Blockchain-Based Decentralized Public Auditing for Cloud Storage

Jiangang Shu, *Member, IEEE,* Xing Zou, Xiaohua Jia, *Fellow, IEEE,* Weizhe Zhang, *Senior Member, IEEE,* and Ruitao Xie, *Member, IEEE*

Abstract—Public auditing schemes for cloud storage systems have been extensively explored with the increasing importance of data integrity. A third-party auditor (TPA) is introduced in public auditing schemes to verify the integrity of outsourced data on behalf of users. To resist malicious TPAs, many blockchain-based public verification schemes have been proposed. However, existing auditing schemes rely on a centralized TPA, and they are vulnerable to tempting auditors who may collude with malicious blockchain miners to produce biased auditing results. In this paper, we propose a blockchain-based decentralized public auditing (BDPA) scheme by utilizing a decentralized blockchain network to undertake the responsibility of a centralized TPA, and also mitigate the influence of tempting auditors and malicious blockchain miners by taking the concept of decentralized autonomous organization (DAO). A detailed security analysis shows that BDPA can preserve data integrity against tempting auditors and malicious blockchain miners. A comprehensive performance evaluation demonstrates that BDPA is feasible and scalable.

Index Terms—Cloud storage, public integrity auditing, identity-based cryptography, blockchain.

1 Introduction

Cloud storage has attracted extensive attention from both academic and industrial research communities for its huge advantages of costs, performance and management [1]–[3]. Cloud users can reduce the expenditure on software, hardware and services by storing data on public cloud servers. And meanwhile they can access outsourced data efficiently and remotely over the Internet without having to stay nearby their computers. Nowadays, more and more users choose to migrate their local data to the cloud storage that is managed by professional cloud service providers (CSPs) such as Amazon's cloud and Google's cloud [4], [5].

Although cloud storage brings a number of benefits for cloud users, data outsourcing has also incurred many critical security issues [6]. One of the most important security concerns is data integrity. In reality, outsourced data may be corrupted because cloud servers may suffer from external rival attacks and internal hardware or software failures [7], [8]. In addition, a cloud server is an independent and untrusted administrative entity and it may delete some cloud data that users have never accessed to save storage space, or hide data loss events to maintain its reputation [9], [10]. Unfortunately, most cloud users often delete locally stored backup data after uploading their data to the cloud server. Due to these abovementioned factors, it is important for users to audit the integrity of their outsourced data periodically.

To ensure the integrity of outsourced data, various cloud storage auditing techniques have been popularly employed [11]–[16]. In traditional public auditing schemes, users often authorize a TPA to conduct public auditing of their outsourced data periodically [17], [18]. The trustworthy TPA can also provide users with dependable auditing results and reduce users' communication and computation burden [19]. However, in traditional public auditing schemes, the TPA is assumed to be honest and reliable, which is a strong assumption in reality because it is quite possible for the auditor to be corrupted [20]. For example, an irresponsible auditor may always produce a valid auditing report without performing the verification process to save the computation costs [21]. To thwart malicious auditors, many blockchain-based public auditing schemes have been proposed [20]–[23]. In most of existing blockchain-based schemes, blockchain technology [24]–[26] is utilized as a secure source of time-dependent pseudorandomness and undeniable storage evidence. Specifically, auditors usually extract block values such as *Nonce* and *BlockHash* from the blockchain to generate random challenge messages which contain indexes of selected data blocks. They also generate log files for auditing procedures and store hash values of log entries in the blockchain.

However, most existing blockchain-based schemes suffer from tempting auditors and have a centralization problem.

- *This work was supported in part by the Natural Science Foundation of China under Grant 61732022, Grant 61802263, Grant 61672195, and Grant 61572370, in part by the Research Grants Council of Hong Kong under Grant CityU 11202419, in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2019B010136001, and in part by the Peng Cheng Laboratory Project of Guangdong Province under Grant PCL2018KP004 and Grant PCL2018KP005. (Corresponding author: Xiaohua Jia.)*
- *J. Shu is with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: shujg@pcl.ac.cn).*
- *X. Zou is with School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518000, China (e-mail: amyzouxing@outlook.com).*
- *X. Jia is with Department of Computer Science, City University of Hong Kong, Kowloon Tong, Hong Kong 999077, China (e-mail: csjia@cityu.edu.hk).*
- *W. Zhang is with School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China, and also with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: wzzhang@hit.edu.cn).*
- *R. Xie is with College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518000, China (e-mail: drtxie@gmail.com).*

In these schemes, the TPA often utilizes the unpredictable values in a proof-of-work (PoW) blockchain where the block values are generally determined by miners. In reality, a malicious auditor can incentivize malicious blockchain miners to ignore newly mined blocks if the generated challenge messages contain certain data blocks. We call this malicious auditor a *tempting auditor*. This biased behavior has non-negligible influence on the auditing results. Moverover, most exisiting blockchain-based schemes lack rigorous proof for the randomness and reliablity of blockchain values. Furthermore, they have a centralized TPA which is required to compute some basic materials (e.g., challenge messages) and verify the cloud server's proof information by computing bilinear mappings. Once the TPA is compromised because of external attacks or internal failures, the auditing procedures can not work normally and may generate incorrect auditing results. In a real-world situation, the availability of auditing services can not be always guaranteed. Therefore, the centralized TPA in existing blockchain-based schemes is vulnerable to a single point of failure (SPOF).

In this paper, we propose a new blockchain-based decentralized public auditing (BDPA) scheme to mitigate the limitations of the tempting and centralized auditor. In BDPA, we explore the idea of DAO [27] to generate challenge messages in the Ethereum blockchain and eliminate the centralized auditor. This new mechanism can provide better flexibility and is more secure against malicious miners. Moverover, we integrate the state-of-the-art public auditing schemes [20], [21] with our framework by removing the centralized TPA and utilizing a smart contract. We also evaluate the performance of the designed smart contract and the integrated schemes. Detailed evaluation results show that BDPA has acceptable performance in terms of efficiency and transmission costs. In addition, our BDPA can integrate with many other auditing schemes, and provide better scalability and availability. Specifically, the main contributions of this paper can be summarized as follows:

- We propose a decentralized public auditing framework for cloud storage, which gets rid of the centralized TPA and resists the tempting auditor who may generate biased results.
- We design a collaborative mechanism to help generate challenge messages by exploring the idea of DAO and a smart contract, which mitigates the impact of malicious blockchain miners.
- We experimentally validate that our BDPA has good performance and is scalable to the existing auditing schemes while removing the hidden dangers of the tempting and centralized auditor.

The rest of the paper is organized as follows. We firstly present the related works in Section 2. In Section 3, we review Xue et al.'s scheme and analyze the weakness of existing blockchain-based schemes. And in Section 4 we present the system model, threat model, design goals and some preliminaries. Then we illustrate detailed construction and security analysis in Section 5. The performance evaluation is shown in Section 6. Finally, we conclude the paper in Section 7.

2 Related Work

2.1 Traditional public auditing

To ensure the integrity of outsourced data, Juels et al. [28] firstly proposed the "proofs of retrievability" (POR) scheme, which relies on indistinguishable blocks hidden among file blocks that serve as sentinels to detect cloud data corruption. After that, Ateniese et al. [9] proposed the "provable data possession" (PDP) technique, which is a variant of POR that can support an unbounded number of challenge queries. Considering the large data volumes and the limitation of communication resources, cloud users usually adopt public auditing schemes to help audit the integrity of their outsourced data periodically. In 2013, Shacham et al. [29] proposed the compact POR scheme, which utilizes homomorphic authenticators to yield compact proofs and also support public verification. Following this work, many public auditing schemes have been proposed [6], [30]. However, these public auditing protocols are mainly based on public key infrastructure (PKI) systems which are inefficient and cumbersome.

To avoid managing users' certificates, identity-based public auditing schemes have been proposed [31], [32]. In these schemes, a publicly known string representing an individual or organization is used as a public key. A trusted third party, also called the private key generator (PKG), generates private keys for all cloud users [33]. However, identity-based schemes have the key escrow problem and are vulnerable to malicious PKG who controls all users' private key [34]. Therefore, certificateless public auditing schemes have been proposed [21], [23], where the key generation process is split between the PKG and the user.

2.2 Blockchain-based public auditing

Recently, investigations of the credibility of TPA have attracted researchers' attention. In most traditional schemes, the auditor is assumed to be honest and reliable. This is a strong assumption because corruption of auditors could happen in practice [20]. With the popularity of blockchain technology, many blockchain-based schemes have been proposed to resist malicious auditors [20]–[23]. In 2014, Armknecht et al. [22] proposed a blockchain-based auditing scheme which utilize random values in Bitcoin [24]. After that, Zhang et al. [23] also proposed a certificateless public auditing scheme in cyber-physical-social systems.

In 2019, Xue et al. [20] proposed an identity-based public auditing scheme which utilized Bitcoin as a random source and stored log hash values in transactions in the Bitcoin blockchain. Later, Zhang et al. [21] also proposed a blockchain-based certificateless auditing scheme against procrastinating auditors which utilize Ethereum platform and gave security analysis for malicious blockchain miners. Recently, Zhang et al. [35] proposed a conditional identity privacy-preserving public auditing mechanism for cloud-based WBANs and integrate Ethereum blockchain into this scheme. However, most of the existing blockchain-based public auditing schemes are based on a centralized TPA and cannot thwart *tempting auditors* who can incentivize miners to generate biased auditing results.

