IPAPA: Incentive Public Auditing Scheme against Procrastinating Auditor

Ying Miao[1], Qiong Huang[1,2*], Meiyan Xiao[1], Willy Susilo[3]

[1] College of Mathematics and Informatics, South China Agricultural University, China
[2] Guangzhou Key Laboratory of Intelligent Agriculture, China
[3] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, Australia
yingmiao2021@163.com, qhuang@scau.edu.cn, maymayxiao@scau.edu.cn, wsusilo@uow.edu.au

Abstract

Cloud storage provides convenience in managing data for users. Data integrity becomes important because data owner (DO) loses control of their data once it is uploaded to the cloud server (CS). Public auditing is used to check data integrity in cloud storage. Traditional public auditing schemes introduce a third-party auditor (TPA) to help users check their data. However, TPA is assumed to be trusted in these schemes, which may not be practical. A dishonest TPA may provide a good report to DO without executing the auditing task timely. If the data loss could not be detected timely, it may cause a great loss to DO. In this paper we aim to solve these problems using blockchain technique. In our scheme DO, TPA and CS interact with blockchain via smart contracts. We utilize a time-locked deposit smart contract to incentive TPA and CS for their fulfillment in the auditing task honestly. Otherwise, they would be amerced. We use storage smart contracts to ensure the auditing process transparency, and utilize zero-knowledge proof to protect DO's privacy. The scheme is extended to support batch auditing to reduce the user's cost. Experimental results show that our scheme is efficient and practical.

Keywords: Blockchain, Public auditing, Incentive, Procrastinating auditor

1 Introduction

With the rapid development of Internet of Things (IoT) and Artificial Intelligence (AI), the digital world is thriving. Various data generated in our daily life promote a rapid demand of cloud storage. Cloud storage service provides a convenient, fast and scalable environment, making storage become easier, thus many infrastructures utilize cloud storage. There are many cloud storage services providers (CSP), such as Amazon AWS Cloudfront, Akamai, Google Drive and etc. According to Gartner[1], public cloud service market worldwide will grow by 17% in 2020, from 227.8 billion dollars in 2019 to 266.4 billion dollars in 2020. However, cloud storage also brings problems [1], such as single point of failure, data breach, data privacy, and data loss. Particularly, data loss accidents are growing rapidly. According to the research by Gemalto[2], there were about 2.6 billion pieces of data stolen, lost or compromised globally in 2017. As valuable resources, data loss may be fatal for data owners or companies. Cloud storage services would hide data loss accidents for their own interests. Therefore, people have tried to seek a way to find out data loss timely in recent years. Public data integrity verification technique [2-3] emerged to solve it, which provides a way for the cloud to prove data possession without transmitting all the data. In order to further provide convenience for users, it introduces a trusted third-party auditor. The traditional third-party auditing model includes three entities, as shown in Figure 1. First of all, the DO splits its data into blocks, signs each one, and outsources the data blocks with the corresponding signatures to the cloud. Secondly, the DO delegates a TPA to check the data integrity timely. After receiving the auditing task, TPA chooses a random subset of data blocks as a challenge. The CS then responds with a piece of proof information according to the challenge. Finally, the TPA verifies the proof and sends the auditing result to the DO.

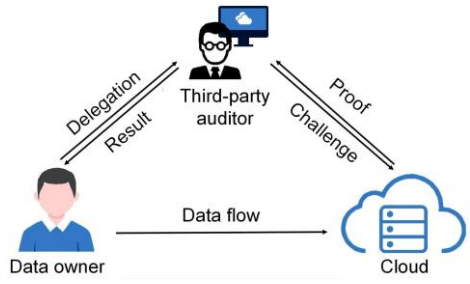


Figure 1. Traditional public auditing

1.1 Problem Statement

Public auditing has been studied for many years, such as privacy preservation for data users, dynamic data auditing, auditing in group data sharing and etc., e.g. [4-11]. However, there are two major problems in these schemes.

1.1.1 On the Vulnerability of Malicious Auditors

In most existing public auditing schemes [4, 5-7, 10-11], users usually do not take participate in the auditing process, thus the auditor is assumed to be honest and reliable. However, if the auditor is dishonest/malicious, it may not conduct the

[1] https://www.gartner.com/en/newsroom/press-releases/2019-11-13-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2020

[2] https://www.globaldots.com/blog/2-6-billion-records-were-stolen-lost-or-exposed-worldwide-in-2017

*Corresponding Author: Qiong Huang; E-mail: qhuang@scau.edu.cn
DOI: 10.53106/160792642022122307006

auditing but always generate a 'fake' data integrity report. Especially, most cloud service providers and third-party auditing services charge for these services. In this case, it is unfair for users since they pay for the service and should have the right to audit the auditor's behavior at the end of each auditing task instead of only knowing the checking result, e.g., $true$ or $false$. Only a few schemes [12-13] consider the loss of users, but these schemes can not resist a procrastinating auditor.

1.1.2 On the Vulnerability of Procrastinating Auditors

Existing public auditing schemes require the auditor to execute auditing tasks periodically in order to find data corruption as soon as possible. Time-sensitive is important in many systems. For example, in the cloud-assisted electronic health system, the electronic health records (EHRs) is time-sensitive [14-15] since the information relates to the state of an illness for patients. What is more, it relates to a patient's life in case of an emergency. Therefore, it is important to check the data's integrity periodically. However, most existing public auditing schemes could not resist a procrastinating auditor. Usually, the agreed frequency of data integrity check is weekly, monthly, or quarterly [16-17] which is not high, due to the cost issue and the burden on the CS side. An honest auditor would perform the check as scheduled, however, a procrastinating auditor may not conduct the auditing task timely until the last second. To avoid such a case, there should be some way to check whether the auditor honestly finishes its obligation.

In order to address these problems, many researchers have studied public data auditing based on blockchain (and smart contracts) technology recently, such as [12-13, 16, 18, 33-34]. However, these schemes only utilize blockchain technique to check the correctness of the proof information but could not resist the procrastinating auditor. And none of these schemes take both of these problems into consideration simultaneously.

1.2 Our Contributions

In this paper, we study the construction of a public auditing scheme aiming to solve the aforementioned problems. We utilize the blockchain technology to protect the interest of DO and to resist the procrastinating TPA and CS. The core idea of our scheme is to use cryptocurrency and smart contracts in blockchain to reward or amerce the TPA and CS automatically according to their fulfillment. Overall, our contributions in this paper could be summarized as follows.

