), 1615–1625 (2013). <https://doi.org/10.1109/tpds.2013.284>
- Abadi, M., et al.: Message-locked encryption for lock-dependent messages. In: Annual Cryptology Conference, pp. 374–391. Springer, Santa Barbara, CA, USA (2013)
- Bellare, M., Sriram, K.: Interactive message-locked encryption and secure deduplication. In: IACR International Workshop on Public Key Cryptography, pp. 516–538. Springer, Gaithersburg, MD, USA (2015)
- El Ghazouani, M., Ahmed El Kiram, M., Er-Rajj, L.: Blockchain & multi-agent system: a new promising approach for cloud data integrity auditing with deduplication. *Int. J. Commun. Network. Inf. Secur.* 11(1), 175–184 (2019)
- Yuan, H., et al.: Blockchain-based public auditing and secure deduplication with fair arbitration. *Inf. Sci.* 541, 409–425 (2020). <https://doi.org/10.1016/j.ins.2020.07.005>
- Li, J., et al.: Deduplication with blockchain for secure cloud storage. In: CCF Conference on Big Data, pp. 558–570. Springer, Xi'an, China (2018)
- Wang, S., Wang, Y., Zhang, Y.: Blockchain-based fair payment protocol for deduplication cloud storage system. *IEEE Access.*

- 7, 127652–127668 (2019). <https://doi.org/10.1109/access.2019.2939492>
31. Xu, Y., et al.: A blockchain-enabled deduplicatable data auditing mechanism for network storage services. *IEEE Trans. Emerg. Top. Comput.* (2020)
32. Huang, H., et al.: Blockchain-based secure cloud data deduplication with traceability. In: *International Conference on Blockchain and Trustworthy Systems*, pp. 295–302. Springer, Dali, China (2020)

How to cite this article: Huang, K., et al.: Blockchain-based deduplication with arbitration and incentives. *IET Inf. Secur.* 1–16 (2022). <https://doi.org/10.1049/ise2.12066>