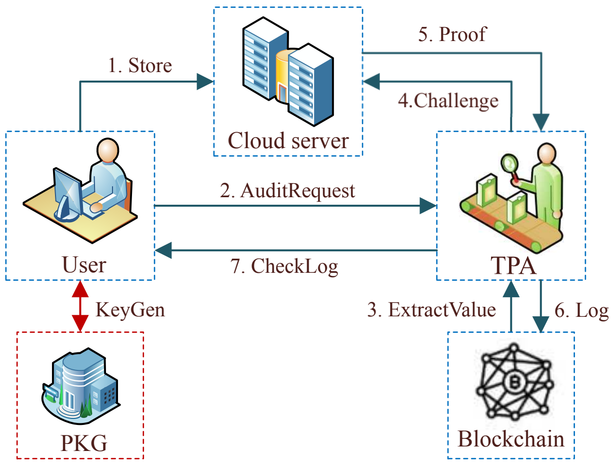


Fig. 1. System model of existing blockchain-based public auditing schemes

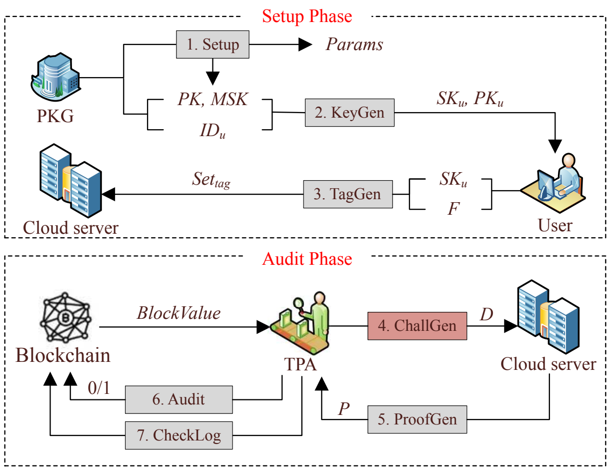


Fig. 2. Procedures of existing blockchain-based public auditing schemes

3 Problem Statement

3.1 Review of Xue et al.s scheme

Recently, Xue et al. [20] proposed an identity-based public auditing (IBPA) scheme against malicious auditors. Later, Zhang et al. [21] proposed a certificateless public verification scheme against procrastinating auditors (CPVPA). They both used the blockchain technology. The general system model of their schemes, as shown in Fig. 1, involves four entities: a PKG, a user, a cloud server (CS), and a TPA.

To clarify the limitations of above schemes, we illustrate technical details of IBPA and show general procedures of blockchain-based public auditing schemes. As demonstrated in Fig. 2, the framework of IBPA among various entities proceeds as follows:

Setup Phase. During this phase, the authorized PKG initializes the system by calling the Setup algorithm and assigns secret keys to each user by calling the KeyGen algorithm. The user can outsource data blocks and corresponding anthentication tags to the cloud server by calling the TagGen algorithm.

Setup$(1^\lambda) \rightarrow (PK, MSK, Params)$. The PKG generates two cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ of same prime order p with P as a generator of $\mathbb{G}_1$, and defines a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Then, it selects a random number $s \in \mathbb{Z}_p$ as the master secret key MSK and computes the public key $PK = sP$. In addition, it also defines a set of resistant hash functions including: $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $h : \mathbb{G}_1 \rightarrow \mathbb{Z}_p$ and $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$. It also generates a common state parameter w. The public parameter is set as:

$$Params = (\mathbb{G}_1, \mathbb{G}_2, e, p, P, H_1, H_2, H, h).$$

KeyGen$(PK, MSK, ID) \rightarrow (SK_u, PK_u)$. Given a user identity ID, it hashes ID to two elements $P_{u,0} = H_1(ID, 0), P_{u,1} = H_1(ID, 1)$ and computes corresponding secret keys as:

$$Q_{u,0} = sP_{u,0},$$
$$Q_{u,1} = sP_{u,1}.$$

The user's secret key is $SK_u = \{Q_{u,0}, Q_{u,1}\}$, and the public key is $PK_u = \{ID, P_{u,0}, P_{u,1}\}$.

TagGen$(Params, F, SK_u) \rightarrow Set_{tag}$. The user splits the data file F into n blocks, $F = m_1||m_2||\cdots||m_n$. For each block m_j, the user chooses a random $r \in \mathbb{Z}_p$ and a file name, and generates an authentication tag (S_j, T_j) using $Params$ and SK_u. The set of all authentication tags $\{S_j, T_j\}_{j\in[1,n]}$ is denoted by Set_{tag}.

$$(S_j, T_j) = (rH_2(w||j) + H(name||j)Q_{u,0} + m_jQ_{u,1}, rP),$$
$$Set_{tag} = \{(S_j, T_j)\}_{j\in[1,n]}.$$

Then the user sends $\{F, Set_{tag}\}$ to the cloud server and deletes the data file F from the local storage.

Audit Phase. In this phase, the TPA generates a challenge message by calling the ChallGen algorithm. The CS generates the proof information by calling the ProofGen algorithm. Then the TPA verifies the proof information by calling the Audit algorithm. The TPA also create log entries and store the hash value in the blockchain. And the user can check the validity of TPA's log file by calling the CheckLog algorithm.

ChallGen$(Params, t) \rightarrow C$. The TPA obtains the $Nonce$ in the corresponding block based on the time t and chooses a random l-element subset $J = \{a_1, a_2, \cdots, a_l\}$ from the set $[1, n]$. Then it chooses a random $v_j \in \mathbb{Z}_p$ for each $j \in J$ and generates a random challenge message $C = \{j, v_j\}$.

ProofGen$(Params, C) \rightarrow D$. After receiving the challenge message C from the TPA, the CS chooses a random number $x \in \mathbb{Z}_p$ and computes four values based on the data blocks, authentication tags and challenge messages.

$$\mu = x^{-1}(\sum_{j=a_1}^{a_l} m_jv_j + h(y)),$$
$$y = xP_{u,1},$$
$$(S, T) = (\sum_{j=a_1}^{a_l} v_jS_j, \sum_{j=a_1}^{a_l} v_jT_j).$$

Then the cloud server sends the proof information $D = \{S, T, \mu, y\}$ to the TPA.

Audit$(Params, D) \rightarrow 0/1$. The TPA audits the proof information D by calculating cryptographic elements and verifying a bilinear equation.

$$e_1 = e(\sum_{j=a_1}^{a_l} H_2(w||j)v_j, rP),$$

$$e_2 = e(\sum_{j=a_1}^{a_l} H(name||j)v_j P_{u,0} + (\mu y - h(y)P_{u,1}), sP),$$

$$e(S,P) \overset{?}{=} e_1 \cdot e_2.$$

This algorithm outputs an auditing result of either 0 or 1 for the TPA, where 0 means reject and 1 means accept. Then the TPA creates a log entry $(t, nonce, C, (S, T, \mu, y), 0/1)$ and store the hash values of log entries in the blockchain.

$\mathsf{CheckLog}(Params, F_{log}) \rightarrow 0/1$. The user can check the log information in the blockchain and verify the correctness of auditing results. The user chooses a random subset $B = \{b_1, b_2, \cdots b_{l'}\}$ from C and computes $S^B = \sum_{j=b_1}^{b_{l'}} v_j S_j$. Then the user verifies the following equation.

$$e_1^{'} = e(\sum_{j=b_1}^{b_{l'}} H_2(w||j)v_j, rP),$$

$$e_2^{'} = e(\sum_{j=b_1}^{b_{l'}} H(name||j)v_j P_{u,0} + (\mu y - h(y)P_{u,1}), sP),$$

$$e(S^B,P) \overset{?}{=} e_1^{'} \cdot e_2^{'}.$$

This algorithm outputs the checking result as 0 or 1 for the user, where 0 means reject and 1 means accept.

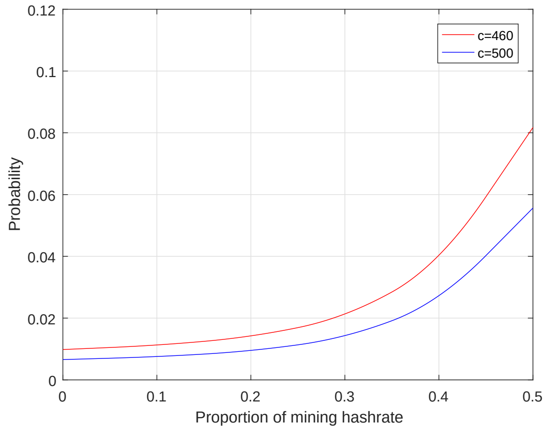
3.2 Weaknesses of existing blockchain-based public auditing schemes

In this section, we describe our motivation and analyse two main vulnerabilities in the latest blockchain-based public auditing schemes IBPA [20] and CPVPA [21]. IBPA and CPVPA both utilized the unpredictable values in the blockchain and designed log entries for the auditor, which explored the possibility of the combination of blockchain technology and public auditing. Nevertheless, they have limitations which can be summarized as follows:

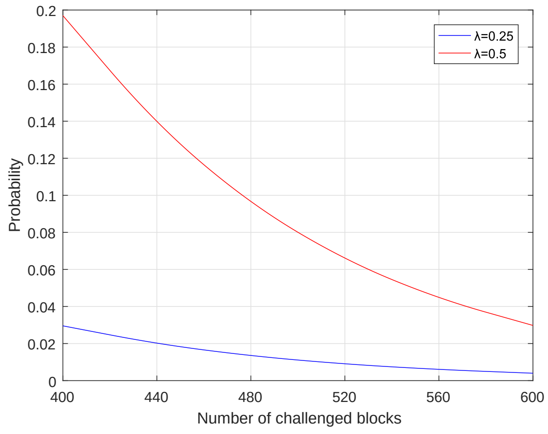
- Collusion. The two latest blockchain-based schemes cannot thwart *tempting auditors* which can collude with miners in the blockchain.
- Centralization. Most of existing public auditing schemes introduce a centralized TPA which may be vulnerable to a single point of failure.

To be more specific, we illustrate the details of their two main limitations (collusion and centralization) in the following discussion.

1) Collusion analysis. In existing schemes, the collusion between tempting auditors and blockchain miners are not fully considered. Specifically, the malicious blockchain miners can check block values before broadcasting newly mined blocks. If the result does not fulfill the requirements of tempting auditors, the miners can throw those blocks away. In addition, the malicious blockchain miners can build a chain longer than the honest one to change the selected data blocks. In such a sitiation, the randomness of challenge messages is compromised. Therefore, the tempting auditor can also collude with the CS to avoid choosing lost data blocks and hide corruption events from users, and this deviates from the primary objective of public verification to detect the data corruption. It is worth clarifying that the resistance against tempting auditors is vitally important for public auditing schemes in practice.



