12-2020

Blockchain-based public auditing and secure deduplication with fair arbitration

Haoran YUAN

Xiaofeng CHEN

Jianfeng WANG

Jiaming YUAN
Singapore Management University, jmyuan@smu.edu.sg

Hongyang YAN

See next page for additional authors

Blockchain-based public auditing and secure deduplication with fair arbitration

Haoran Yuan[a], Xiaofeng Chen[a], Jianfeng Wang[a], Jiaming Yuan[b], Hongyang Yan[c], Willy Susilo[d]

[a]State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, PR China

[b]School of Information Systems, Singapore Management University, Singapore

[c]School of Computer Science, Guangzhou University, Guangzhou, PR China

[d]School of Computing and Information Technology, University of Wollongong, Australia

Published in Information Sciences, December 2020, 541, 409-425

https://doi.org/10.1016/j.ins.2020.07.005

Abstract

Data auditing enables data owners to verify the integrity of their sensitive data stored at an untrusted cloud without retrieving them. This feature has been widely adopted by commercial cloud storage. However, the existing approaches still have some drawbacks. On the one hand, the existing schemes have a defect of fair arbitration, i.e., existing auditing schemes lack an effective method to punish the malicious cloud service provider (CSP) and compensate users whose data integrity is destroyed. On the other hand, a CSP may store redundant and repetitive data. These redundant data inevitably increase management overhead and computational cost during the whole data life cycle. To address these challenges, we propose a blockchain-based public auditing and secure deduplication scheme with fair arbitration. By using a smart contract, our scheme supports automatic penalization of the malicious CSP and compensates users whose data integrity is damaged. Moreover, our scheme introduces a message-locked encryption algorithm and removes the random masking in data auditing. Compared with the existing schemes, our scheme can effectively reduce the computational cost of tag verification and data storage costs. We give a comprehensive analysis to demonstrate the correctness of the proposed scheme in terms of storage, batch auditing, and data consistency. Also, extensive experiments conducted on the platform of Ethereum blockchain demonstrate the efficiency and effectiveness of our scheme.

Keywords: Blockchain, Data auditing, Fair arbitration, Data deduplication

1. Introduction

Cloud computing provides users with flexible computing and storage resources. It significantly reduces the burden of software and hardware management, which attracts many individuals and enterprises to outsource their confidential data to remote cloud servers. However, since users' data is outsourced to the CSPs and stored on the remote cloud, it separates the ownership and management rights of users' data. This makes it difficult for users to verify the integrity of their sensitive data.

Corresponding author. E-mail address: hyang.yan@foxmail.com (H. Yan).

In the cloud computing environment, the outsourced sensitive data may be tampered and deleted, due to various hardware and software vulnerabilities, and even malicious adversary attacks. For example, Google's Europe-west1-b data center in Belgium was affected by lightning strikes in 2015. At the height of the calamity, about 5% of the disks in the data center experienced I/O errors. This led to the permanent loss of 100 gigabytes (GB) data in Europe-west1-b even though Google took a series of precautions [1]. Nearly 50 million Turkish citizens' personal information was accessed by hackers and posted online in a downloadable 6.6 GB file in 2016 [2]. Kromtech security researchers discovered that an Amazon S3 repository could be publicly accessed in 2017. About 316,363 PDF medical reports in this repository were leaked and more than 150,000 patients were affected by this leak [3]. Various hacking attacks and data loss problems indicate that data security has become one of the most critical issues in cloud computing [4–10]. For ensuring the security of users' sensitive data, numerous data auditing schemes have been proposed [11–18]. Data auditing schemes effectively verify the data integrity of outsourced data without downloading the original data. However, the existing schemes still have a defect of fair arbitration, i.e., data owners may not obtain the compensations even if they discover that the outsourced data has been destroyed.

Besides, with the surge in the number of cloud users, the amount of users data is exploding. According to the report from Internet Data Center (IDC), the total amount of digital data grows at a rate that doubles every two years, which is expected to reach 44 trillion gigabytes (GB) in 2020 [19]. Moreover, the Global Datasphere would grow from 33 ZB in 2018 to 175 ZB in 2025, from the investigation Seagate and IDC DataAge White Paper [20]. This trend makes CSPs caught in the swamp to cope with the increasing demand for disk space and bandwidth. To solve this problem, a simple method is to ask CSPs to increase the storage space for adapting users' requirements for high-quality cloud storage services. Nevertheless, CSPs may store repetitive and redundant data, which inevitably occupies a large number of backup and storage space. To address this problem, Bolosky et al. [21] proposed the idea of data deduplication, which enables CSPs to delete the repetitive data and only keep one copy of them to decrease the bandwidth and storage space. Nowadays, a large number of CSPs have applied data deduplication techniques, such as Google Drive [22], Memopal [23] and Dropbox [24]. Research [25] has shown that data deduplication can reduce at least 90% of business application storage and bandwidth costs.

Although data deduplication technology brings significant benefits to CSPs, there remain some problems to be solved. It is generally considered that CSPs are not fully trusted; they are curious about the outsourced data. Thus, a user usually encrypts sensitive data before uploading them to the cloud. Different users choose file keys independently to encrypt the same data and generate different ciphertexts, which hinders the realization of data deduplication. Convergent encryption (CE) is the first realistic scheme to guarantee data confidentiality and support ciphertext deduplication [26]. In this scheme, a user uses a convergent key to encrypt the sensitive data, where the convergent key is computed by hashing the sensitive data. Thus, the same data can always be encrypted to the same ciphertext. This allows the CSP to perform deduplication on ciphertexts. However, users' data may be corrupted due to various network failures during the downloading process and software failures during the decryption process. If the user decrypts the corrupted data, he could not obtain the correct plaintext. Thus, the CE scheme cannot protect the data consistency of users' sensitive data.

To address the above problems, it is fundamental to design a data auditing and secure deduplication scheme with fair arbitration, which enables automatic data auditing without third-party auditors and consistent detection during decryption. In should be stressed that designing a data auditing and secure deduplication scheme with fair arbitration is not a trivial problem. First, existing data auditing schemes [11–13,15] exclusively focus on how to achieve probabilistic auditing or batch auditing and rarely consider fair arbitration. The data owner cannot obtain the corresponding compensation once the data integrity is destroyed, which is unfair to the data owner. Therefore, how to design a data auditing scheme with fair arbitration to punish the malicious CSP and compensate users whose data integrity is destroyed becomes an urgent problem that needs to be solved. Second, many existing data auditing schemes resort to a third-party auditor to check the data integrity of outsourced data and the correctness of the integrity audit results is completely dependent on the trusted third-party. However, finding a fully trusted third-party is often unrealistic. Finally, existing data auditing and data deduplication schemes [16–18] only adopt the CE scheme to realize data deduplication. Therefore, these schemes cannot guarantee data consistency of users' sensitive data.

1.1. Our contribution

In this paper, we propose a blockchain-based public auditing and secure deduplication with fair arbitration. Specifically, our scheme utilizes the blockchain technique and public-key-based homomorphic linear authenticator algorithm to realize data auditing without any third-party auditor and automatic punishment. By integrating the Hash-and-CE-2 scheme [27], our scheme supports data deduplication and consistent detection during decryption. In summary, our contribution can be summarized as the following three folds:

- We propose a blockchain-based public auditing and secure deduplication scheme with fair arbitration. Different from the previous works, our scheme supports data auditing without any third-party auditor and automatically compensates users whose data integrity is damaged by using the smart contract. This effectively guarantees the benefits of users. In addition, our scheme supports data deduplication, which reduces the storage and computation overhead of the CSP.
- We propose a batch data auditing and secure deduplication scheme, which can batch audit multiple auditing tasks at the same time. Moreover, by employing an additional consistent detection mechanism, our scheme guarantees data consistency.

- We provide security analysis of our scheme, and the results show that our scheme can realize the expected security goals. Also, extensive experiments conducted on the platform of Ethereum blockchain demonstrate the efficiency and effectiveness of our scheme.

1.2. Related work

1.2.1. Data auditing

In 2007, Ateniese et al. [11] proposed the first public audit scheme in the provable data possession (PDP) model for verifying the authenticity of data in untrusted servers. By using the homomorphic linear authenticators, users can efficiently verify data integrity without downloading the entire data. However, this scheme cannot guarantee data privacy. The user information will be leaked. For ensuring the retrievability of outsourced data, Juels et al. [28] proposed proof of retrievability (PoR) in 2007. Based on the spot-checking and error-correcting codes, PoR can support data possession and retrievability at the same time. However, this scheme does not support public auditability and the number of audit challenges is limited. In the security model [28], Shacham and Waters [29] proposed an improved PoR scheme with proofs of security based on the BLS signatures. To achieve public and dynamic data auditing for remote data, Wang et al. [12] proposed a data auditing scheme, which can achieve public auditability and data dynamics at the same time. By adopting the bilinear aggregate signature, this scheme can support multiple auditing tasks simultaneously. To protect the data information of users, Wang et al. [13] proposed a privacy-preserving public auditing scheme. This scheme enables third-party auditor (TPA) to audit multiple tasks simultaneously. Wang et al. [15] designed a verifiable auditing scheme, which can protect the search results' correctness and completeness. To support both data integrity and data deduplication in cloud storage, Li et al. [16] proposed the SecCloud and SecCloud + schemes. By introducing the MapReduce algorithm, SecCloud can help users to generate data tags before uploading and auditing data integrity of outsourced data. SecCloud + enables data auditing and deduplication on encrypted data. Liu et al. [17] proposed a message-locked integrity auditing and data deduplication scheme, which can realize the deduplication of authentication tags.