- We analyze the existing problems in recent public auditing schemes and then demonstrate that existing schemes rely on a trust TPA. It cannot resist a procrastinating auditor who may not perform the auditing task but always generate a good report for DO, which violates the core idea of public auditing, i.e., finding the data corruption as soon as possible.
- We propose a public auditing scheme (IPAPA) with incentives based on blockchain, which can resist a malicious auditor and procrastinating ones. We design blockchain-based smart contracts to achieve the auditing process transparency and authenticity and use a time-locked deposit smart contract to resist the procrastinating auditor. Our scheme does not depend on any centralized third party to accomplish the auditing task and can also guarantee the user's privacy by making use of zero-knowledge proof in the auditing process.
- We further extend IPAPA to support batch auditing of user data, in order to reduce the cost on the data owner side. Finally, we implement the proposed scheme based on the public test network of Ethereum. Experimental results show that our scheme is practical and efficient.

1.3 Paper Organization

The remainder of this paper is organized as follows. We introduce related works in Section 2 and preliminaries in 3. In Section 4 we show the system model, threat model and design goals. The proposed public auditing scheme with incentive is described in Section 5. We then present the security analysis and efficiency analysis in Sections 6 and 7, respectively. Finally, we draw the conclusion in Section 8.

2 Related Works

2.1 Provable Data Procession

2.1.1 Traditional Public Auditing

To check the integrity of cloud storage data, Juels et al. [2] proposed a *proof-of-retrievability* (POR) scheme in 2007, which utilizes indistinguishable blocks hidden in file blocks as sentinels to detect data corruption. However, their POR scheme does not take public auditing into consideration and does not support the unbounded number of challenge queries either. Later, Ateniese et al. [3] firstly proposed a public verification scheme based on proof of data possession (PDP), which adopts the challenge and response model and supports unbounded number of challenge queries. Nevertheless, their scheme cannot protect user's privacy. Shacham and Waters [19] then proposed a public-private key based POR scheme, which utilizes homomorphic authenticators to generate compact proofs and supports public auditing.

Following Shacham et al.'s work, many public auditing schemes have been proposed in recent years. These schemes solved different problems in public auditing, such as privacy preservation, data dynamic, data sharing in group and etc. To check the integrity of the outsourced data without leaking any privacy information of the DO, some schemes utilize specific signature technique. For instance, Wang et al. [4] utilized ring signature to protect the signer's privacy, and Li et al. [18] utilized online/offline signatures to reduce online computation overhead with the offline pre-computed results. While some other schemes [5-7] utilized random masking technique or zero knowledge proof to hide data information during the response phase. To support public auditing and data dynamics, existing schemes usually utilized Merkle Hash Tree (MHT) or index table to support data blocks addition, insertion, modification and deletion, such as [8-11, 31-32]. MHT based schemes usually need heavy computation cost and communication cost, while index table solves this problem commendably. Furthermore, there are many schemes [20-21] which focus on checking the integrity of data shared in a group. Many of these schemes utilized proxy-resignature to deal with user revocation.

2.1.2 Blockchain-based Public Auditing

Recently, many researchers have focused on decentralized provable data possession. Specifically, the emerging blockchain technology provides a good solution to provable data possession. In order to resist a malicious TPA, Armknecht et al. [22] firstly utilized Bitcoin as the unpredictable pseudorandomness provider to generate challenge information. Following Armknecht et al.'s work, many schemes [16, 23-26] utilized random block hash or nonce in block to generate challenge blocks. Some schemes utilized blockchain to store information. Yu et al. [30] utilized consortium chain to store audit information. However, it needs the DO to verify the correctness of proof information. Li et al. [18] utilized public chain to store file information to remove central third-party auditor. However, their scheme needs the data user to generate a challenge. Huang et al. [27] utilized three different transactions to record data dynamics in blockchain. Some schemes utilized smart contracts to realize fair judgement recently. For instance, Wang et al. [12] utilized smart contract to achieve fairness in judgement in their scheme. Concretely, smart contract instead of the TPA checks the correctness of proof information. If the check fails, the CS would be amerced according to the smart contract; otherwise, the CS would receive a reward for its honest service. Yuan et al. [13] utilized the same idea to achieve data deduplication and fair arbitration. However, these schemes [12-13] cannot resist a procrastinating auditor.

3 Preliminaries

3.1 Bilinear Pairings

Let G_1 and G_T be two multiplicative cyclic groups of prime order p, respectively, g be a generater of G_1. The map $e: G_1 \times G_1 \to G_T$ is a bilinear pairing if it satisfies the following properties: (1) **Bilinearity**: for all $u, v \in G_1$ and $a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$; (2) **Computability**: for any $u, v \in G_1$, $e(u, v)$ could be efficiently computed; and (3) **Non-degeneracy:** $e(g, g) \neq 1_T$, where 1_T is the identity of G_T.

3.2 Mathematical Assumptions

Let G be a cyclic group of prime order p, g be a random generator of G, and a, b be random elements of Z_p.

- **Computational Diffie-Hellman (CDH) Assumption** Given (g, g^a, g^b), it is computationally intractable to compute g^{ab}. That is, for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the probability of solving CDH problem is negligible, namely,
  $$Pr[a, b \leftarrow \mathcal{A}_{CDH}(g, g^a, g^b): Z = g^{ab}] \leq \varepsilon,$$
  where ε is a negligible function in n.
- **Discrete Logarithm(DL) Assumption** Given h ∈ G, it is computationally intractable to compute $h = g^a$, That is, for any PPT adversary $\mathcal{A}$, the probability of solving DL problem is negligible, namely,
  $$Pr[a \leftarrow \mathcal{A}_{DL}(g, h): a = log_g h] \leq \varepsilon.$$

3.3 Blockchain Structure

The blockchain structure is shown in Figure 2. A block usually contains two parts: block header and block body. Block header stores the basic information of a block, including the previous block hash ($PreBlockHash$), the current block hash ($BlockHash$), the time when the block was generated ($timestamp$), a nonce, the current difficulty to find a solution ($difficulty$) and some other basic information. The block body stores the origin data which is packaged as a series of transactions and stored as a Merkle tree. A transaction usually contains: from (message sender), to (message recipient), value (from the sender to the recipient), data (sent to the recipient) and gas.

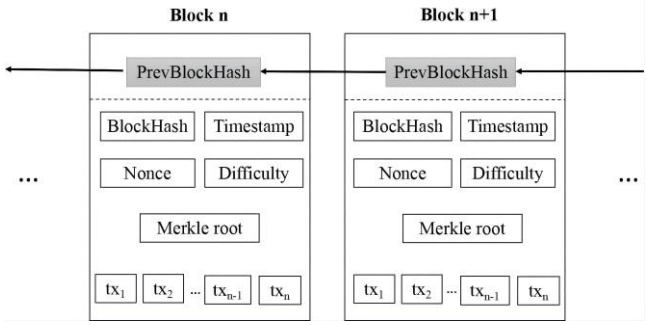


Figure 2. Blockchain structure

4.Public Auditing Scheme with Incentive

4.1 System Model Description

There are three entities in decentralized public auditing scheme with incentive, i.e. data owner (DO), cloud server (CS) and third-party auditor (TPA), as shown in Figure 3. Different from traditional public provable data possession schemes, the CS and TPA not only interact with each other but also interact with the blockchain based on smart contracts.