(a) Probability that the adversary wins w.r.t. mining hashrate



(b) Probability that the adversary wins w.r.t. number of challenged blocks

Fig. 3. Probability that the adversary wins

We consider that the only attack a tempting auditor can perform is to bias the values of decisive block Bl_d corresponds to the user's specified time by incentivizing malicious miners. Here, the adversary's requirement is that the indexes of challenged blocks exclude the corrupted ones. Note that the adversary can know in advance which future block will be decisive for the goal it tries to achieve. Now, we compute the probability that the adversary wins. The adversary model and security game model follow the ones proposed by Pierrot et al. [36].

We assume an adversary $\mathcal{A}$ who aims at biasing Bl_d to break the security of existing blockchain-based public auditing scheme. The probability that the adversary $\mathcal{A}$ wins are shown in Fig. 3. The indexes of corrupted data blocks form a set ε. $\mathcal{A}$ wins whenever none of indexes of challenged blocks fall in ε. Let $\chi : \varepsilon \mapsto \{0, 1\}$ be the characteristic function that predicates whether a given value meets $\mathcal{A}$'s requirement.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2021.3051622, IEEE Transactions on Cloud Computing

And we denote by P the probability for the values of Bl_d to satisfy $\chi(Bl_d) = 1$. We stress that P *is essentially a probability that the corrupted data set can pass the auditor's verification*. As evaluated by [9], [17], [30], when ρ fraction of data is corrupted and c is the number of randomly (uniformly) sampling blocks, we clearly have

$$P = (1 - \rho)^c. \tag{1}$$

We denote by $\mathcal{P}_\mathcal{A}$ the probability that $\mathcal{A}$ wins. And λ denotes $\mathcal{A}$'s relative power with respect to the total power of all miners. We denote that the initial block is indexed by 0 when the malicious miners start mining. The k-th block is the decisive block, whose extract values determine whether or not the adversary wins. $\mathcal{A}$ could try to modify an unpleasant decisive block until the blockchain has been lengthened by Δ blocks. In our analysis, we assume $\lambda < 51\%$. According to [36], we know that:

$$\mathcal{P}_\mathcal{A} = 1 - P_r(e_1) \cdot (P_r(e_2) + P_r(e_3)), \tag{2}$$

where e_1, e_2, e_3 are events defined as:

− e_1: the first broadcast decisive block Bl_d is such that $\chi(Bl_d) = 0$.

− e_2: given that event e_1 occurred, $\mathcal{A}$ fails to find any decisive block $Bl_d^{'}$ such that $\chi(Bl_d^{'}) = 1$.

− e_3: given that event e_1 occurred, $\mathcal{A}$ manages to find a decisive block $Bl_d^{'}$ such that $\chi(Bl_d^{'}) = 1$ but not to compute a branch sufficiently long to replace the main one.

For simplicity, we omit the complex technical equations for calculating the probabilities of event e_1, e_2, e_3. The additional technical details can be found in [36].

We show the probability that the adversary $\mathcal{A}$ wins in Fig. 3, where $\rho = 1\%$. When $\lambda = 1/5, c = 460$, the probability that $\mathcal{A}$ wins is 1.11%. When $\lambda = 1/2, c = 460$, the probability that $\mathcal{A}$ wins is 8.17%, which increases quickly. Note that both probability values are larger than the analytical results in [21]. For $\lambda = 1/4, c = 400$ and a fixed value of $k = 18$ that roughly corresponds to an adversary $\mathcal{A}$ that starts mining with three hours in advance, $\mathcal{P}_\mathcal{A}$ tends to 2.95% which is more than twice the probability value of 1.305% discussed in [21]. This probability that is still considerable in reality because there are enormous cloud users with data auditing requirements.

2) Centralization analysis. In most existing public auditing schemes, there exist a centralized TPA. Once the TPA is compromised due to hardware or software failures, some incorrect auditing results may be computed and recorded in the blockchain, which may give users incorrect auditing results. In other words, the availability of auditing services can not be guaranteed once the TPA fails to work normally. Moreover, it might be too late for users to recover the data loss or damage because they usually check the auditing results over a long period of time.

According to the previous discussion and analysis, we summarize the drawbacks of IBPA [20] and CPVPA [21] as follows. To eliminate the above two limitations, we also propose BDPA which replaces the centralized TPA with a decentralized blockchain network in Section 4.

- **IBPA drawback.** In IBPA [20], the challenge message is generated based on the *Nonce* value of Bitcoin. It has become widespread to utilize the inherent unpredictability of blockchains as a source of public randomness. However, this idea lacks a security model. In IBPA, they do not give any rigorous security analysis when miners are malicious and collude with tempting auditors.
- **CPVPA drawback.** In CPVPA [21],many *BlockHash* values in the Ethereum decide challenge messages. The authors analyzed the situation where malicious miners can sample biased challenge messages. Nevertheless, they only considered the situation where malicious miners can affect the latest block. In other words, the miners can bias challenge messages only by biasing the next broadcast block. In reality, an adversary can be informed by the tempting auditor in advance which future block will be decisive for the goal it tries to achieve.

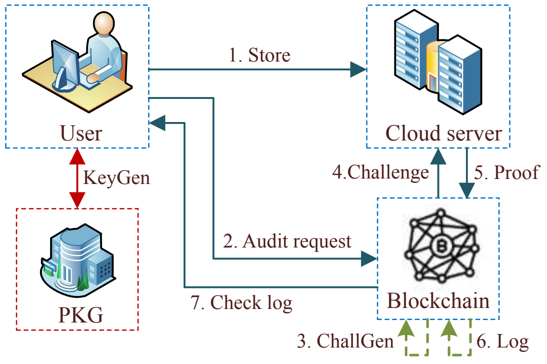


Fig. 4. System model

4 Definitions and Preliminaries

4.1 System model

The system model of BDPA, as shown in Fig. 4, involves four entities: a PKG, a user, a CS, and a blockchain network.

- PKG: The PKG is governed by a trusted authority, which initializes the system parameters and generates private key for the user based its identity.
- User: The user is the data owner with limited storage and computation resources which uploads local data to the CS.
- CS: The CS is managed by a cloud service provider, and provides users with cloud storage services. It usually has not only a large amount of storage space, but also powerful computational capabilities.
- Blockchain: The blockchain is a transparent, immutable and distributed ledger maintained by some permissioned nodes[1]. After the user's auditing requests have been uploaded to the blockchain, these nodes can cooperate to generate the challenge message and verify the proof information. The hash value of log files is logged in the blockchain.

BDPA is to utilize the decentralized blockchain to verify the integrity of outsourced data on the cloud server. Here,

1. The permissioned nodes include some auditors and servers with enough computation power. They still might be dishonest. Note that permissioned nodes in the blockchain network should be authorized for joining but will be excluded once behave maliciously.

we briefly describe the relationships among the entities in the system model. The user splits the file into many blocks and generates a tag for each block. Then the user submit its auditing task to the smart contract. The blockchain generates a random value based on the idea of DAO and chooses a random node as the verifier. The chosen blockchain node chooses a subset of all data blocks as a challenge message. The CS generates the proof based on the challenge message. The chosen node in the blockchain verifies the validity of the proof. The user can also check the behaviour of the blockchain node.

Compared with the system model of existing public auditing schemes [20], [21], we replace a trusted and centralized TPA with the decentralized blockchain network to generate random tuples and compute bilinear pairings. The proposed BDPA is formally defined as follows.

Definition 1. *The BDPA consists of seven algorithms:* ***Setup****,* ***KeyGen****,* ***Store****,* ***ChallGen****,* ***ProofGen****,* ***Audit****, and* ***CheckLog****.*

- **Setup**. This algorithm establishes necessary parameters for the system.
- **KeyGen**. This algorithm generates a private key for each authorized user.
- **Store**. This algorithm splits the user's file into blocks and generates a set of authentication tags which correspond to those data blocks generated by the user.
- **ChallGen**. This algorithm enables the nodes in the blockchain network to generate random challenge messages for checking data integrity.
- **ProofGen**. This algorithm enables the cloud server to generate proof information which proves the data integrity of the outsourced file.
- **Audit**. This algorithm enables the chosen node with enough computation power in blockchain network to verify the proof information locally.
- **CheckLog**. This algorithm checks the validity and correctness of the log file in blockchain for the user.

4.2 Threat model

In the system, the PKG and the user is fully trusted and will not launch any attack. This is the only centralized entity in BDPA. Moreover, we assume that no external adversary can truncate or tamper the communications among various entities. Since we eliminate the centralized TPA in our construction, we only consider threats from the malicious cloud server and the untrusted blockchain.

Malicious cloud server. The cloud server may hide data loss to maintain a good reputation by forging valid proof information. Moreover, it may delete data that the user has never accessed to save storage space.

Malicious blockchain nodes. We follow the existing threat model of blockchain [36] with the integration of the adversary definition in [21]. The nodes in the blockchain may become two types of adversaries:

1) Type I adversary $\mathcal{A}_{\text{I}}$: It is a participant node and can submit specially constructed values when generating challenge messages. It cannot modify other participants' commitments and does not have enough computation power to manipulate newly generated blocks.

2) Type II adversary $\mathcal{A}_{\text{II}}$: It is the malicious miner responsible for generating new blocks in the blockchain and does not participate in the procedures of generating challenges. It can throw newly mined blocks or create a fork[2] to modify an unpleasant block to generate biased blocks, but it cannot manipulate over 51% computing power in practice.

4.3 Design goals

In this paper, we target the secure public integrity auditing scheme for cloud storage systems against tempting and centralized TPA. We need to construct an unbiased and unpredictable challenge message based on the idea of DAO. Existing blockchain-based public auditing schemes cannot resist the tempting auditor who can collude with the CS and incentivize the malicious miners to generate biased challenge messages.