1.2.2. Data deduplication

In 2002, Douceur et al. [26] proposed the first ciphertext deduplication scheme, which is called convergent encryption (CE). In this scheme, a user uses a convergent key to encrypt sensitive data, where the convergent key is derived by hashing the sensitive data. Since the data are the same, the different users always are able to generate the same convergent key. By using the same convergent key, the same data can be encrypted to the same ciphertext. This allows the CSP to perform deduplication on ciphertexts. However, the CE scheme cannot prevent brute-force dictionary attacks. To cope with it, the DupLESS scheme was proposed by Bellare et al. [30]. In the DupLESS scheme, a user generates encryption keys with the help of a dedicated key-server by using oblivious pseudorandom functions (OPRF) protocol. The key server is configured with a system-wide public and private key based on the RSA mechanism. Thus, the key server is able to generate message-locked encryption (MLE) key without knowing the hash value of the original data [31]. The rate-limits mechanism is also adopted in the DupLESS scheme, which limits the query times of MLE key generation. Therefore, brute-force attacks can be efficiently avoided. Bellare et al. [27] proposed the Hash-and-CE-2 (HCE2) scheme and added a tag checking mechanism in this scheme. After decrypting the ciphertext, a user recomputes the tag by using the plaintext and compares it with the original tag. If the newly generated tag is equal to the original tag, the user accepts the ciphertext; else, rejects it. To support efficient and reliable key management, Li et al. [32] proposed a secure data deduplication scheme based on the ramp secret sharing scheme. Shin et al. [33] applied the predicate encryption algorithm in the deduplication scheme. However, this scheme only achieves single-user deduplication. Based on the traceable signatures, Wang et al. [34] proposed a ciphertext deduplication scheme. In the hybrid cloud architecture, Li et al. [35] proposed an authorized data deduplication scheme, which can support authorized duplicate checks. Yuan et al. [36] proposed a scalable and dynamic deduplication scheme, which can support user joining and revocation.

1.2.3. Blockchain

Bitcoin is the first and most popular decentralized cryptocurrency, which allows users online payment without a financial institution [37]. To solve the problem of double-spending, blockchain was adopted in the bitcoin system. Blockchain is an ever-increasing and public distributed transaction ledger, which can record an immutable history of transactions and provide a tamper-proofing ledger without any central authority. The blockchain collects data elements as a block, which contains version, nonce, previous block hash, root hash of Merkle tree and timestamp. After verifying the validity, the transaction can be recorded in the blockchain. In a blockchain, the hash value of the current transaction is used to generate the next block. Therefore, blockchain can efficiently authenticate the history of data and resist modification of chained blocks. To realize public verification of data deletion, Yang et al. [38] proposed a data deletion scheme based on blockchain, which can support public verification. For the fog devices, Huang et al. [39] proposed a bitcoin-based fair payment for outsourcing computation. Alptekin [40] proposed an official arbitration scheme with a secure cloud storage application, which only costs 2 ms and 80 bytes for each update on the stored data to resolve disputes. To support dynamic and public auditing with fair arbitration, Jin et al. [41] proposed fair arbitration protocols, which can support fairness arbitration of potential disputes.

To improve the function of blockchain, the smart contracts were added in the Ethereum [42]. Smart contracts are computer programs running on cryptocurrency (e.g., Ethereum) blockchains, which can execute a pre-agreed program automatically (such as payments and audits) and without a trusted authority. The Ethereum blockchain is described in Fig.1. We use Tx to denote a transaction. The hash value of the current block is denoted by BlockHash, the hash value of the previous block is denoted by PreHash, the timestamp is denoted by Timestamp, and the root value of Merkle hash tree is denoted by RootHash. Based on the Ethereum blockchain, Zhang et al. [43] proposed a secure data provenance scheme, which improves the security and privacy of the data provenance. In this paper, we use the Ethereum blockchain to build our scheme. Specifically, we use the smart contract to charge a certain penalty of the CSP and compensate the users whose data integrity is destroyed.

1.3. Organization

The rest of this paper is organized as follows. In Section 2, we introduce the preliminaries used in our scheme. We give the system architecture and security goals of our scheme in Section 3. In Section 4, we present a detailed description of our proposed scheme. The security analysis and efficient comparison are presented in Section 5. The performance evaluation results are shown in Section 6. Finally, conclusions are made in Section 7.

2. Preliminaries

In this section, we present the definitions and properties of Hash-and-CE-2 [27] and bilinear groups.

2.1. Hash-and-convergent-encryption-2

Hash-and-CE-2 (HCE2) is one of the message-locked encryption [27]. In HCE2, a user uses a convergent key to encrypt sensitive data, where the convergent key is derived by computing the hash of sensitive data. Thus, different users can always generate the same convergent key for the same data. Then, the different users use the same convergent key to encrypt the same data and get the same ciphertext. Therefore, HCE2 can realize data deduplication on the ciphertext. To protect data consistency, an additional tag checking mechanism is adopted in the HCE2 scheme. After decrypting the ciphertext, the user re-generates the tag of data by using the plaintext and compares it with the corresponding tag. The user accepts the data only if the tags are consistent. We describe the Hash-and-convergent-encryption-2 (HCE2) algorithm as follows.

Definition 1. The HCE2 algorithm **HCE2** = (**HCE2.KeyGen**, **HCE2.Encrypt**, **HCE2.Decrypt**, **HCE2.TagGen**) consists of the following algorithms:

- **HCE2.KeyGen**$(P, M_i) \rightarrow (K_i)$ is an MLE key generation algorithm that inputs public parameter P and file M_i, and then outputs the MLE key K_i;
- **HCE2.Encrypt**$(K_i, M_i) \rightarrow (C_i)$ is an encryption algorithm that inputs MLE key K_i and file M_i, and then outputs C_i;
- **HCE2.TagGen**$(C_i) \rightarrow T_i$ is a tag generation algorithm that inputs ciphertext C_i and outputs the tag T_i;
- **HCE2.Decrypt**$(K_i, C_i) \rightarrow (M_i)$ is a decryption algorithm that inputs MLE key K_i and ciphertext C_i, and then outputs the original plaintext M_i.

2.2. Bilinear groups

We review some concepts of bilinear maps, which include computability, bilinearity, non-degeneracy. Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three different multiplicative cyclic groups, where the order is p. We use g_1 and g_2 to denote the generator of $\mathbb{G}_1$ and $\mathbb{G}_2$. $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$is a bilinear map with the following properties:

(1) **Computability:** there exits an efficiently computable algorithm for computing map e;
(2) **Bilinearity:** for all $a \in \mathbb{G}_1, b \in \mathbb{G}_2$ and $x, y \in \mathbb{Z}_p, e(a^x, b^y) = e(a, b)^{xy}$;
(3) **Non-degeneracy:** $e(g_1, g_2) \neq 1$.

3. Model and security goals

3.1. System architecture

Based on the blockchain, we propose a public auditing and secure deduplication scheme with fair arbitration for cloud storage. Our scheme includes two entities: users and CSP. The system model of our scheme is shown in Fig. 2.

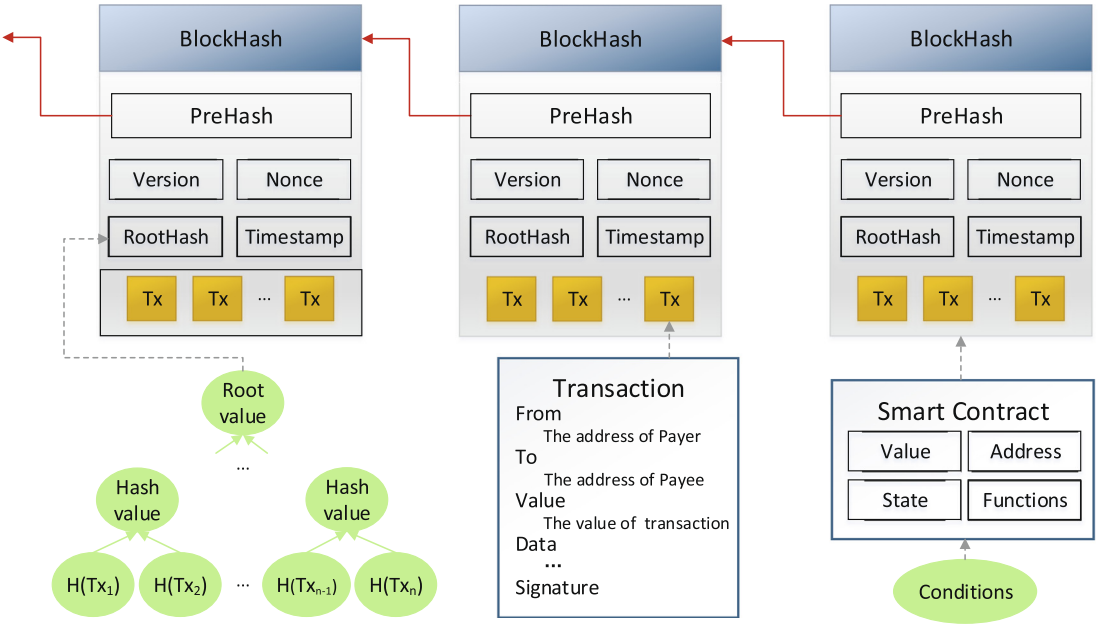

Fig. 1. Ethereum blockchain structure.