(1) **Public Provable Data Possession Algorithms**

- **Setup**$(\mathbf{1^\lambda}) \to$ **pp.** It is run by the DO. It takes as input a security parameter 1^λ, and outputs the public parameter pp.
- **KeyGen**$(\mathbf{pp}) \to (\mathbf{pk}, \mathbf{sk})$**.** It is run by the DO. It takes as input pp, and outputs a public/secret key pair (pk, sk).

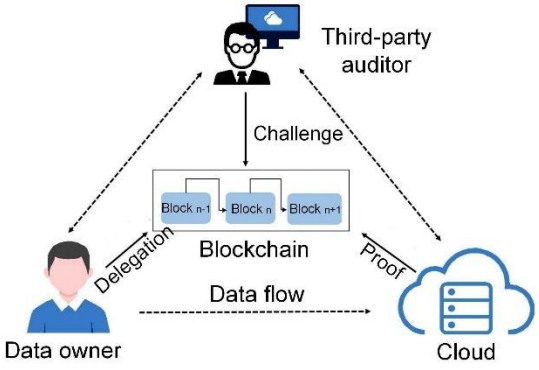


Figure 3. Decentralized system model

- **AuthGen(sk, F) → authenticators.** It is run by the DO. It takes as input sk and a file $F = (m_1, m_2, \cdots, m_n)$, and outputs a set of authenticators σ_{fi} and public verification parameters for F.
- **Proof(TPA, CS) → True/False.** It is run interactively between the TPA and CS. Their common input includes pp and pk.

(2) **Smart Contract Design**

- **Delegation Contract (DLC):** The contract is executed by the DO. It specifies three entities, DO, TPA and CS, and contains file information and task description.
- **Time-locked Deposit-Withdraw Contract (TDWC):** It is executed by the entities DO, TPA and CS, which should deposit some coins according to the task description based on DLC first.
- **Challenge Information Storage Contract (CISC):** It is executed by the TPA. It records the challenge information during the auditing process.
- **Proof Information Storage Contract (PISC):** It is executed by the CS. It records the proof information during the audit process.

4.2 Formal Security Definition of Data Integrity

Data auditing could be passed only if the CS maintains data intactly. Consider the following game played between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$.

(1) Challenger $\mathcal{C}$ calls KeyGen(pp) to generate (pk, sk) and gives pk to $\mathcal{A}$.

(2) $\mathcal{A}$ interacts with $\mathcal{C}$ and makes queries adaptively for some file blocks of F for polynomially many times, $\mathcal{C}$ runs $AuthGen(sk, F)$ and outputs the corresponding tag information of F.

(3) Finally, $\mathcal{A}$ outputs proof information for file F', the proof information includes at least one file block which does not be queried before and can pass verification.

Define the advantage of $\mathcal{A}$ is $Adv_{\mathcal{A}} = \Pr[Proof(TPA, CS) = True]$. We say the adversary wins the above game, if $Adv_{\mathcal{A}}$ is non-negligible.

Definition 1. A public auditing scheme with incentive satisfies data integrity if no PPT adversary $\mathcal{A}$ could win the game above with non-negligible probability.

4.3 Other Security Properties

A secure decentralized public auditing scheme with incentive should also satisfy the following security properties.

Data privacy: It requires the auditing process should preserve the DO's data privacy so that no one could acquire any information about the data through the whole process.

Auditing process publicity and non-modifiability: It requires the auditing process should be public and tamper-proof, in order to prohibit the TPA or CS from cheating.

Timeliness: It requires that the TPA should conduct the auditing task and return the auditing result timely, in order to resist the procrastinating TPA.

Incentive: It requires any dishonest party who does not finish the auditing task timely should be monetarily penalized and honest parties could obtain corresponding rewards fairly.

Batch auditing: It requires multiple auditing tasks from the same DO could be verified simultaneously, in order to reduce gas cost for DO.

5 Our Public Auditing Scheme with Incentive

5.1 Blockchain-based Incentive Mechanism

We design an incentive mechanism based on blockchain and smart contracts. Basically, each entity in the system deposits some coins on the blockchain system, and if the TPA and CS fulfill their obligation honestly, they would be rewarded; otherwise, they would be amerced. Some works have been done to achieve time-locked blockchain deposit protocols, e.g. [28-29], which enable a party (payer) to exchange with other parties (payee) to lock a certain number of coins as guarantee deposit on blockchain. To prevent any malicious party who redeems the deposit arbitrarily, the party cannot redeem the deposit until the deadline even if it owns the secret key.

To make the incentive automatically, smart contract is utilized. Concretely, we design four smart contracts, including *Delegation Contract (DLC)*, *Time-locked Deposit-Withdraw Contract (TDWC)*, *Challenge Information Storage Contract (CISC)* and *Proof Information Storage Contract (PISC)*. Data structure and relationship of the contracts are shown in Figure 4. The left part of Figure 4 shows the data structure of each contract, and the right part shows the relationship between the entities and the contracts. All the functions of each contract are described inside the corresponding box.

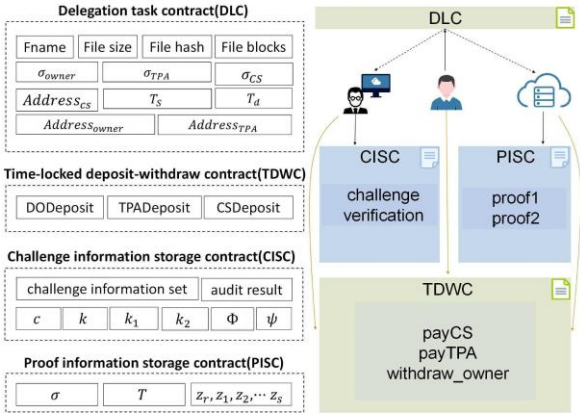


Figure 4. The structure of smart contracts and the call relationship

Delegation Contract (DLC). Before an auditing task starts, the DO should post the task. The delegation contract includes basic file information, e.g. file name, size, hash and etc. It publishes the auditing task execution time period, e.g. task start time T_s and task end time T_d. The contract also includes three wallet addresses/public keys, e.g. DO address, TPA address, and CS address. Only the one with the secret wallet key could transfer and sign the task.

Time-locked Deposit-Withdraw Contract (TDWC). To improve the credibility, each entity in the system is required to make a time-locked deposit as a guarantee to regulate its behavior and to follow the protocol, which will be assigned to designated entities by the predefined smart contracts after the

appointed time. The contract can only be executed by the DO, CS and TPA, respectively, described as below.