In our BDPA, there are no centralized auditor but we introduce a untrusted blockchain network. Therefore, the unbiased construction of challenge information is crucial and essential in our scheme. To ensure efficient data integrity auditing for outsourced data under the aforementioned system model and threat model, the proposed BDPA aims to simultaneously achieve the following security and utility goals:

- Security. The CS must store the user's data correctly when it passes verification. The adversaries in the untrusted blockchain cannot deceive the user by forging auditing records.
- Efficiency. The public auditing is light weight, in other words, the integrated decentralized scheme should be performed with acceptable communication and computation overhead.
- Availability. The public auditing service is robust and available for users. Even if some nodes in the blockchain are compromised, our BDPA can still work normally.
- Scalability. The proposed scheme is scalable and can integrate with existing auditing schemes easily.

4.4 Bilinear maps

Let $\mathbb{G}_1$ be an additive cyclic group and $\mathbb{G}_2$ be a multiplicative cyclic group with the same prime order p. Let P be a generator of $\mathbb{G}_1$. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 arrow \mathbb{G}_2$ has the following three properties:

- Bilinearity: $\forall x, y \in \mathbb{Z}_p^*$ and $Q, R \in \mathbb{G}_1$, we have $e(xQ, yR) = e(Q, R)^{xy}$.
- Non-degeneracy: $e(P, P) \neq 1$.
- Computability: there exists an efficient algorithm to compute $e(Q, R)$ for any input Q and R.

Discrete Logarithm (DL) Problem: Given $\mathbb{G}_1$ with a generator P, for any $Q = xP \in \mathbb{G}_1$; compute $x \in \mathbb{Z}_p$.

Computation Diffe-Hellman (CDH) Problem: Given $\mathbb{G}_1$ with a generator P, for , and given xP and yP with unknown $x, y \in \mathbb{Z}_p^*$; compute xyP.

2. Note that the value of any specific block can be changed via forking: in blockchain protocols, consensus does not go to the version of a block that arrived first, but to the version of the block that belongs to the longest chain.

4.5 Blockchain

A blockchain [24] is a one-way linear set of data units, in which each data unit is called a *block*. Each block contains a pointer *PreBlockHash*, a timestamp *Time*, and multiple transactions *TXs*. The *PreBlockHash* field points to the previous block and chains the ledger chronologically. The *Time* field records the time when the block was added to the blockchain. A transaction generally represents the events of transferring tokens from one participant to another. The participants who verify transactions are called *miners*. All of data blocks are linked in chronological order using a cryptographic hash function which ensures immutability of data [37]–[40]. In this way, the blockchain is inherently verifiable, and those recorded transactions are resistant to tampering.

Generally, a blockchain network is maintained by some nodes according to a common consensus mechanism. The blockchain technique can be generally classified into three types: public blockchain, permissioned blockchain and private blockchain. In a public blockchain, users or nodes can conduct transactions without referring to a centralized authority (i.e. anyone can join or quit from the generating of new block at any time). Popular public blockchains include Bitcoin and Ethereum which are PoW-based blockchain types. In reality, they are quite energy intensive, and suffer from heavy power consumption and 51% attack. In contrast, permissioned and private blockchains are maintained by some trusted and permissioned nodes.

Famous permissioned blockchain such as Hyperledger[3] provides suitable verification of participants' identity and are suitable for enterprise-grade deployments. Actually, permissioned blockchain is more appropriate for our proposal, because permissioned nodes are more trustworthy and can be utilized to initialize random tuples as challenge messages. Nevertheless, the permissioned blockchain need cooperation between many cloud service providers, which is currently not considered in the proposed BDPA. Besides, the blockchain which utilizes proof-of-stake (PoS) consensus mechanism such as Peercoin can also be used in our construction. In a PoS-based blockchain, validators are responsible for ordering transactions and creating new blocks so that all nodes can agree on the state of the network. A user's stake is used as a way to incentivise good validator behaviour. In a real-world situation, PoS provides better security than PoW against 51% attack and seems to be the future of blockchain technology.

In our implementation, we choose Ethereum[4], a popular platform with outstanding performance, as our experiment environment to illustrate the feasibility of BDPA, while omiting users in the blockchain network because of their limited storage and computation capabilities. The Ethereum platform can provide the function of smart contract and is suitable for BDPA's implementation. Moreover, the Ethereum blockchain is moving to PoS from PoW and can provide stronger immunity to centralization in the future. Note that our BDPA can also be constructed on a permissioned blockchain.

3. https://www.hyperledger.org/
4. https://ethereum.org/en/

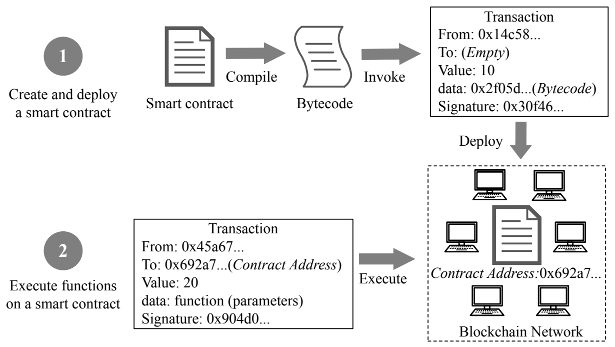


Fig. 5. Smart contract overview

4.6 Smart contract

The concept of smart contracts was firstly introduced by Nick Szabo [41]. He came to a conclusion that any decentralized ledger can be used as self-executable contracts. These digital contracts could be converted into codes and allowed to be run on a blockchain.

Smart contract [42] is a computerized transaction protocol that executes the terms of a contract and is compiled by a Turing complete language (e.g, Solidity) into a piece of bytecode. As demonstrated in Fig. 5, after compilation, a smart contract will be chained into the blockchain as a transaction. The provided functions or application binary interfaces (ABIs) in smart contracts can be triggered by publishing a transaction or sending a message from other contracts. Note that a smart contract is a special account, which also owns its address. One can transfer tokens to a smart contract or invoke functions of a smart contract. Since the the function calls and other operations are recorded in the blockchain, a smart contract is often treated as a traceable and unmodifiable execuation environment.

4.7 DAO

A DAO is a decentralized autonomous organization that is run through rules encoded as computer programs called smart contracts. A DAO's financial transaction records and programmed rules are maintained on a blockchain.

Existing schemes [20]–[22] which utilize the blockchain data as a random seed are not trustworthy because miners with prior knowledge have the ability to manipulate blockchain data, and thus can indirectly affect the random number generator (RNG). If RNG is based on blockchain data, it will give blockchain miners capacity to construct random numbers in their favor. In order to solve the problem of generating random values in Ethereum, RANDAO[5] has been proposed as an infrastructure for Ethereum. To be more specific, it is a DAO that anyone can participate in, and the random number is jointly generated by all participants together.

In our BDPA, we explore the idea of RANDAO by designing the basic processes which generate random tuples and handle the verification process in the blockchain. The participants should be permissioned and will be excluded once behave maliciously.

5. https://github.com/randao/randao

5 BDPA

In this section, we firstly describe our detailed construction of BDPA which integrates with Xue et al.'s scheme, and we also show how our smart contract works. Then we present the security analysis for the auditing scheme and resistance against untrusted blockchain. Finally, we show the remark and give further discussion.

5.1 Construction of BDPA

In this section, we describe the detailed construction of BDPA, which can integrate with the existing public auditing schemes. Specifically, IBPA-BDPA denotes the scheme which integrates the auditing procedures of IBPA and BDPA framework. And CPVPA-BDPA denotes the scheme which integrates the auditing procedures of CPVPA and BDPA framework. Now we show the core idea of BDPA by constructing the IBPA-BDPA scheme.

As illustrated in Fig. 4, we use a decentralized blockchain to replace a centralized TPA. Generally, we split the procedures into two phases: *Setup phase* and *Audit phase*. The procedures of BDPA among various entities are similar to the existing schemes except for the ChallGen, Audit and CheckLog algorithm.

Setup Phase. This phase is essentially the same as existing public auditing schemes. During this phase, the PKG initializes the system by calling the Setup algorithm and assigns secret keys to each user by calling the KeyGen algorithm. Then the user can outsource data blocks and tags to the CS by calling the TagGen algorithm.

Setup$(1^\lambda) \rightarrow (PK, MSK, Params)$. The PKG generates two cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ of same prime order p with P as a generator of $\mathbb{G}_1$, and defines a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Then, it selects $s \in \mathbb{Z}_p$ as the master secret key MSK and computes the public key PK is $Q = sP$. It also defines a set of hash functions including: $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $h : \mathbb{G}_1 \rightarrow \mathbb{Z}_p$ and $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$. It also generates a common state parameter w. The public parameter is set as:

$$Params = (\mathbb{G}_1, \mathbb{G}_2, e, p, P, H_1, H_2, H, h).$$

KeyGen$(PK, MSK, ID) \rightarrow (SK_u, PK_u)$. Given a user identity ID, it hashes ID to two elements $P_{u,0} = H_1(ID, 0), P_{u,1} = H_1(ID, 1)$ and computes $Q_{u,0} = sP_{u,0}$, $Q_{u,1} = sP_{u,1}$. The user's secret key is $SK_u = \{Q_{u,0}, Q_{u,1}\}$, and the public key is $PK_u = \{ID, P_{u,0}, P_{u,1}\}$.

TagGen$(Params, F, SK_u) \rightarrow Set_{tag}$. The user splits the data file F into n blocks, $F = m_1||m_2||\cdots||m_n$. For each block m_j, the user chooses a random $r \in \mathbb{Z}_p$ and computes $R = rP$. The user also selects a file name, and generates an authentication tag (S_j, T_j) using $Params$ and SK_u:

$$(S_j, T_j) = (rH_2(w||j) + H(name||j)Q_{u,0} + m_jQ_{u,1}, R).$$

The set of all authentication tags $\{S_j, T_j\}_{j\in[1,n]}$ is $Set_{tag} = \{(S_j, T_j)\}_{j\in[1,n]}$. Then the user sends $\{F, Set_{tag}\}$ to the CS and deletes the data file F from the local storage.