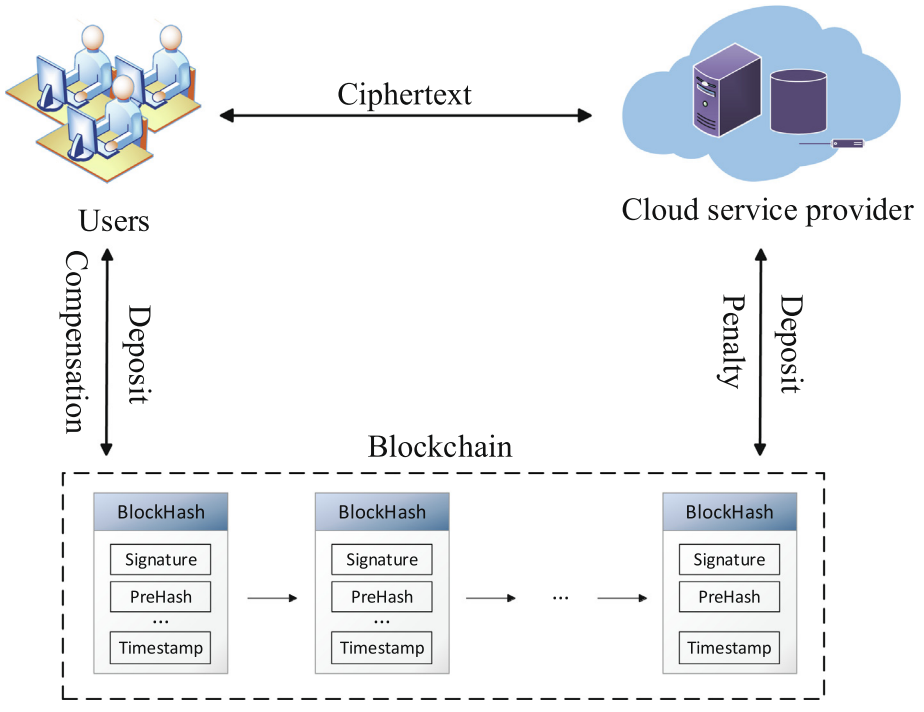

Fig. 2. The cloud storage model.

- Users: Users are the entities who want to upload sensitive data *M* to the CSP and download data later. Users encrypt the sensitive data before outsourcing them to the CSP for avoiding privacy information leakage. After uploading the sensitive data, users remove *M* for saving storage space.
- CSP: A CSP is an entity that provides storage services to users. The CSP is assumed honest-but-curious in our scheme. That means it will honestly execute the protocol but try to extract the content and information of users' outsourced data. The CSP cannot obtain the plaintext information of users' data directly by using the encryption algorithm in our scheme.

3.2. Threat model and security goals

We consider that the CSP is the main adversary. The malicious behaviors of the adversary are described as follows. First, the CSP may delete outsourced data that users rarely access to save storage costs for their economic interests. Thus, it is possible to obtain extra storage fees from users. Second, due to various software bugs, economically motivated hackers and hardware faults, the outsourced data stored on the CSP may be tampered or deleted. CSP may hide the data corruption incidents from users to maintain reputation.

We aim to achieve the following four security goals in this paper.

- Data privacy: We require that the CSP and malicious adversaries cannot obtain the user's plaintext.
- Storage correctness: We require that if a CSP can pass a smart contract's audit, it must store the entire user's data correctly.
- Batch auditing: We require that a smart contract can verify multiple audit tasks of users' sensitive data at the same time. If a CSP generates the correct response of integrity challenges, it must faithfully store all of the challenge blocks.
- Data consistency: In the cloud environment, users' data may be corrupted due to various network failures during the downloading process. If the user decrypts the corrupted data, he could not obtain the correct plaintext. In addition, various software failures may occur during the decryption process. Therefore, it is necessary to verify whether the decrypted data is equal to the original data. We require that users can determine whether the decrypted data is equal to the original plaintext.

4. Scheme construction

In this section, we first propose the main idea of our scheme. Then, we describe our scheme in detail.

4.1. Main idea

To solve the problems of data auditing without third-party auditor and integrity detection, we propose a blockchain-based public auditing and secure deduplication scheme with fair arbitration. The main idea is that a user first encrypts the sensitive data by using the HCE2 algorithm. Due to the intrinsic property of HCE2, users with the same data can always generate the same encryption key. By using the same encryption key, the identical data can always be encrypted to the same ciphertext. Then, the user uploads them to the CSP and the CSP makes a comparison between the stored data and the newly uploaded data. If the same data is found, it means that the same data has been stored in the CSP. Then, the CSP no longer stores the new data for saving storage space. The CSP also signs a smart contract with the user to achieve fair arbitration without any third-party. In a smart contract, a user and the CSP take a certain amount of deposit as input, respectively. If the CSP completely stores the user's data, the smart contract sends the user's deposit as an audit fee to the miner and returns the CSP's deposit to the CSP. If the CSP destroys the data integrity of the user's sensitive data, the smart contract charges the CSP's deposit and compensates the user. When performing data integrity auditing, the user first sends an audit challenge to the CSP. After receiving the challenge, the CSP calculates and sends a homomorphic verification tag to the smart contract according to the user's request. Finally, the smart contract verifies the homomorphic verification tag returned by the CSP. The overview of our scheme is shown in Fig. 3.

4.2. A concrete scheme

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three different multiplicative cyclic groups, where the order of the groups is p. We use g to denote the generator of $\mathbb{G}_2$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$is a bilinear map. $H(\cdot)$ is a secure hash function $\{0,1\}^* \rightarrow \mathbb{G}_1$. $h(\cdot)$ is a secure hash function $\mathbb{G}_T \rightarrow \mathbb{Z}_p$. f is a pseudorandom function: $\{0,1\}^* \rightarrow n$, where n denotes the total number of ciphertexts. $Sig(\cdot)$ is a signature algorithm. Let P be a public parameter of HCE2.

- **Setup**: Each user first randomly chooses a key pair (spk, ssk) as the private key and public key for signing, a random number $x \leftarrow \mathbb{Z}_p$ and computes $v \leftarrow g^x$. Then, the user randomly chooses an element $u \leftarrow \mathbb{G}_1$. The secret parameters are $sk = (x, ssk)$. The public parameters of our scheme are $pk = (v, u, P, spk, g, e(u, v))$.
- **Encrypt**: To upload a file F, a user first splits the file F into a set of chunks $\{M_1, M_2, \ldots, M_n\}$. For M_i, the user performs the following operations.

  – Execute the MLE key generation algorithm to generate MLE key $K_i \leftarrow$ **HCE2.KeyGen**(P, M_i).
  – Encrypt the plaintext M_i by computing $C_i \leftarrow$ **HCE2.Encrypt**(K_i, M_i) and generate tag $T_i \leftarrow$ **HCE2.TagGen**(C_i) by using the ciphertext C_i.

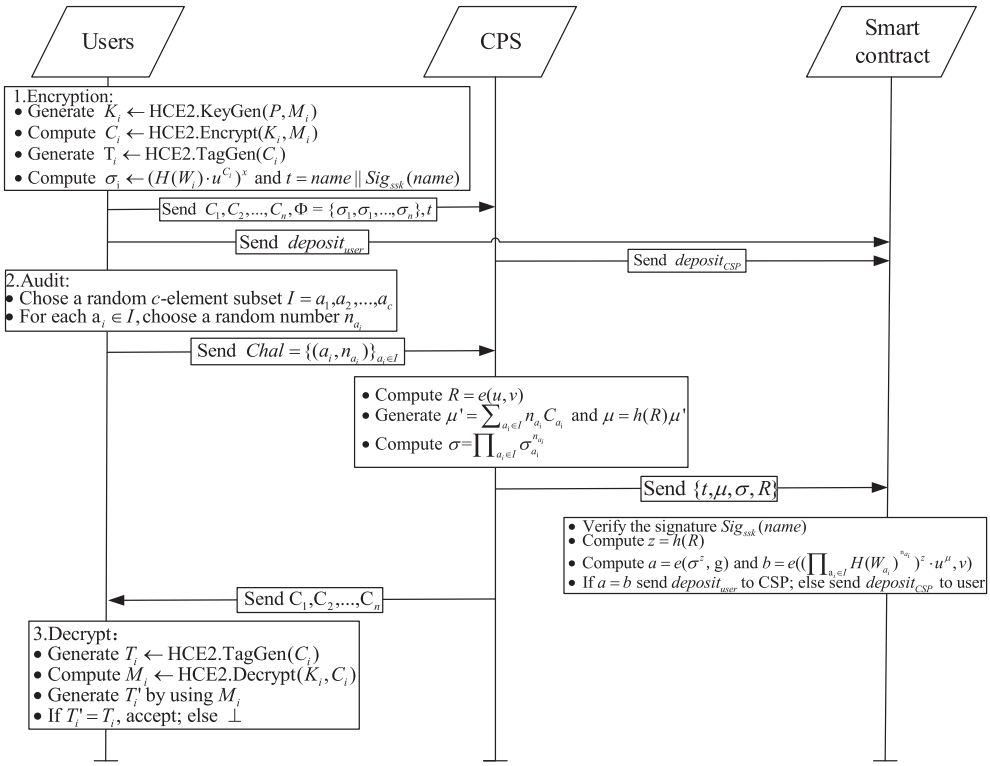

Fig. 3. Overview of our scheme.