(*Deposit phase*). Each auditing task starts from time T_s and ends at time T_d. Before T_s, the three entities should deposit a certain number of coins in the contract. The coins are then locked on the contract until T_d, and the deposit could be withdrawn later, which is controlled by the contract based on if the task is completed successfully. The deposit process of TDWC contract is shown in Table 1.

(*Incentive phase*). We utilize interactive public provable data possession scheme. In order to prevent the TPA or CS from being procrastinating, we propose to use the reward-or-amerce and time locked deposit to improve their positivity. The reward-or-amerce depends on different states of the TPA and CS. In this paper we define six different states, including T_{s_1}, T_{s_2}, T_{s_3} for TPA, and C_{s_1}, C_{s_2}, C_{s_3} for CS. As shown in Table 2 and Figure 5, there are four cases in the auditing process.

The first case is (C_{s_1}, T_{s_1}), meaning the TPA does not raise a challenge for auditing as scheduled, and its deposit would then be deducted and transferred to the DO. The second case includes (C_{s_1}, T_{s_2}) and (C_{s_2}, T_{s_2}), meaning the CS does not enter the first proof phase and the second phase timely respectively, and its deposit would be deducted and transferred to the DO. The third case is (C_{s_3}, T_{s_2}), meaning the TPA does not finish the verification timely, and its deposit would be deducted and transferred to the DO. The last case is (C_{s_3}, T_{s_3}), meaning both the TPA and CS fulfill their obligation in the auditing task, and the TPA would get the service fee fee_{TPA}. Besides, if the auditing result shows that the DO's data is intact, the CS should be rewarded for the data storage service; otherwise, $deposit_{CS}$ would be deducted and transferred to the DO.

Table 1. The deposit process of TDWC

DODeposit. Before T_s, the DO sends $DODeposit_i := \{owner_i, address, now, T_s, T_d, coins(fee_{TPA} + fee_{CS})\}$ to the blockchain, where $coins(fee_{TPA} + fee_{CS})$ is the deposit that can only be unlocked after T_d and will be transferred into CS account and TPA account only if CS and TPA fulfill their tasks delegated by the DO honestly.
CSDeposit. Before T_s, the CS should be authorized and receive its storage task (from the DO). The CS sends $CSDeposit := \{Address_{CS}, now, T_s, T_d, coins(deposit_{CS})\}$ to the blockchain, where $coins(deposit_{CS})$ is the deposit that can only be unlocked T_d, and is also the fine if the CS does not pass the auditing check, i.e. the CS does not store the DO's data intactly.
TPADeposit. Before T_s, TPA should be authorized and receive its auditing task from DO. The TPA sends $TPADeposit := \{Address_{TPA}, now, T_s, T_d, coins(deposit_{TPA})\}$, where $coins(deposit_{TPA})$ is the deposit that can only be unlocked after T_d, and is also the fine if the TPA does not execute the auditing task as scheduled.

Table 2. Incentive based on four different states

States	T_{s_1}	T_{s_2}	T_{s_3}
C_{s_1}	$deposit_{TPA} \to DO$ $deposit_{CS} \to CS$ $fee_{TPA} + fee_{CS} \to DO$	$deposit_{TPA} \to TPA$ $deposit_{CS} \to DO$ $fee_{TPA} + fee_{CS} \to DO$	——
C_{s_2}	——	$deposit_{CS} \to DO$ $deposit_{TPA} \to TPA$ $fee_{TPA} + fee_{CS} \to DO$	——
C_{s_3}	——	$deposit_{TPA} \to DO$ $deposit_{CS} \to CS$ $fee_{TPA} + fee_{CS} \to DO$	$True$ $deposit_{TPA} + fee_{TPA} \to TPA$ $deposit_{CS} + fee_{CS} \to CS$ $False$ $deposit_{TPA} + fee_{TPA} \to TPA$ $deposit_{CS} + fee_{CS} \to DO$

Storage Contracts. The storage contracts include *challenge information storage contract (CISC)* and *proof information storage contract (PISC)*. CISC includes functions $challenge$ and $verification$. It can only be executed by the TPA before T_d, and requires that the TPA should have enough balance on the smart contract. The $challenge$ function stores the challenge information generated by the TPA, and the $verification$ function stores the verification result. PISC includes functions $proof1$ and $proof2$. It can only be executed by the CS before T_d, and requires that the CS should have enough balance on the smart contract. Both $proof1$ and $proof2$ store the proof information returned by the CS.

5.2 Our Scheme

In this part we introduce our decentralized public auditing scheme with incentive. Formally, our scheme works as follows.

(1) $\boldsymbol{pp \leftarrow Setup(1^{\lambda})}$. Taking a security parameter 1^{λ} as input, the algorithm chooses a large prime p, two multiplicative cyclic groups G, G_T, a generator g of G, a bilinear pairing $e: G \times G \to G_T$, cryptographic hash functions $H_0, H_1: \{0,1\}^* \to G$, $H_2: \{0,1\}^* \to Z_p^*$, a pseudorandom function $\pi: Z_p^* \times [1,n] \to Z_p^*$ and a pseudorandom permutation $\phi: Z_p^* \times [1,n] \to [1,n]$. Besides, the algorithm

picks at random $h, u_1, u_2, \ldots, u_s \in G$ and computes $\eta = \mathrm{e}(\mathrm{g}, \mathrm{h})$. It outputs the public parameter $pp = (p, G, G_T, g, e, H_0, H_1, H_2, h, u_1, u_2, \ldots, u_s, \eta, \pi, \phi)$.

(2) $\boldsymbol{KeyGen(pp)}$**.** The algorithm takes the public parameter pp as input, and generates a signing key pair (spk, ssk) with $spk = g^{ssk}$ and another key pair $(\alpha \in Z_p, \mathrm{v} = g^{\alpha})$ which is used for generating authenticators of file blocks. It outputs $(pk, sk) = ((spk, v), (\alpha, ssk))$ as the public/secret key pair of the DO. Let η_i denote $e(u_i, v)$ for $i \in [1, s]$, which could be pre-computed by the relevant entities given the public key v and pp.

(3) $\boldsymbol{AuthGen(sk, F)}$. The algorithm takes a file F and the DO's secret key as input. It firstly applies an erasure code such as RS code on F to obtain an encoded version F', and splits F' into n blocks. Let f denote the unique identifier of file F. Each block is further fragmented into s sectors $\{\{m_{fi,j}\}_{j=1}^{s}\}_{i=1}^{n}$, which is an element of Z_p. The DO selects a unique file name F_{fn} from a sufficiently large domain. Let $t_{fn} = F_{fn}||n$. The DO computes $T_{fn} = H_0(t_{fn})^{ssk}$ and sets the file tag $f_{t_{fn}} = t_{fn}||T_{fn}$. For each $i \in [1, n]$, the DO computes an authenticator σ_{fi} for block i as $\sigma_{fi} = (H_1(F_{fn}||fi) \cdot \prod_{j=1}^{s} u_j^{m_{fi,j}})^{\alpha}$.