Audit Phase. During this phase, the user submits an auditing request to the blockchain. The blockchain generates a challenge message by calling the ChallGen algorithm, which is implemented in a smart contract. After receiving the challenge message from the blockchain, the CS generates the proof information by calling the ProofGen algorithm.

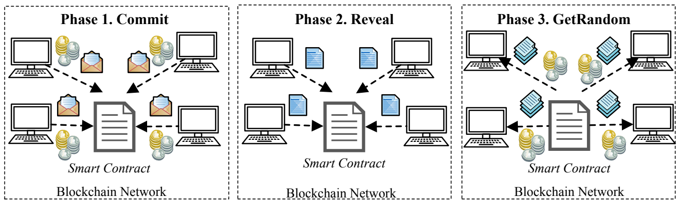


Fig. 6. The phases of a campaign

The blockchain chooses a random node with enough computing ability to verify the proof information by calling the Audit algorithm. The user can verify the blockchain node's behavior by calling the CheckLog algorithm.

ChallGen$(Params, endTime) \rightarrow C$. After the user submits an auditing request to the blockchain, the smart contract will launch some campaigns which ends at predefined time $endTime$ to generate random challenge messages $C = \{j, v_j\}$ among some permissioned participants.

As described in Fig. 6, each campaign generally has three phases: *Commit*, *Reveal* and *GetRandom*. The time for each phase can be specified by the user. The campaign is implemented in the smart contract in the blockchain and can be viewed as a source of secure and transparent execution procedure.

- Commit Phase. Every participant submits the hash of its secret value and a certain amount of deposit as a guarantee of honesty. These participants generally have enough computing ability and can finish bilinear computations.
- Reveal Phase. Every participant reveals its secret value and the smart contract will check whether the hash value of revealed secret is the same as the submitted hash value during the Commit phase. If a malicious participant fails to reveal its secret or reveal different secret value in this phase, its deposit will not refunded and it will be added to the blacklist for generating challenge messages.
- GetRandom Phase. The smart contract will check the secret numbers successfully collected and calculate the random number from all participants' secret values and return the deposits and bonus to honest participants.

As shown in Algorithm 1, we realize the ChallGen algorithm and other functions in blockchain using a single smart contract. After the user send an auditing request to the blockchain, the smart contract will call newTask to create a task with id $taskID$, and launch some campaigns which can be added to the Task. Then each campaign generates a random value. Finally, the task will obtain many random values from all campaigns. Note that in our current construction, we create many campaigns for an auditing task. This method can also be replaced by creating just one campaign in the blockchain network. Then the smart contract can choose a random node to verify the proof information from all participants in this campaign. The chosen permissioned node can generate challenge messages based on the random value from this campaign and verify the proof information generated by the cloud server.

Algorithm 1 Smart Contract of ChallGen

```
Require: Function name, invoked parameters
Ensure: Setting up functions.
structure $Campaign$
% Define the structure of a campaign which generates a random number.
  $bnum; commitBalkline; commitDeadline;$
% the block numbers that specify the period
  $deposit; bountypot;$
% the deposit and bonus
  $random; settled; commitNum; revealsNum;$
% final result and status
  $consumers; participants; commitments;$
structure $Task$
% Define the structure of an auditing request of a user.
  $taskId; fileName; n; numChallenges;$
% the total number of data blocks and challenged blocks
  $auditor; campaigns; challenges; proof;$
  $deposit; bountypot; result;$
function newTask ($name, n, c, deposit$)
% Invoked by a user who has an auditing request.
  $require(msg.sender \in PL);$
%$PL$ denotes permissioned nodes' addresses.
  $taskID = tasks.length++;$
  $task = tasks[taskID];$
  $numTasks++;$
  $task.taskId = taskID;$
  $task.fileName = name;$
  $task.n = n;$
  $task.numChallenges = c;$
  $task.deposit = deposit;$
  $task.bountypot = msg.value;$
function genChallenge ($taskID$)
  $task = tasks[taskID];$
  for $i = 0 \rightarrow task.campaigns.length$ do
      $campaignID = task.campaigns[i];$
      $j = campaigns[campaignID].random\%task.n;$
      $v_j = campaigns[campaignID].random;$
      $task.challenges.push(Challenge(j, v_j));$
  end for
function getRandom ($campaignID$)
  $Campaignstoragec = campaigns[campaignID];$
  if $c.commitNum == c.revealsNum$ then
      $c.settled = true;$
      return $c.random;$
  end if
```

ProofGen$(Params, C) \rightarrow D$. After receiving the challenge message C from the blockchain, the CS chooses a random number $x \in \mathbb{Z}_p$ and computes four values based on the data blocks, authentication tags and challenge messages as defined in IBPA [20].

$$\mu = x^{-1}(\sum_{j=a_1}^{a_l} m_j v_j + h(y)),$$
$$y = xP_{u,1},$$
$$(S,T) = (\sum_{j=a_1}^{a_l} v_j S_j, \sum_{j=a_1}^{a_l} v_j T_j).$$

Then the cloud server sends the proof information $D = \{S, T, \mu, y\}$ to the chosen node in the blockchain network.

Audit$(Params, D) \rightarrow 0/1$. The chosen node with enough computation power audits the proof information D by verifying a bilinear equation.

$$e_1 = e(\sum_{j=a_1}^{a_l} H_2(w||j)v_j, R),$$
$$e_2 = e(\sum_{j=a_1}^{a_l} H(name||j)v_j P_{u,0} + (\mu y - h(y)P_{u,1}), Q),$$
$$e(S,P) = e_1 \cdot e_2. \tag{3}$$

This algorithm outputs an auditing result of either 0 or 1, where 0 means reject and 1 means accept. Then the verifier creates a log entry $(taskID, endTime, C, (S, T, \mu, y), 0/1)$ and stores the hash values of log entries in the blockchain.

CheckLog$(Params, F_{log}) \rightarrow 0/1$. The user can check the log information in the chosen blockchain verifier and verify the correctness of auditing results. The user firstly checks whether the challenge message C is the the same as the values generated in the smart contract. Then the user chooses a random subset $B = \{b_1, b_2, \cdots b_{l'}\}$ from C and computes $S^B = \sum_{j=b_1}^{b_{l'}} v_j S_j$. Then the user verifies the following equation.

$$e_1^{'} = e(\sum_{j=b_1}^{b_{l'}} H_2(w||j)v_j, R),$$
$$e_2^{'} = e(\sum_{j=b_1}^{b_{l'}} H(name||j)v_j P_{u,0} + (\mu y - h(y)P_{u,1}), Q),$$
$$e(S^B, P) = e_1^{'} \cdot e_2^{'}. \tag{4}$$

This algorithm outputs the checking result as 0 or 1 for the user, where 0 means reject and 1 means accept.

Correctness. The correctness of the auditing scheme in BDPA depends on the correctness of equation (3) and (4). The correctness proof is shown as follows:

$$\begin{aligned}
e(S,P) &= e(\sum_{j=a_1}^{a_l} v_j S_j, P) \\
&= e(\sum_{j=a_1}^{a_l} v_j (rH_2(w||j) + H(name||j)Q_{u,0} + m_j Q_{u,1}), P) \\
&= e_1 \cdot e(\sum_{j=a_1}^{a_l} H(name||j)v_j P_{u,0} + \sum_{j=a_1}^{a_l} m_j v_j P_{u,1}, Q) \\
&= e_1 \cdot e(\sum_{j=a_1}^{a_l} H(name||j)v_j P_{u,0} + (\mu y - h(y)P_{u,1}), Q) \\
&= e_1 \cdot e_2.
\end{aligned}$$

Note that the concrete algorithms of BDPA including Setup, KeyGen, TagGen, ProofGen can be adjusted according to specific auditing schemes. The ChallGen algorithm is designed based on the idea of DAO. The algorithm Audit is called by the random chosen permissioned node in the blockchain. And the CheckLog algorithm is called by the user to audit the blockchain node's behavior. If the result is incorrect, the chosen node will be added to the blacklist and be rejected in the following auditing procedure.

5.2 Security analysis

In this section, we analyze the security of BDPA from the following two aspects: 1) the auditing scheme's storage correctness guarantee and 2) the resistance to type I adversary $\mathcal{A}_{\text{I}}$ and type II adversary $\mathcal{A}_{\text{II}}$ defined in the blockchain.

Theorem 1. *BDPA achieves the storage correctness guarantee, that is, if the cloud server passes the auditing procedure, it must guarantees the cloud data authenticity.*

Proof. In reality, the storage correctness relies on the integrated public auditing scheme. To prove the cloud data integrity, we need to prove that the auditing scheme integrated in our BDPA achieves the data authenticity defined in [29]. Next, we firstly prove that a part of the proof information μ cannot be forged in **Game 1**, and we prove the part of signature $\{S, T, \mu\}$ cannot be forged in **Game 2**. Now we assume that the malicious cloud server could forge data blocks and signatures, and also pass the auditing verification. The details of these two games are illustrated as follows.

Game 1. The malicious CS could forge a valid proof as $\{S, T, \mu^*, y\}$, where $\mu^* = x^{-1}\cdot(\sum_{j=a_1}^{a_l} m_j^* v_j + h(y))$. Clearly, there exists at least a data block such that $\Delta m_j = m_j^* - m_j \neq 0$. According to the auditing procedure, the following two equation holds:

$$e(S,P) = e_1 \cdot e(\sum_{j=a_1}^{a_l} h_j v_j P_{u,0} + (\mu y - h(y)P_{u,1}), Q),$$
$$e(S,P) = e_1 \cdot e(\sum_{j=a_1}^{a_l} h_j v_j P_{u,0} + (\mu^* y - h(y)P_{u,1}), Q),$$

where $e_1 = e(\sum_{j=a_1}^{a_l} H_2(w||j)v_j, rP)$ and we denote $h_j = H(name||j)$. Here, we will show how the malicious CS can solve the DL problem in **Game 1** by constructing a simulator.