– Generate the authenticator $\sigma_i \leftarrow (H(W_i) \cdot u^{c_i})^x \in \mathbb{G}_1$, where $W_i = name||i$ and $name \in \mathbb{Z}_p$ is uniformly and randomly chosen by the user as the identifier of file F. $\Phi = \{\sigma_i\}_{1 \leqslant i \leqslant n}$ denotes the set of authenticators.

– Generate file tag $t = name||Sig_{ssk}(name)$, where $Sig_{ssk}(name)$ is the signature of file F. Then, the user sends ciphertexts $C_1, C_2, \ldots, C_n, \Phi$ and t to the CSP [1].

After receiving the ciphertexts $C_1, C_2, \ldots, C_n$, the CSP makes a comparison between the stored data and the newly uploaded ciphertexts. If the same ciphertext is found, it means that the same ciphertext has been stored in the CSP. Then, the CSP no longer stores the new ciphertext for saving storage space. This efficiently reduces storage and management overhead. Meantime, the user signs a smart contract with the CSP. The user sends $deposit_{user}$ to the smart contract as his deposit and the CSP sends $deposit_{CSP}$ to the smart contract as its deposit. If the integrity verification is passed, the smart contract automatically sends the user's deposit $deposit_{user}$ to the miner as the audit fee and returns the deposit $deposit_{CSP}$ to the CSP. Else, the smart contract sends the CSP's deposit $deposit_{CSP}$ to the user as the penalty and sends $deposit_{user}$ to the miner.

- **Audit**: In order to audit the data integrity of the user's outsourced data, the audit processes are described as follows.

– Based on the current blockhash, a user generates a random c-element subset $I = a_1, a_2, \ldots, a_c$ of set $[1, n]$, where $a_i = f(blockhash||i)$ [2]. For each element $a_i \in I$, the user generates a random number $n_{a_i} = H(blockhash||a_i)$. The metadata $chal$ denotes the positions of the challenge blocks. The user sends $chal = \{(a_i, n_{a_i})\}_{a_i \in I}$ to the CSP.

– After receiving the challenge metadata $chal = \{(a_i, n_{a_i})\}_{i \in I}$, the CSP sets $R = e(u, v) \in \mathbb{G}_T$. Then, the CSP generates a linear combination of sampled blocks $\mu' = \sum_{a_i \in I} n_{a_i} C_{a_i}$ and computes $\mu = h(R)\mu'$, where $h(R) \in \mathbb{Z}_p$. In addition, the CSP generates an aggregated authenticator $\sigma = \prod_{a_i \in I} \sigma_{a_i}^{n_{a_i}} \in \mathbb{G}_1$. Finally, the CSP sends $\{t, \mu, \sigma, R\}$ to the smart contract.

– After receiving the response of challenge $\{t, \mu, \sigma, R\}$, the smart contract executes Algorithm 1.

[1] Regarding the safety of cloud-stored data, a centralized data center will use distributed redundant data storage to protect data safety. Only in a redundant way can the data be reliably protected upon damage. Therefore, our scheme does not deduplicate redundant copies that are used to enhance data security but deduplicate copies that are not used for improving security.

[2] It should be emphasized that the challenge value is calculated by the latest hash value of the blockchain, and the malicious adversary cannot predict the next challenge value. Therefore, our scheme can effectively prevent man-in-the-middle attacks and replay attacks.

Algorithm 1 Auditing of smart contract.

Require: The file tag t and responses of challenge $\{\mu, \sigma, R\}$;
Ensure: Result of integrity audit t;
1: Verify the signature $Sig_{ssk}(name)$ via spk. If the verification fails, let $t = 0$ and break;
2: Compute $z = h(R)$;
3: Compute $a = e(\sigma^z, g)$ and $b = e((\prod_{a_i \in I} H(W_{a_i})^{n_{a_i}})^z \cdot u^{\mu}, v)$;
4: **if** $a = b$
5: Send $deposit_{user}$ to the miner and send $deposit_{CSP}$ to the CSP;
6: Set $t = 1$;
7: **else**
8: Send $deposit_{CSP}$ to the user and send $deposit_{user}$ to the miner;
9: Set $t = 0$;
10: **end if**
11: **return** t.

The correctness of $a = b$ is elaborated as follows:

$$
\begin{aligned}
a = e(\sigma^z, g) &= e(((\prod_{a_i \in I} H(W_{a_i}) \cdot u^{C_{a_i}})^{x \cdot n_{a_i}})^z, g) \\
&= e(((\prod_{a_i \in I} H(W_{a_i})^{n_{a_i}} \cdot u^{n_{a_i} C_{a_i}})^z, g)^x \\
&= e((\prod_{a_i \in I} H(W_{a_i})^{n_{a_i}})^z \cdot u^{\mu z}, v) \\
&= e((\prod_{a_i \in I} H(W_{a_i})^{n_{a_i}})^z \cdot u^{\mu}, v)
\end{aligned}
$$

- **Decrypt**: After downloading the ciphertext $C_1, C_2, \ldots, C_n$, the user performs the following operations.

  – Input the ciphertext C_i into tag generation algorithm and generate the tag $T_i \leftarrow$ **HCE2.TagGen**(C_i).
  – Input the key K_i and plaintext C_i into the decryption algorithm and get the plaintext $M_i \leftarrow$ **HCE2.Decrypt**(K_i, C_i).
  – Generate the tag $T\prime_i$ by using M_i and compare it with T_i. If $T\prime_i = T_i$, the user accepts plaintext M_i; else, rejects it.

4.3. Batch auditing

To improve the efficiency of data auditing, our scheme supports batch data auditing. In the batch auditing scheme, we aggregate the data authenticators of multiple data into one data authenticator. As a result, our scheme can audit multiple tasks at the same time. The batch auditing scheme is described as follows.

- **Setup**: The setup phase is the same as the above auditing scheme, so we omit here.
- **Encrypt**: To upload files $F = F_1, F_2, \ldots, F_s$, a user first splits each file $F_d (1 \leqslant d \leqslant s)$ into a set of chunks $\{M_1^d, M_2^d, \ldots, M_n^d\} (1 \leqslant d \leqslant s)$ [3]. For M_i^d, the user performs the following operations.

  – Execute $K_i^d \leftarrow$ **HCE2.KeyGen**(P, M_i^d) to generate the convergent key K_i^d.
  – Encrypt the plaintext M_i^d by computing $C_i^d \leftarrow$ **HCE2.Encrypt**(K_i^d, M_i^d) and generate tag $T_i^d \leftarrow$ **HCE2.TagGen**(C_i^d) by using the ciphertext C_i^d.
  – Generate the authenticator $\sigma_i^d \leftarrow (H(W_i^d) \cdot u^{C_i^d})^x$, where $W_i^d = name_d || i$ and $name_d$ is uniformly and randomly chosen by the user from $\mathbb{Z}_p$ as the identifier of file F_d. We use $\Phi_d = \{\sigma_i^d\}_{1 \leqslant i \leqslant n}$ to denote the set of authenticators.
  – Use key ssk to generate file tag $t_d = name_d || Sig_{ssk}(name_d)$, where $Sig_{ssk}(name_d)$ is the signature of the file F_d. Then, the user sends ciphertexts $C_d = \{C_1^d, C_2^d, \ldots, C_n^d\}$ and verification metadata Φ_d, t_d to the CSP.

[3] We assume that each file F_d has the same number of block n in our scheme.

After receiving the ciphertext $C_1, C_2, \ldots, C_s$, the CSP signs a smart contract with the user. The user sends $deposit_{user}$ to the smart contract as his deposit and the CSP sends $deposit_{CSP}$ to the smart contract as its deposit. If the integrity verification is passed, the smart contract sends the user's deposit $deposit_{user}$ to the miner as the audit fee and returns $deposit_{CSP}$ to the CSP. Else, the smart contract sends the CSP's deposit $deposit_{CSP}$ to the user as the penalty and sends $deposit_{user}$ to the miner.

Algorithm 2 Batch auditing of smart contract.