The DO then uploads $\{f_{t_{fn}}, \{\{m_{fi,j}\}_{j=1}^{s}, \sigma_{fi}\}_{i=1}^{n}\}$ to the cloud. Finally, it posts an auditing task based on DLC to the blockchain, as shown in Table 3.

Table 3. Post delegation task based on delegation contract (DLC)

Input: file name, file size, file hash, file blocks, Owner's signature, TPA address, TPA's signature, CS address, CS's signature, r, start_time, end_time, delegation fee, CS fee, CS deposit, TPA deposit.
Output: Update contract DLC
require(now<T_s);
$owner = msg.sender$;
owner_message=prefixed(keccak256(msg.sender, file name, file size, file hash, file blocks, r));
CS_message=prefixed(keccak256($Address_{CS}$, file name, file size, file hash, file blocks, r));
TPA_message=prefixed(keccak256($Address_{TPA}$, file name, file size, file hash, file blocks, r));
require(recoversigner(owner_message,σ_{owner}) ==owner);
require(recoversigner(CS_message, σ_{CS}) == $Address_{CS}$);
require(recoversigner(TPA_message, σ_{TPA}) ==$Address_{TPA}$);
TPA ← $Address_{TPA}$; csp← $Address_{CS}$;
start_time ← T_s; end_time ← T_d;
$fee_{TPA} = \_fee_{TPA}$; $fee_{CS} = \_fee_{CS}$;
$deposit_{CS} = \_deposit_{CS}$;
$deposit_{TPA} = \_deposit_{TPA}$;

(4) $\boldsymbol{True/False \leftarrow Proof(P(F, \{\sigma_i\}, f_t), V(pk))}$. After the DO launches an auditing task, the DO, TPA and CS make their deposit respectively via the deposit process of the TDWC contract. Then they run the following interactive proof protocol.

Individual file auditing: The auditing task refers to a single file F_{fn}.

(a) The TPA chooses at random $k_1, k_2, k, \psi \in Z_p$ and a random integer $c \in [1, n]$, and computes $\Psi = g^k h^{\psi}$. It sends the commitment Ψ and challenge $chal = \{c, k_1, k_2\}$ to the CS, and sends the challenge to the blockchain based on CISC as in Table 4. Finally, the TPA sends the blockheight and transaction id $C = \{C_{ht}, C_{ix}\}$ to the CS.

Table 4. Challenge Information Storage Contract (CISC)

Output: Update contract CISC
enum ContractStates{open, midstate, closed}
enum TPAStates{$T_{s_1}, T_{s_2}, T_{s_3}$}
ContractStates contractstate←ContractStates.open;
TPAStates tpastate←TPAStates. T_{s_1};
function challenge ($uint[c]\_index, uint[c]\_v_c$, $uint$ _Ψ) $public$
require(msg.sender==$Address_{TPA}$);
require(now<T_d);
require($balanceOf[msg.sender]$> $deposit_{TPA}$);
require(tpastate==$TPAStates.T_{s_1}$);
require(contractstate==ContractStates.open);
$Index \leftarrow \_index$; $\_V_c = \_v_x$; $\Psi = \_\Psi$;
tpastate← $TPAStates.T_{s_3}$;
contractstate←ContractStates.midstate;
function verification ($uint$ _k, $uint$ _ψ, bool_audit result)
require(msg.sender==$Address_{TPA}$);
require(now<T_d);
require(balanceOf[msg.sender]> $deposit_{TPA}$);
require(tpastate== $TPAStates.T_{s_2}$);
require(contractstate==ContractStates.midstate);
$k \leftarrow \_k$; $\psi \leftarrow \_\psi$;
audit result← audit result;
tpastate← $TPAStates.T_{s_3}$;
contractstate←ContractStates.closed;

(b) Upon receiving $chal$, the CS computes

$$K_1 = H_2(F_{fn}||k_1||nonce_1||nonce_2|| \cdots ||nonce_{\varphi}),$$
$$K_2 = H_2(F_{fn}||k_2||nonce_1||nonce_2|| \cdots ||nonce_{\varphi}),$$

where $nonce_1$, $nonce_2$, …, $nonce_{\varphi}$ are φ successive random numbers in blocks from height $C_{ht} - \varphi + 1$ to C_{ht}. Then it sets the challenge set as $C_f = \{(i_f, v_{i_f})\}_{i \in [1,c]}$, where $i_f = \phi(K_1, i)$, $v_{i_f} = \pi(K_2, i)$, chooses at random $r, \rho_r, \rho_1, \ldots, \rho_s \in Z_p^*$, computes $\sigma = \mathrm{h}^r \prod_{(i_f, v_{i_f}) \in C_f} \sigma_{fi_f}^{v_{i_f}}$, $T = \eta^{\rho_r} \prod_{j=1}^{s} \eta_j^{\rho_j}$, and executes the $proof1$ function of contract PISC (see Table 5). The CS uploads σ and T to the blockchain, and sends the blockheight and transaction id $P_1 = \{P1_{ht}, P1_{ix}\}$ to the TPA.

(c) After receiving $P1$, the TPA sends (k, ψ) to the CS.

(d) The CS checks if $\Psi = g^k h^{\Psi}$, and aborts if it does not hold. It then computes $z_r = \rho_r - kr$, and

$$\forall j \in [1, s], \mu_j = \sum_{(i_f, v_{i_f} \in C_f)} v_{i_f} m_{fi_f, j},\ z_j = \rho_j - k\mu_j.$$

It executes the $proof2$ function of PISC, uploads $z_r, z_1, \cdots, z_s$ to the blockchain, and sends the blockheight and transaction id $P_2 = \{P2_{ht}, P2_{ix}\}$ to the TPA.

(e) After receiving $P2$, the TPA verifies the file tag $f_{t_{fn}}$ by checking if $e(g, T_{fn}) = e(spk, H_0(t_{fn}))$ and

$$\left(\frac{e(\sigma, g)}{e(\prod_{(i_f, v_{i_f}) \in chal} H_1(F_{fn}||fi_f)^{v_{i_f}}, v)}\right)^k = \frac{T}{\eta^{z_r} \eta_1^{z_1} \eta_2^{z_2} \ldots \eta_s^{z_s}},$$

and aborts if either equation fails to hold. It then uploads the auditing result to the blockchain via $verification$ function of CISC and sends the blockheight and transaction id $V = \{V_{ht}, V_{ix}\}$ to the CS.