In the Setup algorithm, the simulator sets $Q = sP \in \mathbb{G}_1$ as the public key and the instance of the DL problem. The simulator operates the random oracles H_1, H_2, h and H, and stores lists of queries. Then, the simulator randomly chooses $\alpha, \beta, b, \eta, d \in \mathbb{Z}_p$, and sets the hash values as $h_j = H(name||j) = \eta \in \mathbb{Z}_p$, $H_2(w||j) = bP \in \mathbb{G}_1$, $P_{u,0} = H_1(ID,0) = \alpha P \in \mathbb{G}_1$ and $P_{u,1} = H_1(ID,1) = \beta P \in \mathbb{G}_1$. According to the verification equation (3) for the correct proof information, we could have

$$\begin{aligned} e(S,P) &= e_1 \cdot e(\sum_{j=a_1}^{a_l} h_j v_j P_{u,0} + (\mu y - h(y)P_{u,1}), Q) \\ &= e_1 \cdot e(\sum_{j=a_1}^{a_l} h_j v_j P_{u,0} + \sum_{j=a_1}^{a_l} m_j v_j P_{u,1}, sP) \\ &= e(r\sum_{j=a_1}^{a_l} H_2(w||j)v_j + s\sum_{j=a_1}^{a_l}(h_j v_j P_{u,0} + m_j v_j P_{u,1}), P). \end{aligned}$$

The simulator continues to interact with the malicious CS. Finally, we could obtain that above equation also holds for data block m_j^* and the forged proof information $\{S, T, \mu^*, y\}$. Therefore, we could compute $e(S-S, P) = e(s\sum_{j=a_1}^{a_l}(m_j^* - m_j)v_j P_{u,1}, P) = e(\sum_{j=a_1}^{a_l} s\Delta m_j v_j \beta P, P)$. Since $Q = sP$, we have $\sum_{j=a_1}^{a_l} \beta\Delta m_j v_j Q = 0$. As we know, $\mathbb{G}_1$ is an additive cyclic group. $\forall A, B \in \mathbb{G}_1$, there exists $\xi \in \mathbb{Z}_p$, such that $B = \xi A$. Thus, each $v_j Q$ could also be represented as $v_j Q = v_j sP = xA + yB$, where $x, y \in \mathbb{Z}_p$. Obviously, the following equation holds: $\sum_{j=a_1}^{a_l} \beta\Delta m_j(xA + yB) = 0$. Thus, we could solve the DL problem by calculating:

$$B = -(\textstyle\sum_{j=a_1}^{a_l} \Delta m_j x / \sum_{j=a_1}^{a_l} \Delta m_j y) \cdot A.$$

By our setting, we have $\Delta m_j = m_j^* - m_j \neq 0$, and the random number $y \in \mathbb{Z}_p$ is unkown for the malicious CS. The denominator is 0 with a probability of $1/p$. Therefore, we could construct a simulator which can be utilized by the the malicious cloud server to solve the DL problem with a probability of $1 - 1/p$. This probability is non-negligible and can lead to a contradiction.

Game 2. The malicious CS could forge a valid aggregate signature as $\{S^*, T^*, \mu^*, y\}$, which is different from the expected proof information $\{S, T, \mu, y\}$. Clearly, we could have $\{S^*, T^*\} \neq \{S, T\}$. In the auditing process, the expected proof information satisfies the bilinear pairing equation as follows:

$$\begin{aligned} e(S,P) &= e_1 \cdot e(\sum_{j=a_1}^{a_l} h_j v_j P_{u,0} + (\mu y - h(y)P_{u,1}), Q) \\ &= e(\sum_{j=a_1}^{a_l} rH_2(w||j)v_j + s\sum_{j=a_1}^{a_l}(h_j v_j P_{u,0} + m_j v_j P_{u,1}), P). \end{aligned}$$

Here, we will show how the malicious CS can construct a simulator to solve the CDH problem in **Game 2**, that is, the simulator aims to compute sP' from P, sP and P'.

In the Setup algorithm, the simulator sets $Q = sP \in \mathbb{G}_1$ as the PKG's public key, without awareness of the secret value s. Then, the simulator can operate the random oracles H_1, H_2, h and H, and save lists of queries to reply coincidentally. Next, the simulator randomly chooses $\alpha, \beta, b, \eta, d \in \mathbb{Z}_p$, and sets the hash values as $H(name||j) = \eta \in \mathbb{Z}_p$, $H_2(w||j) = bP \in \mathbb{G}_1$, $P_{u,0} = H_1(ID,0) = \alpha P' \in \mathbb{G}_1$ and $P_{u,1} = H_1(ID,1) = \beta P' \in \mathbb{G}_1$. According to the ProofGen algorithm, we could have $T = \sum_{j=a_1}^{a_l} v_j T_j = \sum_{j=a_1}^{a_l} rPv_j$. Therefore, we have

$$\begin{aligned} e(S,P) &= e(\sum_{j=a_1}^{a_l} rbPv_j + s\sum_{j=a_1}^{a_l}(\eta v_j \alpha P' + m_j v_j \beta P'), P) \\ &= e(bT + s\sum_{j=a_1}^{a_l}(\eta v_j \alpha P' + m_j v_j \beta P'), P). \end{aligned}$$

The simulator continues to interact with the malicious cloud server. Finally, the malicious CS successfully forges the valid proof information $\{S^*, T^*, \mu^*, y\}$ which is different from the expected $\{S, T, \mu, y\}$. The above equation also holds for the forged proof information, we further have

$$e(S^*, P) = e(bT^* + s\sum_{j=a_1}^{a_l}(\eta v_j \alpha P' + m_j^* v_j \beta P'), P).$$

Similarly, we could further calculate $e(S^* - S, P)$ by using the previous two equations. Thus, we could compute:

$$\begin{aligned} e(S^* - S, P) &= e(b(T^* - T) + s \sum_{j=a_1}^{a_l} (m_j^* - m_j) v_j \beta P', P) \\ &= e(b(T^* - T) + \sum_{j=a_1}^{a_l} s \Delta m_j v_j \beta P', P). \end{aligned}$$

So we get $S^* - S = b(T^* - T) + \sum_{j=a_1}^{a_l} s \Delta m_j v_j \beta P'$. And we could reschedule this equation to solve the hardness assumption of CDH problem. In particular, we could get:

$$\begin{aligned} S^* - S + b(T - T^*) &= sP' \beta \textstyle\sum_{j=a_1}^{a_l} \Delta m_j v_j \\ sP' &= (S^* - S + b(T - T^*)) \cdot (\beta \textstyle\sum_{j=a_1}^{a_l} \Delta m_j v_j)^{-1}. \end{aligned}$$

By our setting, we have $\Delta m_j \neq 0$, $S^* - S \neq 0$ and $T^* - T \neq 0$. Each random number $\beta, v_j \in \mathbb{Z}_p$ is unknown for the malicious CS. Therefore, the denominator is 0 with a probability of $1/p^2$. Thus, we could construct a simulator which can be utilized by the the malicious cloud server to solve the CDH problem with a probability of $1 - 1/p^2$. This probability is non-negligible and can lead to a contradiction. This concludes the proof that BDPA achieves the storage correctness guarantee and can ensure the cloud data integrity. □

Theorem 2. *BDPA is able to thwart type I adversary $\mathcal{A}_{\text{I}}$ and type II adversary $\mathcal{A}_{\text{II}}$ defined in the blockchain network.*

Proof. As described in Section 4.2, adversary $\mathcal{A}_{\text{I}}$ is the participant node which can submit specially constructed values when generating challenge messages. Note that $\mathcal{A}_{\text{I}}$ must submit and reveal its secret in corresponding phases otherwise its deposit will not be refunded and it will be excluded from other campaigns.

In our smart contract, the random tuples are generated by all participants together. The idea is based on DAO. Each user's secret value can be used as the last commitment and cannot be predicted by nodes in the blockchain. Therefore, even if $\mathcal{A}_{\text{I}}$ submits a carefully constructed secret value, the resulting random value is still not predictable as long as there exists at least one honest participant.

As described in Section 4.2, adversary $\mathcal{A}_{\text{II}}$ is the malicious blockchain miner which is responsible for generating new blocks and do not participate in the procedures of generating challenge messages. It can throw newly mined blocks or create a fork to modify an unpleasant block to generate biased blocks, but it cannot manipulate over 51% computing power in practice.

Note that in our construction, we design a mechanism based on the idea of DAO and do not use any block values such as *Nonce* and *BlockHash*. Therefore, even if $\mathcal{A}_{\text{II}}$ can influence the generated block values, it does not have any impact on the generation of challenge messages in the smart contract. The security of auditing procedures relies on the random challenge messages. Note that the behaviors of all participants are traceable and unpredictable. Therefore, in our framework, we can guarantee the randomness of challenge message and ensure the security of the whole framework. □

TABLE 1
Notations for operations

Symbol	Operation
$Hash_G$	hash operation $\{0,1\}^* \to \mathbb{G}_1$
Exp_G	group exponentiation operation on $\mathbb{G}_1$
Mul_G	group multiplication operation on $\mathbb{G}_1$
$Pair_G$	group pairing operation on $\mathbb{G}_1$
$Hash_{Z_p}$	hash operation $\{0,1\}^* \to \mathbb{Z}_p$
Mul_{Z_p}	multiplication operation in $\mathbb{Z}_p$
Add_{Z_p}	addition operation in $\mathbb{Z}_p$
C_f	computing a PRF $f(\cdot)$
$\|x\|$	the number of bits of x

5.3 Remark and further discussion

In our proposed BDPA, we show how to integrate our framework with existing auditing schemes by modifying some algorithms. The core difference between our BDPA and those existing schemes is that we adapt the blockchain to generate challenge messages and handle auditing in a transparent and verifiable way. In addition, we posit the potential of a permissioned blockchain combined with the idea of DAO to mitigate the centralization and collusion problems illustrated in Section 3.2.