Require: The file tag $t_1, t_2, \ldots, t_s$ and responses of challenge $\{\mu_1, \mu_2, \ldots, \mu_s, \sigma_1, \sigma_2, \ldots, \sigma_s, R\}$;
Ensure: Result of integrity audit t;
1: Verify each signature $Sig_{ssk}(name_d)(1 \leqslant d \leqslant s)$ via spk. If any verification fails, let $t = 0$ and break;
2: Compute $z = h(R)$;
3: Compute $a = e(\prod_{d=1}^{s} \sigma_d^z, g)$;
4: Compute $b = \prod_{d=1}^{s} e((\prod_{a_i \in I} H(W_{a_i}^d)^{n_{a_i}})^z \cdot u^{\mu_d}, v)$;
5: **if** $a = b$
6: Send $deposit_{user}$ to the miner and send $deposit_{CSP}$ to the CSP;
7: Set $t = 1$;
8: **else**
9: Send $deposit_{CSP}$ to the user and send $deposit_{user}$ to the miner;
10: Set $t = 0$;
11: **end if**
12: **return** t.

- **Audit**: In order to audit the data integrity of the user's outsourced data, the audit processes are described as follows.

  – Based on the current blockhash, a user generates a random c-element subset $I = a_1, a_2, \ldots, a_c$ of set $[1, n]$, where $a_i = f(blockhash||i)$. For each element $a_i \in I$, the user generates a random number $n_{a_i} = H(blockhash||a_i)$. The metadata $chal$ specifies the positions of the challenge blocks. The user sends $chal = \{(a_i, n_{a_i})\}_{a_i \in I}$ to the CSP.
  – After receiving the metadata of challenge $chal = \{(a_i, n_{a_i})\}_{i \in I}$, the CSP sets $R = e(u, v) \in \mathbb{G}_T$. For each $C_d (1 \leqslant d \leqslant s)$, the CSP generates a linear combination of sampled blocks $\mu_d = \sum_{a_i \in I} n_{a_i} C_{a_i}^d$ and computes $\mu_d = h(R)\mu_d$, where $h(R) \in \mathbb{Z}_p$. In addition, the CSP generates an aggregated authenticator $\sigma_d = \prod_{a_i \in I} (\sigma_{a_i}^d)^{n_{a_i}} \in \mathbb{G}_1$. Finally, the CSP sends $\{\mu_1, \mu_2, \ldots, \mu_s, \sigma_1, \sigma_2, \ldots, \sigma_s, R\}$ to the smart contract.
  – After receiving the response of challenge $\{t_1, t_2, \ldots, t_s, \mu_1, \mu_2, \ldots, \mu_s, \sigma_1, \sigma_2, \ldots, \sigma_s, R\}$, the smart contract executes Algorithm 2.

5. Analysis of our proposed scheme

5.1. Security Analysis

In this subsection, we give a comprehensive analysis to demonstrate the security of the proposed scheme in terms of data privacy, storage correctness, batch auditing and data consistency. We assume that the underlying basic tools are secure, which include homomorphic linear authenticator, one-way hash function, Hash-and-CE-2 scheme and symmetric encryption scheme. These assumptions ensure the security of our scheme. During the data uploading phase, users use a symmetric encryption algorithm (such as AES-256) to encrypt the data and upload it to the CSP. Therefore, the **data privacy** of outsourcing data can be protected.

5.1.1. Storage correctness

We prove that the CSP cannot generate valid proof of challenge without honest storing the entire original data as follows.

Theorem 5.1. *Assume that the computational Diffie-Hellman problem is hard in bilinear groups and the digital signature scheme is existentially unforgeable, in random oracle model, unless the adversary correctly generates the proof (t, μ, σ, R) by using challenge chal and ciphertext C, the probability that the auditor accepts this proof is negligible.*

Proof 1. In random oracle model, we assume that there exists an extractor μ. With the valid signature σ and μ, our theorem follows the previous schemes [13,29].

The extractor can control the random oracle $h(\cdot)$. Then, the extractor is able to answer the hash query issued by the CSP. We assume that the extractor is an adversary. To response a challenge $z = H(R)$ of the extractor, the CSP outputs $\{\sigma, \mu, R\}$ such that the following equation is satisfied:

$$e(\sigma^z, g) = e((\prod_{a_i \in I} H(W_{a_i})^{n_{a_i}})^z \cdot u^{\mu}, v). \tag{1}$$

Assume that the extractor can reverse a CSP in the execution of the protocol to the point just before the challenge $h(R)$ is given. Then, the extractor can set $h(R)$ to be $z^* \neq z$. The CSP returns $\{\sigma, \mu^*, R\}$ such that:

$$e(\sigma^{z^*}, g) = e((\prod_{a_i \in I} H(W_{a_i})^{n_{a_i}})^{z^*} \cdot u^{\mu^*}, v). \tag{2}$$

Recalled that $\sigma_i \leftarrow (H(W_i) \cdot u^{C_i})^x$. We divide (1) by (2):

$$e(\sigma^{z-z^*}, g) = e((\prod_{a_i \in I} H(W_{a_i})^{n_{a_i}})^{z-z^*} \cdot u^{\mu-\mu^*}, v)$$

$$e(\sigma^{z-z^*}, g) = e((\prod_{a_i \in I} H(W_{a_i})^{n_{a_i}})^{z-z^*}, g^x) e(u^{\mu-\mu^*}, g^x)$$

$$\sigma^{z-z^*} = (\prod_{a_i \in I} H(W_{a_i})^{n_{a_i}})^{x(z-z^*)} \cdot u^{x(\mu-\mu^*)}$$

$$(\prod_{a_i \in I} \sigma_{a_i}^{n_{a_i}})^{z-z^*} = (\prod_{a_i \in I} H(W_{a_i})^{n_{a_i}})^{x(z-z^*)} \cdot u^{x(\mu-\mu^*)}$$

$$u^{x(\mu-\mu^*)} = (\prod_{a_i \in I} (\sigma_{a_i} / H(W_{a_i})^x)^{n_{a_i}})^{z-z^*}$$

$$u^{x(\mu-\mu^*)} = (\prod_{a_i \in I} (u^{xC_{a_i}})^{n_{a_i}})^{z-z^*}$$

$$\mu - \mu^* = (\sum_{a_i \in I} C_{a_i} n_{a_i}) \cdot (z - z^*)$$

$$\sum_{a_i \in I} C_{a_i} n_{a_i} = (\mu - \mu^*)/(z - z^*).$$

Finally, the extractor can obtain $\{\sigma, \mu\prime = (\mu - \mu^*)/z - z^*\}$ as a valid response of basic proofs of retrievability scheme [29]. □

Theorem 5.2. *The proposed scheme guarantees correctness of batch auditing.*

Proof 2. The batch auditing involves s challenges. The correctness of batch auditing is proved as follows:

$$a = e(\prod_{d=1}^{s} \sigma_d^z, g) = e(\prod_{d=1}^{s} ((\prod_{a_i \in I} H(W_{a_i}^d) \cdot u^{C_{a_i}^d})^{x \cdot n_{a_i}})^z, g)$$

$$= e(\prod_{d=1}^{s} ((\prod_{a_i \in I} H(W_{a_i}^d)^{n_{a_i}} \cdot u^{n_{a_i} C_{a_i}^d})^z, g)^x$$

$$= e(\prod_{d=1}^{s} (\prod_{a_i \in I} H(W_{a_i}^d)^{n_{a_i} z} \cdot u^{\mu_d z}), v)$$

$$= e(\prod_{d=1}^{s} (\prod_{a_i \in I} H(W_{a_i}^d)^{n_{a_i} z} \cdot u^{\mu_d}), v)$$

$$= \prod_{d=1}^{s} e(\prod_{a_i \in I} H(W_{a_i}^d)^{n_{a_i} z} \cdot u^{\mu_d}, v)$$

Therefore, our scheme can guarantee the correctness of batch auditing. □

Finally, we analyze the **data consistency** of the proposed scheme. In downloading and decryption processes, a user's data may be corrupted by various network and software failures. To verify the correctness of decrypted data, our scheme adopts a consistent detection mechanism. After downloading the ciphertext $C = C_1 || T$ from the CSP, the user u first generates the tag T by using the ciphertext $C = C_1 || T$. Then, the user decrypts the ciphertext C_1 and generates the plaintext $M\prime$ by using the file

key K. Finally, the user re-generates tag $T\prime$ by using the plaintext $M\prime$. After generating the tag T and $T\prime$, the user u checks whether $T\prime = T$. If $T\prime = T$, the user u accepts the message. Otherwise, the user drops the message. Therefore, our scheme guarantees data consistency.

5.2. Comparison

Table 1 presents the comparison among four data auditing schemes, which consists of the provable data possession (PDP) scheme [11], Wang et al.'s scheme [13], Li et al.'s scheme [16] and our scheme, in terms of probabilistic audit, batch auditing, privacy-preserving, data deduplication, and fair arbitration.

All the data auditing schemes support probabilistic audit and batch auditing, which can reduce the computing and management overhead of the CSP and users. PDP cannot guarantee the privacy-preserving so that the sensitive information of users' data will be leaked. By using homomorphic linear authenticator and random masking, Wang et al.'s scheme [13] can prevent TPA from learning any information of the users' sensitive data stored on the CSP during the data auditing. By combining data deduplication with data auditing, Li et al.'s [16] scheme and our scheme allow users to encrypt the sensitive data before they upload the sensitive data to the CSP. Therefore, Li et al.'s scheme and our scheme also protect the privacy information of outsourced data.