Table 5. Proof Information Storage Contract (PISC)

Output: Update contract PISC
enum ContractStates{open,midstate,closed}
enum CSStates{C_{s_1},C_{s_2},C_{s_3}}
ContractStates contractstate←ContractStates.open;
CSStates csstate←CSStates. C_{s_1};

function $proof1(uint\_\sigma, uint\_T)$ public
1 require(msg.sender==$Address_{CS}$);
2 require(now<T_d);
3 require(balanceOf[msg.sender]> $deposit_{CS}$);
4 require(csstate==$CSStates.\ C_{s_1}$);
5 require(contractstate==ContractStates.open);
6 $\sigma \leftarrow \_\sigma$; $T = \_T$;
7 csstate← $CSStates.C_{s_2}$;
8 contractstate ← ContractStates.midstate;

function $proof2(uint[s+1]\_Z)$ public
1 require(msg.sender==$Address_{CS}$);
2 require(now<T_d);
3 require(contractstate==ContractStates.midstate);
4 require(csstate==$CSStates.C_{s_2}$);
5 require(balanceOf[msg.sender]> $deposit_{CS}$);
6 $Z \leftarrow \_Z$;
7 csstate← $CSStates.C_{s_3}$;
8 contractstate← ContractStates.closed;

Batch files auditing: Suppose the auditing task refers to multiple files. Let AF be the set of files to be audited.

(a) This step is the same as individual file auditing (a).

(b) Upon receiving $chal$ from the TPA, for each file $f \in AF$, the CS computes

$K_1 = H_2(F_{fn}||k_1||nonce_1||nonce_2|| \cdots ||nonce_\varphi)$,

$K_2 = H_2(F_{fn}||k_2||nonce_1||nonce_2|| \cdots ||nonce_\varphi)$,

where $nonce_1$, $nonce_2$, …, $nonce_\varphi$ are φ successive random numbers in blocks from height $C_{ht} - \varphi + 1$ to C_{ht}. Then it sets the challenge set $C = \{C_f\}_{f\in AF}$, where $C_f = \{(i_f, v_{i_f})\}_{i\in[1,c]}$ and $i_f = \phi(K_{f1}, i), v_{i_f} = \pi(K_{f2}, i)$. It also chooses at random $r, \rho_r, \rho_1, \ldots, \rho_s \in Z_p^*$ and computes

$$\sigma = \mathrm{h}^r \prod_{f\in AF}\left(\prod_{(i_f,v_{i_f})\in C_f} \sigma_{fi_f}^{v_{i_f}}\right), T = \eta^{\rho_r} \prod_{j=1}^{s} \eta_j^{\rho_j}.$$

It executes the $proof1$ function of PISC, uploads σ and T to the blockchain, and sends the blockheight and transaction id $P1 = \{P1_{ht}, P1_{ix}\}$ to the TPA.

(c) After receiving $P1$, the TPA sends (k, φ) to the CS.

(d) The CS checks if $\Psi = g^k h^{\Psi}$, and abort if it does not hold. It computes $z_r = \rho_r - kr$ and for each $j \in [1, s]$,

$\mu_j = \sum_{f\in AF}(\sum_{(i_f,v_{i_f}\in C_f)} v_{i_f} m_{fi_f,j})$, $z_j = \rho_j - k\mu_j$.

It executes the $proof2$ function of PISC, uploads $z_r, z_1, \cdots, z_s$ to the blockchain, and sends the blockheight and transaction id $P2 = \{P2_{ht}, P2_{ix}\}$ to the TPA.

(e) After obtaining the proof from the blockchain, the TPA verifies the file tag $f_{t_{fn}}$ by checking if

$e(g, \prod_{f\in AF} T_{fn}) = e(spk, \prod_{f\in AF} H_0(t_{fn}))$, and

$$\left(\frac{e(\sigma,g)}{e\left(\prod_{f\in AF}(\prod_{(i_f,v_{i_f})\in chal} H_1(F_{fn}||fi_f)^{v_{i_f}}),v\right)}\right)^k = \frac{T}{\eta^{z_r}\eta_1^{z_1}\eta_2^{z_2}\ldots\eta_s^{z_s}}. \quad (1)$$

and abort if either equation fails to hold. The TPA sends the auditing result to the blockchain via the $verification$ function of CISC and sends the blockheight and transaction id $V = \{V_{ht}, V_{ix}\}$ to the CS.

(5) ***Incentive.*** This phase includes $payCS$ part and $payTPA$ part, which are shown in Table 6. After the task deadline T_d, the TPA and CS would be rewarded or amerced by the smart contract according to the task state. Since the transcript of an auditing process as well as the TPA's auditing result would be uploaded onto the blockchain, if the TPA reports a fake auditing result, although it may get the auditing fee, the information on the chain could serve as the evidence of the TPA's misbehavior, and the DO could resort to the court for juridical help.

Table 6. Time-locked Deposit-Withdraw Contract (TDWC)

Output: Update contract TDWC
enum ContractStates{open,midstate,closed}
ContractStates contractstate←ContractStates.open;

function payCS() public payable
1 require(balanceOf[CS]>= $deposit_{CS}$);
2 require(balanceOf[owner]> fee_{TPA}+fee_{TPA});
3 require(balanceOf[TPA]>= $deposit_{TPA}$);
4 require(msg.sender==$Address_{CS}$);
5 require(now>T_d);
6 require(contractstate==ContractStates.open);
7 **if**((tpastate==T_{s_2}) && (csstate==C_{s_3}))
8 balanceCS← balanceOf[CS];
9 balanceOf[CS]→0;
10 msg.owner.transfer(balanceCS);
11 contractstate←ContractStates.midstate;
12 selfdestruct($Address_{CS}$);
13 **end**
14 **if**((tpastate==T_{s_3}) && (csstate==C_{s_3})) && (audit_result==true)
15 balanceOf[owner]← balanceOf[owner]- fee_{CS};
16 balanceCS←balanceOf[CS]; balanceOf[CS]→0;
17 msg.owner.transfer(fee_{CS}+balanceCS);
18 contractstate←ContractStates.midstate;
19 selfdestruct($Address_{CS}$);
20 **end**
21 **if**(tpastate==$TPAStates.T_{s_1}$)
22 balanceCS← balanceOf[CS]; balanceOf[CS]→0;
23 msg.owner.transfer(balanceCS);
24 contractstate←ContractStates.midstate;
25 selfdestruct($Address_{CS}$);
26 **end**