The claimed decentralization is for the main procedures regards of the auditor in the public auditing scheme, that is, we replace the TPA with the decentralized blockchain which executes the ChallGen and Audit algorithms. We have the centralized PKG which only participates the Setup, KeyGen algorithm. The PKG is the necessary entity in an identity-based auditing scheme, which generates the system parameters and users' private keys. Note that in most existing public auditing schemes [43]–[45], they all have the trustworthy PKG or KGC which is an important and necessary entity in the cryptographic scheme.

There is a potential threat in our BDPA because our construction is based on RANDAO. The last participant to reveal its secret already knows others' revealed numbers and has a degree of influence on the aggregated random number. To be more specific, the last participant can calculate the result with or without their number. If one number is better for them than the other, it has incentive to exert minor degree of manipulation. However, this problem can be mitigated by feeding the aforementioned random number into a verifiable delay function (VDF). A VDF requires a specified number of sequential steps to evaluate, yet produces a unique output that can be efficiently and publicly verified. More details about VDF can be found in [46].

6 Performance Evaluation

In this section, we evaluate the performance of IBPA-BDPA and CPVPA-BDPA which are integrated with two existing auditing schemes IBPA [20] and CPVPA [21] in terms of cryptographic part and blockchain part. All the notations used in the following evaluation are defined in Table 1.

6.1 Implementation setting

In our experiment, there are two parts: cryptographic implementations which are conducted based on the existing public auditing schemes, and blockchain implementation which

TABLE 2
Testing platform information

Operating System	Windows 10
CPU	Intel Core i5-7300HQ CPU 2.50 GHz
Memory	8.00 GB RAM
Configuration	1. nodejs v10.16.3 2. npm v6.9.0 3. truffle v5.0.0 4. ganache v1.2.1

generates random tuples for challenge messages based on the idea of RANDAO.

Crypotographic implementation. In the cryptographic experiment, we implement the public auditing schemes of BDPA in Java with JPBC[6] library in version 2.0 and IntelliJ IDEA platform. In the implementations, we choose a type A pairing with a 160-bit prime p and the base field size is 512-bit. To evaluate the practical performance, we integrate two different auditing schemes into our framework and evaluate the performance. The auditing methods we use are the same as the original auditing schemes and thus the time cost of auditing is the same as them.

Blockchain implementation. In the blockchain experiment, we implement our proposed scheme with Solidity which is a Javascript-like language designed for writing smart contracts in Ethereum. As shown in Table 2, we utilize the truffle suite to write smart contracts and choose Ganache to simulate a private blockchain environment. Our implementation mainly considers core functions in RANDAO. And we deploy the smart contract on our personal computer which runs a private blockchain. Note that the private Ethereum can avoid transaction fees and provide the same accuracy as the public one.

After our design, the smart contract is published in the private chain, we use Web3j in version 4.5.5 to evaluate the functions. Our implementations allow a user to delegate auditing requests and check log files in blockchain. Each node in the blockchain is identified by an address with the corresponding public/private key pair. The user can submit its auditing requests to the blockchain network by executing the functions in the smart contrat. We remark that the verification of the proof information can also be handled by the existing blockchain-based bilinear pairing outsourcing systems such as [47].

6.2 Cryptographic performance evaluation

We show the computation costs on the server side of IBPA-BDPA and CPVPA-BDPA in Table 3, where c denotes the total number of challenged data blocks. In our implementation of IBPA-BDPA and CPVPA-BDPA, the algorithms for tag generation and proof generation are the same with IBPA [20] and CPVPA [21]. Therefore, the computation costs on the user side (tag generation) and server side (proof generation) are the same with original schemes [20], [21]. Since the challenging messages in our construction are computed by a smart contract in the Ethereum blockchain, this requires additional computation costs. However, such additional costs

6. http://gas.dia.unisa.it/projects/jpbc/

TABLE 3
Computation costs on the cloud server side

Scheme	Computation cost
IBPA-BDPA	$(2c+1)\cdot Exp_G + 2c\cdot Mul_G + (c+1)\cdot Mul_{Z_p} + (c+1)\cdot Add_{Z_p} + Hash_{Z_p}$
CPVPA-BDPA	$c\cdot Exp_G + c\cdot Mul_G + c\cdot Mul_{Z_p} + c\cdot Add_{Z_p} + 2c\cdot C_f$

TABLE 4
Computation costs of proof verification

Scheme	Computation cost
IBPA-BDPA	$3\cdot Pair_G + (2c+3)\cdot Exp_G + (2c+1)\cdot Mul_G + Div_G + c\cdot Hash_G + (c+1)\cdot Hash_{Z_p} + c\cdot Mul_{Z_p}$
CPVPA-BDPA	$4\cdot Pair_G + (3c+2)\cdot Exp_G + 3c\cdot Mul_G + (c+4)\cdot Hash_G + (2c+3)\cdot Hash_{Z_p} + 2c\cdot MulZ_p + 2c\cdot C_f$

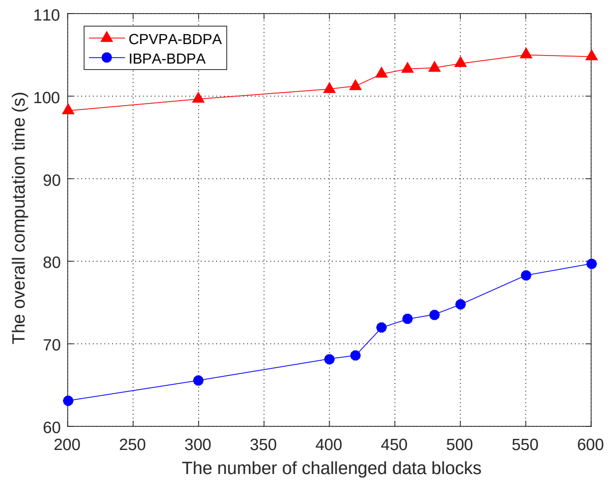


Fig. 7. The overall computation time versus the number of challenged data blocks

ensure that availability of auditing service even when some nodes fail, and also protects from tempting auditors.

We show a comparison of computation costs of proof verification in Table 4, and show the overall computation delay of IBPA-BDPA and CPVPA-BDPA which contains the computation costs on the user side, the server side and verification delay in Fig. 7. In BDPA, the transmition cost between the cloud server and the blockchain is constant, which is the size of the generated challenge messages and the proof information.

6.3 Blockchain performance evaluation

We conduct the following experiment to show the feasibility of our proposed BDPA. Firstly, we compile and deploy the designed smart contract into our private chain and get the contract's address. Then, we test the functionality of the *ChallGen* algorithm. Note that the random tuples generated in the blockchain are not used directly as challenge messages due to the limitaions of representing group elements. The chosen node to verify the proof information can obtain the random value from the Ethereum, and then generate challenge messages locally. For simplicity, we only

TABLE 5
Time cost (in ms) of functions in smart contract

Time	commit	reveal	getRandom	genChallenge
Average	63.9	130.68	59.54	85.2
Min	53	103	50	76
Max	83	158	83	111

TABLE 6
Comparison in security and utility

Scheme	Anti-collusion	Avaliability	Scalability
IBPA [20]	N	N	N
CPVPA [21]	N	N	N
BDPA	Y	Y	Y

use predefined settings to represent the user's auditing task. We remark that our implementations can also be applied in practice to different blockchain platforms.

In order to evaluate the efficiency of BDPA, we execute 1000 times to measure the approximate time cost of core functions related to the *ChallGen* algorithm (i.e. commit, reveal, getRandom and genChallenge). Note that getRandom is a *eth_call* function, which does not trigger a block-generation event. The approximate time cost shown in Table 5 is acceptable in reality compared with the overall time costs of an auditing task.

6.4 Security and utility comparison

As shown in Table 6, we compare the properties of BDPA with IBPA [20] and CPVPA [21] in consideration of security and feasibility. Since IBPA and CPVPA both have the centralized TPA, they are vulnerable to the single point of failure. Moreover, they cannot resist tempting auditors because the challenge information can be biased based on our previous discussion in Section 3.2. In this paper, the proposed BDPA is scalable, available, and has acceptable efficiency and provides strong security with the decentralized architecture.

7 Conclusion and Future work

In the paper, we proposed a blockchain-based decentralized public auditing scheme, called BDPA. We integrated two different mechanisms to show that our scheme is secure, feasible, available and scalable. In addition, we proved the security of BDPA against malicious cloud server and untrusted blockchain nodes. Finally, we implemented each algorithm in BDPA and analyzed its performance in comparison with the existing schemes. Through theoretical analysis and simulation study, we showed that the BDPA scheme has acceptable performance with scalability, availability and better security against tempting auditors.

In regards to future work, we will further investigate how to select node in blockchain to verify the proof information. In addition, we will explore how to integrate some existing blockchain-based bilinear outsourcing schemes to handle the computation of TPA. Moreover, we will design an efficient public auditing scheme and also investigate how to remove the centralized PKG in our future work.

References

[1] Y. Zhang, C. Xu, H. Li, and X. Liang, Cryptographic public verification of data integrity for cloud storage systems, *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 4452, 2016.

[2] J. Xue, C. Xu, and L. Bai, Dstore: A distributed system for outsourced data storage and retrieval, Future Generation Computer Systems, vol. 99, pp. 106-114, 2019.

[3] Y. Zhang, C. Xu, J. Zhao, X. Zhang, and J. Wen, Cryptanalysis of an integrity checking scheme for cloud data sharing, Journal of Information Security and Applications, vol. 23, pp. 68-73, 2015.

[4] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable Computation over Large Database with Incremental Updates," *Computer Security - ESORICS 2014 Lecture Notes in Computer Science*, pp. 148-162, 2014.

[5] C. Liu, C. Yang, X. Zhang, and J. Chen, "External integrity verification for outsourced big data in cloud and IoT: A big picture," *Future Generation Computer Systems*, vol. 49, pp. 58-67, 2015.

[6] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel and Distributed Systems*, vol. 24, no. 9, pp. 1717-1726, 2013.

[7] J. Ni, Y. Yu, Y. Mu, and Q. Xia, "On the Security of an Efficient Dynamic Auditing Protocol in Cloud Storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2760-2761, 2014.