Li et al.'s scheme uses the convergence encryption scheme to achieve data deduplication. Thus, this scheme cannot protect data consistency. In our scheme, we use the HCE2 algorithm to achieve data deduplication. In the HCE2 scheme, an additional tag checking mechanism is adopted, which can effectively discover whether the decrypted data is equal to the original plaintext. Besides, the previous schemes do not consider the problem of fair arbitration. After finding that the CSP has destroyed the users' sensitive data, the users still cannot effectively obtain corresponding compensation, which is extremely unfair to the users. To solve this problem, we apply a smart contract in our scheme, which can automatically execute data auditing without relying on the TPA. Moreover, when data integrity is compromised, our scheme can punish malicious CSP and compensate users whose data integrity is destroyed. Therefore, our scheme can realize fair arbitration.

Table 2 presents the computational cost of Wang et al.'s scheme [13] and our scheme. Bil denotes the operation of the bilinear map. Mul denotes the operation of multiplication. Add denotes the operation of addition. Exp denotes the exponent operation. n denotes the number of data blocks being challenged and k denotes the number of auditing tasks during the phase of batch auditing. In the phase of proof generation, our scheme reduces one exponent operation and one additional operation compared with Wang et al.'s scheme. In the phase of batch proof generation, our scheme reduces k addition operations and one exponent operation compared with Wang et al.'s scheme. In the phase of integrity verification and batch verification, our scheme reduces one multiplication operation compared with Wang et al.'s scheme. The results of the comparison show that our scheme has less computational cost than Wang et al.'s scheme.

6. Performance evaluation

In this section, we provide a thorough experimental evaluation of our scheme. We implement our scheme in the Java programming language by using the JPBC library v2.0.0 and Solidity v0.5.1. The test environment is Intel(R) Core(TM) i7-7820HK CPU 2.90 GHz 16.0 GB RAM, Windows 10. We test the solidity program in the Remix-IDE [44]. To verify the performance of our scheme and compare it with the Wang et al.'s scheme [13] and Li et al.'s scheme [16], we mainly use the MLE key generation time, proof generation time, batch proof generation time, gas cost of integrity verification and gas cost of batch verification as the evaluation metrics. Proof generation time and batch proof generation time are the time that the CSP uses

Table 1
Comparison of Data Auditing Schemes.

Scheme	PDP [11]	Wang et al. [13]	Li et al. [16]	Our scheme
Probabilistic audit	✓	✓	✓	✓
Batch auditing	✓	✓	✓	✓
Privacy-preserving	×	✓	✓	✓
Data deduplication	×	×	✓	✓
Fair arbitration	×	×	×	✓

Table 2
Computational Cost of Data Auditing Schemes.

Scheme	Wang et al. [13]	Our scheme
Proof generation	Bil + (n + 1) Exp + 2nMul + nAdd	Bil + nExp + 2nMul + (n − 1) Add
Batch proof generation	k(2nMul + nAdd + nExp)+Bil + Exp	k(2nMul+(n − 1) Add + nExp)+Bil
Integrity verification	2Bil+(n + 3) Exp+(n + 1) Mul	2Bil+(n + 3) Exp + nMul
Batch verification	(k + 1) Bil+(kn + 2 k − 1) Mul+(n + 3) kExp	(k + 1) Bil+(kn + 2 k − 2) Mul+(n + 3) kExp

to generate proof of integrity audit and generate batch proof of integrity audit. Gas cost of integrity verification and batch verification are the cost that the smart contract uses to verify the integrity and verify the result of batch verification. According to [13,16], we set the number of challenge blocks to 300 and 460. The data size ranges from 1 MB to 10 MB. The results of performance evaluation are the average of 20 experiments. The test time does not include the communication time between the user and the CSP.

6.1. MLE key generation performance

We first measure the performance of the MLE key generation of our scheme and Li et al.'s scheme [16]. To test the convergent key generation time, we use the SHA-128 and SHA-256 as the hash functions, where the data size from 1 MB to 10 MB. We set the size of the block to 1 KB and generate an encryption key for each block. The detail of the key generation time is shown in Fig. 4. The results show that the MLE key generation time of our scheme is almost the same as Li et al.'s scheme.

6.2. Encryption and decryption performance

Different from the existing data auditing schemes [11–13,15], our scheme generates a homomorphic linear authenticator of the outsourced data on the ciphertext. Although our scheme introduces encryption and decryption time, our scheme can

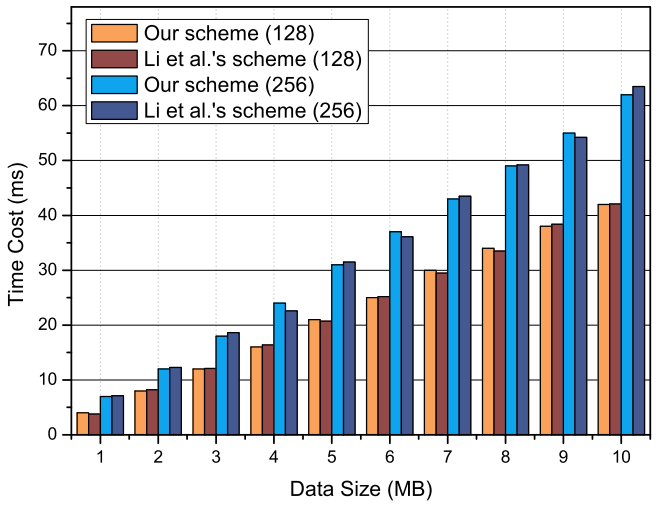

Fig. 4. MLE key generation time.

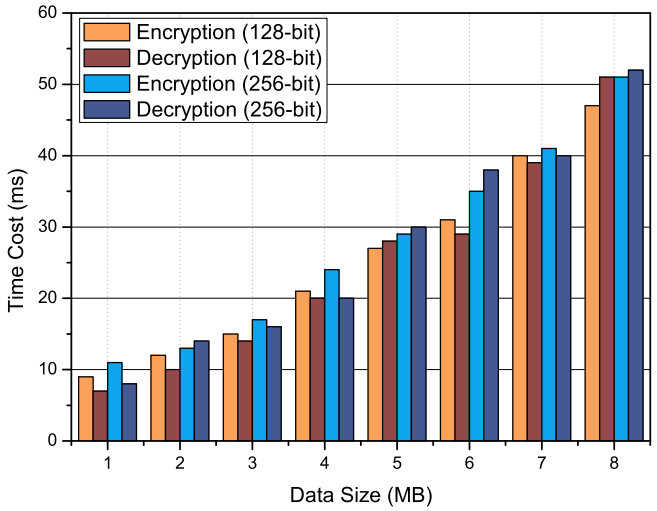

Fig. 5. Encryption and decryption time.

support data deduplication and prevent the auditor from learning any knowledge about the users' sensitive data. To test the encryption and decryption time, we use the AES-128 and AES-256 algorithms in our scheme, where the data size from 1 MB to 8 MB. The cost of encryption and decryption time is shown in Fig. 5.

6.3. Upload performance

To evaluate the effect of data deduplication, we test the data uploading time of our scheme and Wang et al.'s scheme [13]. We set the data size and the key size to 1 MB and 128-bit, respectively. The data uploading time includes MLE key generation time, data encryption time and tag generation time. The evaluation results are shown in Fig. 6. The results show that the data uploading time in our scheme is almost the same as Wang et al.'s scheme.

Our scheme focuses on achieving data auditing without any third-party and fair arbitration, yet we observe the problems of user revocation. To achieve dynamic user management, we can introduce an access control algorithm or CP-ABE method to further improve the flexibility of our scheme. We pose this problem as future work.

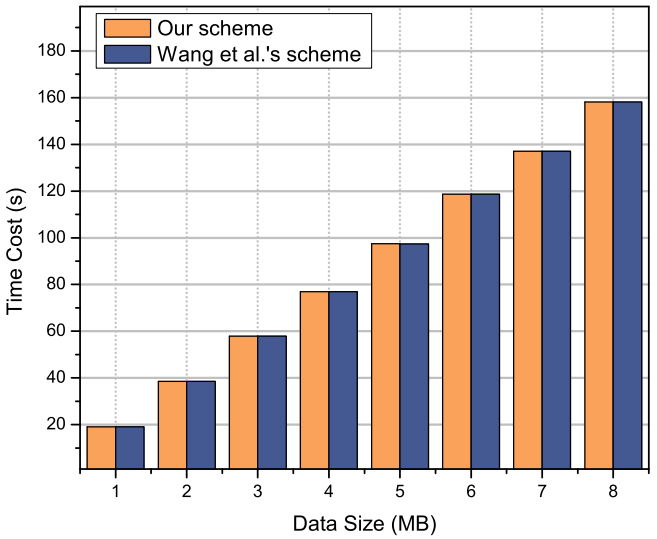

Fig. 6. Data uploading time.