function withdraw_owner() public payable
1 require(msg.sender==owner);
2 require(now>T_d);
3 require(contractstate==ContractStates.closed);
4 balanceowner=balanceOf[owner];
5 balanceCS=balanceOf[CS];
6 balanceTPA=balanceOf[TPA];

```
7  balanceOf[owner]→0; balanceOf[CS]→0;
   balanceOf[TPA]→0;
8  msg.sender.transfer(balanceowner+balanceTPA
   +balanceCS);
9  selfdestruct(Address_owner);
function payTPA() public payable
1  require(balanceOf[owner]> fee_TPA);
2  require(now>T_d);
3  require(msg.sender==Address_TPA);
4  require(contractstate==ContractStates.midstate);
5  if((csstate==C_s1) && (tpatate==T_s2))
6  balanceTPA←balanceOf[TPA];
7  balanceOf[TPA]→ 0;
8  msg.sender.transfer(balanceTPA);
9 contractstate←ContractStates.closed;
10 selfdestruct(Address_TPA);
11 end
12  if((csstate==C_s2) && (tpatate==T_s2))
13  balanceTPA←balanceOf[TPA];
14  balanceOf[TPA]→ 0;
15  msg.sender.transfer(balanceTPA);
16 contractstate←ContractStates.closed;
17 selfdestruct(Address_TPA);
18 end
19 if((csstate== CSStates.honest_CS) &&
   (tpastate==TPAStates.T_S1)&& (audit_ result==true))
20  balanceOf[CS]← balanceOf[CS]- fee_TPA;
21  balanceTPA← balanceOf[TPA]; balanceTPA→ 0;
22  msg.sender.transfer(fee_TPA+balanceTPA);
23  contractstate←ContractStates.closed;
24  selfdestruct(Address_TPA);
25  end
```

Figure 5 shows the entire interaction process among the DO, TPA, CS and the blockchain.

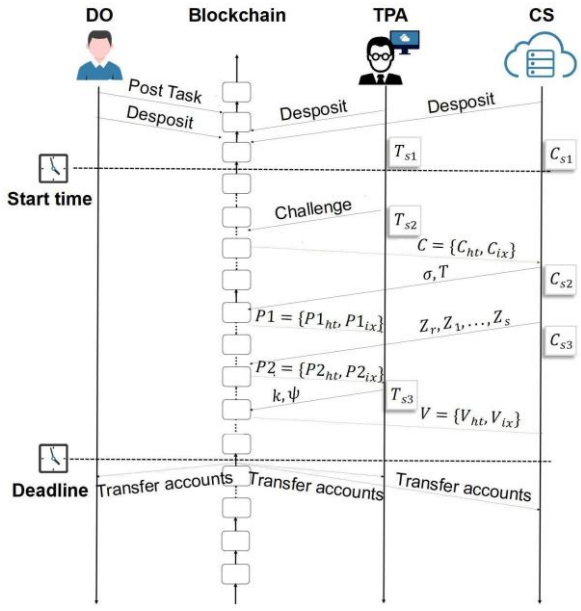


Figure 5. Interaction with the blockchain

6 Security Analysis

6.1 Security Proof of Data Integrity

The correctness of our scheme could be verified in a straightforward way, so we omit it here. Below we show our scheme satisfies data integrity defined in Sect. 4.2.

The theorem below shows that our scheme could protect users' data from being modified without authentication.

Theorem 1: The CS could pass the auditing by the TPA only if it possesses the DO's data intactly.

Proof. The proof is based on Theorem 4.2 of [19]. A challenger $\mathcal{C}$ controls the random oracles $H_0(\cdot)$, $H_1(\cdot)$ and $H_2(\cdot)$ and provides valid responses. The CS is treated as an adversary $\mathcal{A}$ against data integrity. If $\mathcal{A}$ wins with non-negligible probability, we construct a simulator $\mathcal{S}$ playing the role of $\mathcal{C}$, to solve the CDH and DL problems.

Game 0. This game is simply the challenge game defined in Sect. 4,2. $\mathcal{S}$ generates the public parameters, sets $v = g^{\alpha}$ and $u_j = g^{a_j} g^{bb_j}$ where $j \in [1, s], a_j, b_j \in Z_p$, and sends all the public parameters to $\mathcal{A}$. For each challenge block i, $\mathcal{S}$ chooses $r_i \in Z_p$, sets $H_1(F_{fn}||fi) = g^{r_i} / \prod_{j=1}^{s} (g^{a_j m_{fi,j}} \cdot g^{bb_i m_{fi,j}})$, and computes $\sigma_{fi} = (H_1(F_{fn}||fi) \cdot \prod_{j=1}^{s} u_j^{m_{fi,j}})^{\alpha} = g^{\alpha r_i}$. For a challenge from the TPA, suppose that the CS outputs $proof = \{T, \sigma, z_r, z_1, \cdots, z_s\}$ that would be obtained from an honest prover and could pass the verification.

Game 1. It is the same as Game 0, with the exception that the adversary is able to forge part of the proof information. $\mathcal{C}$ records each response generated by $\mathcal{A}$. Let $proof = \{T, \sigma, z_r, z_1, \cdots, z_s\}$ be the expected proof from the CS for a given challenge. $\mathcal{C}$ declares failure and aborts if

(1) the response $proof' = \{T, \sigma, z_r, z_1, \cdots, z'_{\xi}, \cdots, z_s\}$ is valid, and

(2) the response $proof'$ is different from the expected $proof$.

Analysis. By the correctness, the expected proof would make the following equation hold:

$$\left(\frac{e(\sigma, g)}{e\left(\prod_{(f\_n, i\_f, v_{i\_f}) \in chal} H_1(F_{f\_n} || f\_i_{f\_}) ^{v_{i\_f}}, v \right)} \right)^{k} = \frac{T}{\eta^{z_r} \eta_1^{z_1} \eta_2^{z_2} \cdots \eta_s^{z_s}} . \quad \textbf{(2)}$$

Assume that the adversary's output is valid as well and is different from the expected one at position ξ. Then we have

$$\left(\frac{e(\sigma,g)}{e\left(\prod_{\left(i_{f'},v_{i_{f'}}\right)\in chal} H_1(F_{f'n}||f'i_{f'})^{v_{i_{f'}}},v\right)}\right)^k = \frac{T}{\eta^{z_r}\eta_1^{z_1}\eta_2^{z_2}\cdots\eta_\xi^{z'_\xi}\cdots\eta_s^{z_s}}. \quad (3)$$

Dividing Eq. (3) by Eq. (2), we obtain

$$e(u_\xi,v)^{z_\xi} = e(u_\xi,v)^{z'_\xi},$$

and then $u_\xi^{\Delta z_\xi} = (g^{a_\xi} g^{bb_\xi})^{\Delta z_\xi} = 1$, (e.g. $\Delta z_\xi = z'_\xi - z_\xi \neq 0$), which gives a solution to the DL problem, i.e. $g^b = g^{-a_\xi/b_\xi}$, that is, $b = -a_\xi/b_\xi (mod\ p)$. Notice that the probability of game failure is the same as that of $b_\xi = 0(mod\ p)$, which is $1/p$. Therefore, the probability of solving DL problem is $\epsilon_{DL} = (1-1/p)\epsilon_1$, where ϵ_1 is the probability of $\mathcal{A}$ winning in Game 1. If ϵ_1 is non-negligible, so is ϵ_{DL}.