[8] Y. Zhang, C. Xu, N. Cheng, and X. Shen, Secure encrypted data deduplication for cloud storage against compromised key servers, in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 16.

[9] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. of ACM CCS*, 2007, pp. 598-609.

[10] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation, *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 3, pp. 676688, 2017.

[11] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient Remote Data Possession Checking in Critical Information Infrastructures," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 8, pp. 1034-1038, 2008.

[12] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New Publicly Verifiable Databases with Efficient Updates," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 546-556, Jan. 2015.

[13] M. Sookhak, A. Gani, H. Talebian, A. Akhunzada, S. U. Khan, R. Buyya, and A. Y. Zomaya, "Remote Data Auditing in Cloud Computing Environments," *ACM Computing Surveys*, vol. 47, no. 4, pp. 1-34, 2015.

[14] M. Kolhar, M. M. Abu-Alhaj, and S. M. A. El-Atty, "Cloud Data Auditing Techniques with a Focus on Privacy and Security," *IEEE Security & Privacy*, vol. 15, no. 1, pp. 42-51, 2017.

[15] T.-Y. Wu, Y. Lin, K.-H. Wang, C.-M. Chen, J.-S. Pan, and M.-E. Wu, "Comments on a privacy preserving public auditing mechanism for shared cloud data," *Proceedings of the 4th Multidisciplinary International Social Networks Conference on ZZZ - MISNC 17*, 2017.

[16] L. Sun, C. Xu, Y. Zhang, and K. Chen, An efficient iO-based data integrity verification scheme for cloud storage, *Sci. China Inf. Sci.*, vol. 62, no. 5, pp. 59 101:159 101:3, 2019.

[17] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. of ESORICS*, 2009, pp. 355-370.

[18] H. Tian, Z. Chen, C.-C. Chang, Y. Huang, T. Wang, Z.-A. Huang, Y. Cai, and Y. Chen, "Public audit for operation behavior logs with error locating in cloud storage," *Soft Computing*, vol. 23, no. 11, pp. 3779-3792, Jan. 2018.

[19] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847-859, 2011.

[20] J. Xue, C. Xu, J. Zhao, and J. Ma, "Identity-based public auditing for cloud storage systems against malicious auditors via blockchain," *Science China Information Sciences*, vol. 62, no. 3, 2019.

[21] Y. Zhang, C. Xu, X. Lin, and X. S. Shen, "Blockchain-Based Public Integrity Verification for Cloud Storage against Procrastinating Auditors," *IEEE Transactions on Cloud Computing*, to appear, doi:10.1109/TCC.2019.2908400.

[22] F. Armknecht, J.-M. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, "Outsourced Proofs of Retrievability," *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS 14*, 2014.

[23] Y. Zhang, C. Xu, S. Yu, H. Li, and X. Zhang, "SCLPV: Secure Certificateless Public Verification for Cloud-Based Cyber-Physical-Social Systems Against Malicious Auditors," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 159-170, 2015.

[24] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." https://bitcoin.org/bitcoin.pdf.

[25] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," *IEEE International Congress on Big Data (BigData Congress)*, 2017.

[26] J.-H. Hsiao, R. Tso, C.-M. Chen, and M.-E. Wu, "Decentralized E-Voting Systems Based on the Blockchain Technology," *Advances in Computer Science and Ubiquitous Computing Lecture Notes in Electrical Engineering*, pp. 305-309, 2017.

[27] A. Norta, "Designing a Smart-Contract Application Layer for Transacting Decentralized Autonomous Organizations," *Communications in Computer and Information Science Advances in Computing and Data Sciences*, pp. 595-604, 2017.

[28] A. Juels and B. S. K. Jr, "Pors: Proofs of retrievability for large files," in *Proc. of ACM CCS*, 2007, pp. 583-597.

[29] H. Shacham and B. Waters, "Compact proofs of retrievability," *Journal of Cryptology*, vol. 26, no. 3, pp. 442-483, 2013.

[30] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy preserving public auditing for secure cloud storage," *IEEE Trans. Computers*, vol. 62, no. 2, pp. 362-375, 2013.

[31] J. Zhao, C. Xu, F. Li, and W. Zhang, "Identity-Based Public Verification with Privacy-Preserving for Data Storage Security in Cloud Computing," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E96.A, no. 12, pp. 2709-2716, 2013.

[32] X. Zhang, H. Wang, and C. Xu, "Identity-based key-exposure resilient cloud storage public auditing scheme from lattices," *Information Sciences*, vol. 472, pp. 223-234, 2018.

[33] C. Gentry, Practical Identity-Based Encryption Without Random Oracles," *Advances in Cryptology - EUROCRYPT 2006 Lecture Notes in Computer Science*, pp. 445-464, 2006.

[34] S. S. Al-Riyami and K. G. Paterson, Certificateless public key cryptography, in *Proc. of ASIACRYPT*, 2003, pp. 452473.

[35] X. Zhang, J. Zhao, C. Xu, H. Li, and Y. Zhang, Cipppa: Conditional identity privacy-preserving public auditing for cloud-based wbans against malicious auditors, *IEEE Transactions on Cloud Computing*, to appear, doi: 10.1109/TCC.2019.2927219.

[36] C. Pierrot and B. Wesolowski, "Malleability of the blockchains entropy," *Cryptography and Communications*, vol. 10, no. 1, pp. 211-233, 2018.

[37] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, Healthdep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems, *IEEE Trans. Ind. Informatics*, vol. 14, no. 9, pp. 41014112, 2018.

[38] J. Xue, C. Xu, and Y. Zhang, Private blockchain-based secure access control for smart home systems, *KSII Trans. Internet Inf. Syst.*, vol. 12, no. 12, pp. 60576078, 2018.

[39] Y. Zhang, C. Xu, J. Ni, H. Li, and X. Shen, Blockchain-assisted public-key encryption with keyword search against keyword guessing attacks for cloud storage, *IEEE Transactions on Cloud Computing*, to appear, doi:10.1109/TCC.2019.2923222.

[40] Y. Zhang, C. Xu, N. Cheng, H. Li, H. Yang, and X. Shen, Chronos$^+$: An accurate blockchain-based time-stamping scheme for cloud storage, *IEEE Trans. Serv. Comput.*, vol. 13, no. 2, pp. 216229, 2020.

[41] N. Szabo, "Formalizing and Securing Relationships on Public Networks," *First Monday*, vol. 2, no. 9, Jan. 1997.

[42] P. Catchlove, "Smart Contracts: A New Era of Contract Use," *SSRN Electronic Journal*, 2017.

[43] Y. Tu, J. Gan, Y. Hu, R. Jin, Z. Yang, and M. Liu, Decentralized identity authentication and key management scheme, in *2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2)*, 2019, pp. 26972702.

[44] Y. Miao, Q. Huang, M. Xiao, and H. Li, Decentralized and privacy-preserving public auditing for cloud storage based on blockchain, *IEEE Access*, vol. 8, pp. 139 813139 826, 2020.

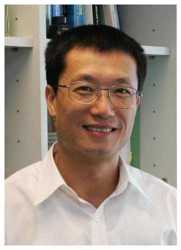
[45] X. Jia, N. Hu, S. Su, S. Yin, Y. Zhao, X. Cheng, and C. Zhang, Irba: An identity-based cross-domain authentication scheme for the internet of things, *Electronics*, vol. 9, no. 4, p. 634, 2020.

[46] B. Wesolowski, "Efficient Verifiable Delay Functions," *Advances in Cryptology - EUROCRYPT 2019 Lecture Notes in Computer Science*, pp. 379-407, 2019.

[47] C. Lin, D. He, X. Huang, X. Xie, and K.-K. R. Choo, Blockchain-based system for secure outsourcing of bilinear pairings, *Information Sciences*, vol. 527, pp. 590 601, 2020.

Jiangang Shu (M'17) received the Ph.D degree in computer science from City University of Hong Kong (CityU), Hong Kong, in 2019. He is currently a Research Scientist in Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China. He was a postgraduate visiting student in Secure Mobile Centre, Singapore Management School, Singapore, and a Postdoctoral Fellow in CityU. His research interests include AI privacy, crowdsourcing & cloud security, IOT security, applied cryptography, data security and privacy, searchable encryption.

Xing Zou (M'18) received the B.Eng. degree in computer science and technology from Wuhan University, Wuhan, China, in 2012. She is currently a master student in School of Computer Science and Technology at Harbin Institute of Technology (Shenzhen), Shenzhen, China. Her research interests are blockchain technology and data security.



Xiaohua Jia (F'13) received his BSc (1984) and MEng (1987) from University of Science and Technology of China, and DSc (1991) in Information Science from University of Tokyo. He is currently Chair Professor with Dept of Computer Science at City University of Hong Kong. His research interests include cloud computing and distributed systems, computer networks and mobile computing. Prof. Jia is an editor of IEEE Internet of Things, IEEE Transactions on Parallel and Distributed Systems (2006-2009), Wireless Networks, Journal of World Wide Web, Journal of Combinatorial Optimization, etc. He is the General Chair of ACM MobiHoc 2008, TPC Co Chair of IEEE GlobeCom 2010 Ad Hoc and Sensor Networking Symposium, Area-Chair of IEEE INFOCOM 2010 and 2015. He is Fellow of IEEE.

Weizhe Zhang (SM'19) is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology, China. He has authored over 100 academic papers in journals, books, and conference proceedings. His research interests are primarily in parallel computing, distributed computing, cloud and grid computing, and computer network. He serves on a number of journal editorial boards. He has edited more than 10 international journal special issues as a guest editor and has served for many international conferences as the chair or a committee member.

Ruitao Xie received the B.Eng. degree from the Beijing University of Posts and Telecommunications in 2008 and the Ph.D. degree in computer science from the City University of Hong Kong in 2014. She is currently an Assistant Professor with the College of Computer Science and Software Engineering, Shenzhen University. Her research interests include AI networking and mobile computing, distributed systems, and cloud computing.