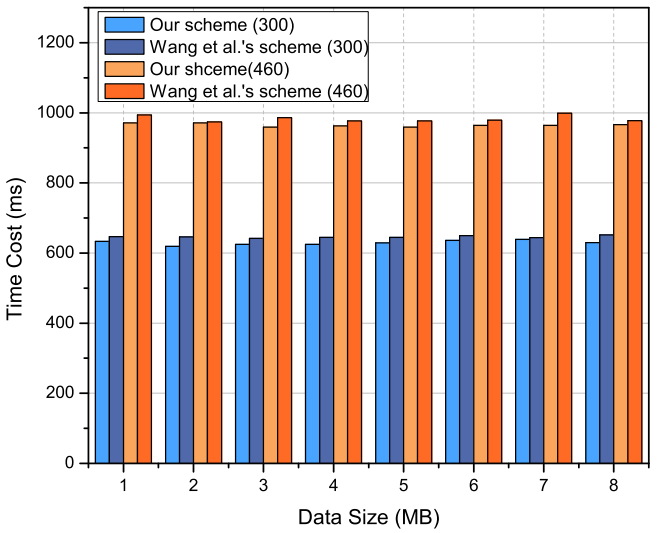

Fig. 7. Proof generation time.

6.4. Proof generation and batch proof generation performance

We measure the proof generation and batch proof generation time of our scheme and Wang et al.'s scheme [13]. Since both our scheme and Wang et al.'s scheme use a probabilistic integrity audit algorithm, the proof generation time for these two schemes is constant. Since our scheme does not require a random masking method to achieve privacy-preserving public auditing, the proof generation and batch proof generation time of our scheme is shorter than Wang et al.'s scheme. The evaluation results are shown in Fig. 7.

To improve the efficiency of data auditing, our scheme and Wang et al.'s scheme support batch proof generation. To test the performance of batch proof generation, we set the data size to 10 MB. The batch auditing time is shown in Fig. 8.

6.5. Gas cost of integrity verification and batch verification

To achieve fair arbitration, we use the smart contract to verify the audit results of outsourced data automatically. To test the cost of verifying, we use the solidity to program the smart contract and experiment on the Ethereum blockchain. We measure the gas cost of integrity verification and batch verification of our scheme and Wang et al.'s scheme [13]. Since our scheme does not require a random masking method to support privacy-preserving public auditing, the gas cost of our scheme is less than Wang et al.'s scheme. The evaluation results of the gas cost of integrity verification and batch verification

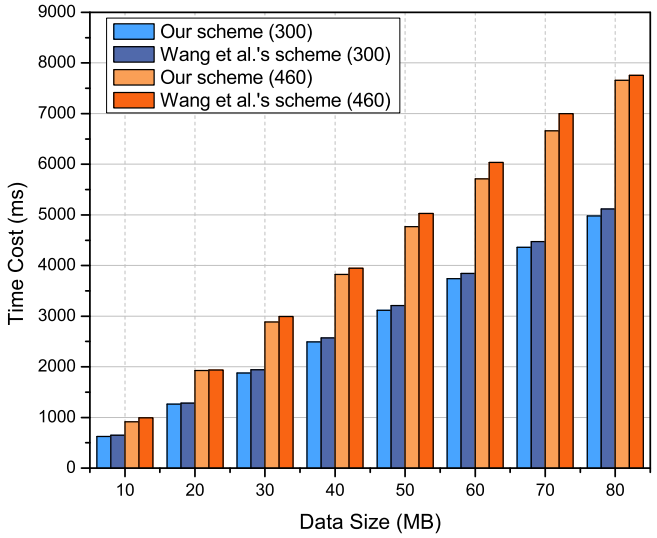

Fig. 8. Batch proof generation time.

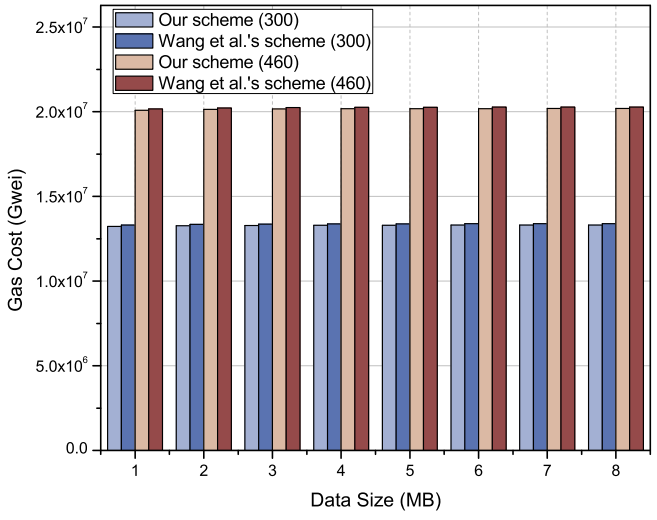

Fig. 9. Gas cost of integrity verification.

are shown in Fig. 9 and Fig. 10. It can be known from Fig. 9 that when the number of randomly selected data blocks is constant, the overhead of the miner to verify the data integrity of the outsourcing data does not increase with the increase of the data size. The consumed gas value is 1.33×10^7 Gwei (0.013 Ether) when verifying 300 randomly selected data blocks, and the consumed gas value is 2.01×10^7 Gwei (0.020Ether) when verifying 460 randomly selected data blocks.

6.6. Probabilistic audit performance

As previous work [11,13] showed that if t percent of the total data is corrupted by the CSP and every challenge block for auditing is chosen uniformly, then random sampling c blocks can realize the detection probability $P_x = 1 - (1 - t)^c$, where the number of the challenged data blocks is denoted by c and the total number of data blocks is denoted by n. In Fig. 11 and Fig. 12, we show P_x as a function of n and c for two values of t. If $t = 1\%$ of the total number of block n, then the data auditing can achieve at least 95% and 99% probability to find the misbehavior by asking 300 blocks and 460 blocks, respectively. If the sampling strategies are rational, our scheme can realize efficient data auditing and communication overhead can be reduced.

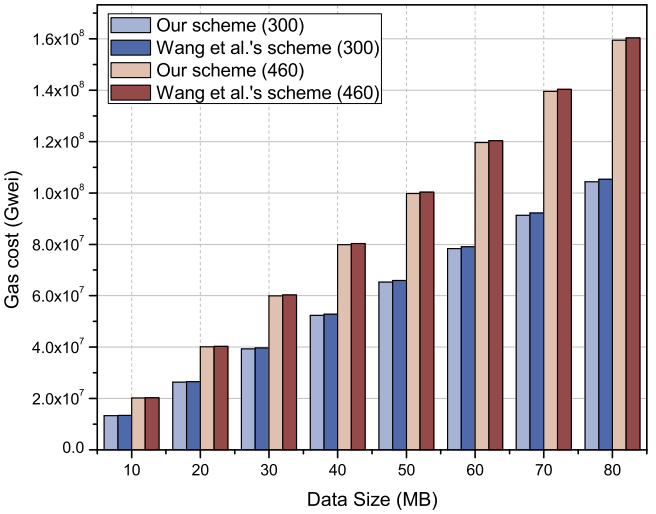

Fig. 10. Gas cost of batch verification.

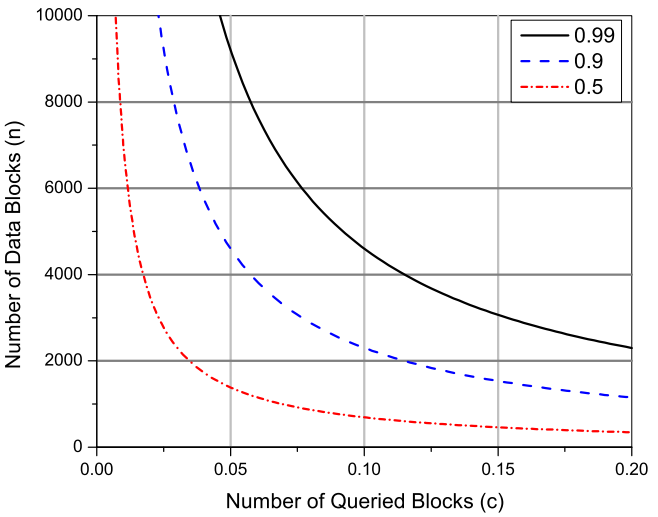

Fig. 11. $t = 1\%$ of n.

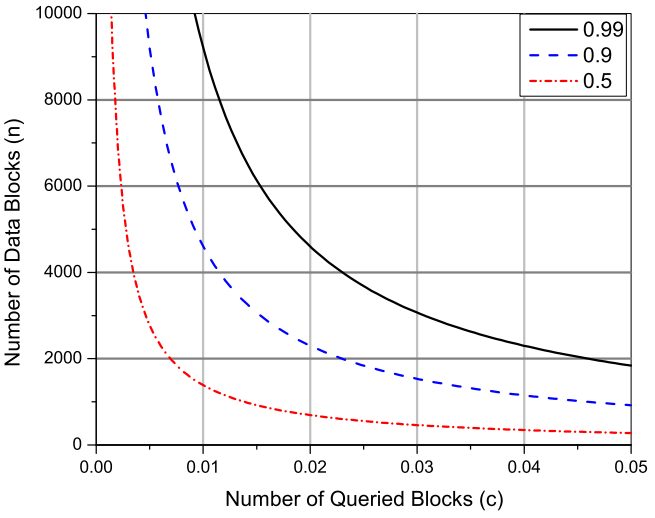

Fig. 12. $t = 5\%$ of n.

7. Conclusion

In this paper, we propose a blockchain-based public auditing and secure deduplication scheme with fair arbitration. By employing a homomorphic linear authenticator and smart contract, our scheme supports data auditing without relying on any third-party auditor. Besides, when the users' data integrity is compromised, our scheme can automatically punish the malicious CSP and compensate users whose data integrity is destroyed. In addition, our scheme supports data deduplication on encrypted data, which reduces the storage overhead and management cost of the CSP. We also prove that our scheme can achieve the desired security goals and provide detailed experimental results. The performance analysis shows that our scheme is efficient. In future work, we plan to address the problem of dynamic user management and evaluate how our scheme performs for other storage workloads.

CRediT authorship contribution statement

Haoran Yuan: Writing - original draft, Methodology, Software, Validation. **Xiaofeng Chen:** Methodology. **Jiaming Yuan:** Software. **Hongyang Yan:** Conceptualization. **Willy Susilo:** Supervision.

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.

Acknowledgment

This work is supported by National Cryptography Development Fund (No. MMJJ20180110) and Shandong Provincial Key Research and Development Program of China (No. 2019JZZY020129).

References

[1] Geek, Google suffers data loss as data center gets hit by lightning, URL:https://www.geek.com/, 2015 (accessed 21, May 2019).
[2] Telegraph, Personal details of 50 million turkish citizens leaked, URL:https://www.telegraph.co.uk/, 2016 (accessed 11, May 2019).
[3] Kromtech, Patient home monitoring service leaks private data, URL:https://kromtech.com/blog/security-center/, 2017 (accessed 21, June 2019).
[4] X. Chen, J. Li, J. Ma, Q. Tang, W. Lou, New algorithms for secure outsourcing of modular exponentiations, IEEE Trans. Parallel Distrib. Syst. 25 (9) (2014) 2386–2396.
[5] G. Xu, H. Li, Y. Dai, K. Yang, X. Lin, Enabling efficient and geometric range query with access control over encrypted spatial data, IEEE Trans. Inform. Forensics Security 14 (4) (2019) 870–885.
[6] S. Xu, G. Yang, Y. Mu, Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation, Inf. Sci. 479 (2019) 116–134.
[7] X. Chen, J. Li, X. Huang, J. Ma, W. Lou, New publicly verifiable databases with efficient updates, IEEE Trans. Dependable Sec. Comput. 12 (5) (2015) 546–556.
[8] X. Wang, J. Ma, Y. Miao, X. Liu, R. Yang, Privacy-preserving diverse keyword search and online pre-diagnosis in cloud computing, IEEE Transactions on Services Computing (2019), https://doi.org/10.1109/TSC.2019.2959775.
[9] J. Ning, J. Xu, K. Liang, F. Zhang, E. Chang, Passive attacks against searchable encryption, IEEE Trans. Inform. Forensics Security 14 (3) (2019) 789–802.

[10] H. Yuan, X. Chen, J. Li, T. Jiang, J. Wang, R. Deng, Secure cloud data deduplication with efficient re-encryption, IEEE Transactions on Services Computing. (2019), https://doi.org/10.1109/TSC.2019.2948007.
[11] G. Ateniese, R.C. Burns, R. Curtmola, J. Herring, L. Kissner, Z.N.J. Peterson, D.X. Song, Provable data possession at untrusted stores, in: Proceedings of the Conference on Computer and Communications Security, CCS, 2007, pp. 598–609.
[12] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, Enabling public auditability and data dynamics for storage security in cloud computing, IEEE Trans. Parallel Distrib. Syst. 22 (5) (2011) 847–859.
[13] C. Wang, S.S.M. Chow, Q. Wang, K. Ren, W. Lou, Privacy-preserving public auditing for secure cloud storage, IEEE Trans. Computers 62 (2) (2013) 362–375.
[14] Y. Zhang, R.H. Deng, X. Liu, D. Zheng, Blockchain based efficient and robust fair payment for outsourcing services in cloud computing, Inf. Sci. 462 (2018) 262–277.
[15] J. Wang, X. Chen, X. Huang, I. You, Y. Xiang, Verifiable auditing for outsourced database in cloud computing, IEEE Trans. Comput. 64 (11) (2015) 3293–3303.
[16] J. Li, J. Li, D. Xie, Z. Cai, Secure auditing and deduplicating data in cloud, IEEE Trans. Comput. 65 (8) (2016) 2386–2396.
[17] X. Liu, W. Sun, W. Lou, Q. Pei, Y. Zhang, One-tag checker: Message-locked integrity auditing on encrypted cloud deduplication storage, in: IEEE Conference on Computer Communications, INFOCOM, 2017, pp. 1–9.
[18] Y. Wu, Z.L. Jiang, X. Wang, S. Yiu, P. Zhang, Dynamic data operations with deduplication in privacy-preserving public auditing for secure cloud storage, in: IEEE International Conference on Computational Science and Engineering, 2017, pp. 562–567.
[19] IDC, Rich data and the increasing value of the internet of things, URL:https://www.emc.com/, 2014 (accessed 1, August 2019).
[20] Seagate, Dataage white paper: the digitization of the world, URL:https://www.seagate.com/cn/zh/our-story/data-age-2025/, 2018 (accessed 23, January 2019).
[21] W.J. Bolosky, S. Corbin, D. Goebel, J.R. Douceur, Single instance storage in windows 2000, in: Conference on Usenix Windows Systems Symposium, 2000.
[22] GoogleDrive, Backup and sync, free and safe download, URL:http://drive.google.com, 2012 (accessed 21, March 2019).
[23] Memopal, Back up all your files with memopal online backup, URL:https://www.memopal.com, 2018 (accessed 16, July 2019).
[24] Dropbox, Store, sync and share your files online, URL:http://www.dropbox.com, 2007 (accessed 23, January 2018).
[25] M. Dutch, Understanding data deduplication ratios, in: SNIA Data Management, Forum (2008) 1–13.
[26] J.R. Douceur, A. Adya, W.J. Bolosky, P. Simon, M. Theimer, Reclaiming space from duplicate files in a serverless distributed file system, ICDCS, 2002, pp. 617–624.
[27] M. Bellare, S. Keelveedhi, T. Ristenpart, Message-locked encryption and secure deduplication, in: Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vol. 28, 2013, pp. 296–312.
[28] A. Juels, Jr Kaliski B. S., PORs: proofs of retrievability for large files, in: Proceedings of the ACM Conference on Computer and Communications Security, CCS, 2007, pp. 584–597.
[29] H. Shacham, B. Waters, Compact proofs of retrievability, in: Advances in Cryptology – ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security December 7–11, 2008. Proceedings, Melbourne, Australia, 2008, pp. 90–107.
[30] S. Keelveedhi, M. Bellare, T. Ristenpart, Dupless: Server-aided encryption for deduplicated storage, in: Proceedings of the 22th USENIX Security Symposium, 2013, pp. 179–194.
[31] D. Chaum, Blind signatures for untraceable payments, in: Advances in Cryptology, Proceedings of CRYPTO (1982) 199–203.
[32] J. Li, X. Chen, M. Li, J. Li, P.P. Lee, W. Lou, Secure deduplication with efficient and reliable convergent key management, IEEE Trans. Parallel Distrib. Syst. 25 (6) (2014) 1615–1625.
[33] Y. Shin, K. Kim, Equality predicate encryption for secure data deduplication, in, Proc. Conf. Inf. Security Cryptol. (2012) 64–70.
[34] J. Wang, X. Chen, J. Li, K. Kluczniak, M. Kutylowski, A new secure data deduplication approach supporting user traceability, in: 10th International Conference on Broadband and Wireless Computing, Communication and Applications, BWCCA, 2015, pp. 120–124.
[35] J. Li, Y.K. Li, X. Chen, P.P.C. Lee, W. Lou, A hybrid cloud approach for secure authorized deduplication, IEEE Trans. Parallel Distrib. Syst. 26 (5) (2015) 1206–1216.
[36] H. Yuan, X. Chen, T. Jiang, X. Zhang, Z. Yan, Y. Xiang, Dedupdum: Secure and scalable data deduplication with dynamic user management, Inf. Sci. 456 (2018) 159–173.
[37] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, URL:https://bitcoin.org/bitcoin.pdf, 2008 (accessed 4, January 2020).
[38] C. Yang, X. Chen, Y. Xiang, Blockchain-based publicly verifiable data deletion scheme for cloud storage, J. Network Computer Appl. 103 (2018) 185–193.
[39] H. Huang, X. Chen, Q. Wu, X. Huang, J. Shen, Bitcoin-based fair payments for outsourcing computations of fog devices, Future Generation Comp. Syst. 78 (2018) 850–858.
[40] A. Küpçü, Official arbitration with secure cloud storage application, Comput. J. 58 (4) (2015) 831–852.
[41] H. Jin, H. Jiang, K. Zhou, Dynamic and public auditing with fair arbitration for cloud data, IEEE Trans. Cloud Computing 6 (3) (2018) 680–693.
[42] V. Buterin, Ethereum white paper, URL:https://www.mendeley.com/, 2014 (accessed 27, September 2019).
[43] Y. Zhang, X. Lin, C. Xu, Blockchain-based secure data provenance for cloud storage, in: Information and Communications Security - 20th International Conference, ICICS, 2018, pp. 3–19.
[44] Ethereum, Browser-only ethereum ide and runtime environment, URL:https://remix.ethereum.org, 2018 (accessed 1, January 2020).