Game 2. It is the same as Game 1, except that $\mathcal{A}$ is able to forge some more part of the proof information. Namely, $\mathcal{C}$ records each response generated by $\mathcal{A}$, declares failure and aborts if

(1) the response $proof'' = \{T, \sigma', z_r, z_1, \cdots, z'_\xi, \cdots, z_s\}$ is valid, and

(2) the response $proof''$ is different from the expected $proof$.

Analysis. For the adversary's output, we have

$$\left(\frac{e(\sigma',g)}{e\left(\prod_{(i_{f'},v_{if'})\in chal} H_1(F_{f'n}||f'i_{f'})^{v_{i_{f'}}},v\right)}\right)^k = \frac{T}{\eta^{z_r}\eta_1^{z_1}\eta_2^{z_2}\cdots\eta_\xi^{z'_\xi}\cdots\eta_s^{z_s}}. \quad (4)$$

Dividing Eq. (4) by Eq. (2), we obtain $e(\sigma/\sigma', g)^k = \eta_\xi^{-(z_\xi - z'_\xi)}$, that is,

$$e\left(\left(\frac{\sigma}{\sigma'}\right)^k, g\right) = e\left(u_\xi, v\right)^{-\left(z_\xi - z'_\xi\right)} = e(g^{-a_\xi \Delta z_\xi a} g^{-bb_\xi \Delta z_\xi a}, g),$$

Hence, we have $g^{ab} = \left(\frac{\sigma'^k}{\sigma^k v^{a_\xi \Delta z_\xi}}\right)^{\frac{1}{b_\xi \Delta z_\xi}}$.

Notice that the probability of game failure is the same as that of $b_\xi \Delta z_\xi = 0(mod\ p)$. Because b_ξ is chosen randomly by $\mathcal{C}$, the probability that $b_\xi \Delta z_\xi = 0(mod\ p)$ is $1/p$. Therefore, the probability of solving CDH problem is $\epsilon_{CDH} = (1-1/p)\epsilon_2$, where ϵ_2 is the probability of $\mathcal{A}$ winning in Game 2. If ϵ_2 is non-negligible, so is ϵ_{CDH}.

Game 3. It is the same as Game 2, except that $\mathcal{A}$ is able to forge any part of the proof information. Namely, $\mathcal{C}$ records each response generated by $\mathcal{A}$, declares failure and aborts if

(1) the response $proof''' = \{T, \sigma', z_r, z'_1, \cdots, z'_\xi, \cdots, z'_s\}$ is valid, and

(2) the response $proof'''$ is different from the expected $proof$.

Analysis. For the adversary's output, we have

$$\left(\frac{e(\sigma',g)}{e\left(\prod_{(i_{f'},v_{if'})\in chal} H_1(F_{f'n}||f'i_{f'})^{v_{i_{f'}}},v\right)}\right)^k = \frac{T}{\eta^{z_r}\eta_1^{z_1}\eta_2^{z'_2}\cdots\eta_\xi^{z'_\xi}\cdots\eta_s^{z'_s}}. \quad (5)$$

Dividing Eq. (5) by Eq. (2), we obtain

$$e\left(\left(\frac{\sigma}{\sigma'}\right)^k, g\right) = e(u_1,v)^{-(z_1 - z'_1)} \cdots e(u_s,v)^{-(z_s - z'_s)} = \prod_{j=1}^s e(u_j,v)^{-(z_j - z'_j)}. \quad (6)$$

For each $j \in [1,s]$, denote by $\Delta z_j = z_j - z'_j$. Since $z_j \neq z'_j$, it holds that $\Delta z_j \neq 0$. From Eq. (6), we know that

$$(\sigma/\sigma')^k = g^{-a\sum_{j=1}^s a_j \Delta z_j} g^{-ab\sum_{j=1}^s b_j \Delta z_j},$$

which gives a solution to the CDH problem, e.g.

$$g^{ab} = \left(\frac{\sigma'^k}{\sigma^k g^{a\sum_{j=1}^s a_j \Delta z_j}}\right)^{\frac{1}{\sum_{j=1}^s b_j \Delta z_j}}.$$

Notice that the probability of game failure is the same as that of $\sum_{j=1}^s b_j \Delta z_j = 0\ (mod\ p)$, which is $1/p$. Therefore, the probability that solving CDH problem is $(1-1/\mathrm{p})\epsilon_3$, where ϵ_3 is the probability of $\mathcal{A}$ winning in Game 3. If ϵ_3 is non-negligible, so is ϵ_{CDH}.

Combining the results above, we have that the CS could only pass the auditing with a negligible probability if the data stored on it is not intact.

This completes the proof of the theorem.

6.2 Analysis of Other Security Properties

Data Privacy: In our scheme, the TPA and CS interactively perform the auditing process by zero-knowledge proof. So there is not any data leakage within the process and TPA could not acquire any information about the DO's data. Therefore, data privacy is achieved in our scheme.

Auditing process publicity and non-modifiability: In our scheme, the auditing process is public and tamper-proof since the whole process is executed and recorded via smart contract in blockchain. Therefore, the scheme is successful to prohibit the TPA or CS from cheating.

Timeliness: Each auditing task starts from time T_s and ends at time T_d in our scheme. TPA conducts the auditing task and returns the auditing result before T_d, otherwise, it will be punished. Therefore, the timeliness of the scheme is achieved.

Incentive: In our scheme, Once the DO launches an auditing task, TPA and CS make their deposit respectively via the *deposit phase* in the TDWC contract. Each entity would be rewarded if it finishes the auditing task timely, so incentive is realized in our scheme.

Batch Auditing: In our scheme, multiple auditing tasks from the same DO can be verified simultaneously, thus the cost of DO is reduced. Therefore, batch auditing is realized in our scheme.

7 Comparison and Efficiency

7.1 Property Comparison

We compare our scheme with some related auditing schemes, e.g. [5, 12-13] in Table 7, in terms of the properties

of privacy preservation, auditing process publicity and non-modifiability, batch auditing, timeliness and incentive. It can be seen that all the schemes support privacy preservation, and all achieve the auditing process publicity and non-modifiability except [5]. Both our scheme and [13] support batch auditing. Only our scheme achieves timeliness and incentive. From the comparison, we know that our scheme has the advantage in resisting the procrastinating TPA and in the auditing process transparency.

Table 7. Property comparison

Property	[5]	[12]	[13]	IPAPA
Privacy preservation	√	√	√	√
Auditing*	×	√	√	√
Batch auditing	×	×	√	√
Timeliness	×	×	×	√
Incentive	×	×	×	√

Auditing*: Auditing process publicity and non-modifiability

7.2 Efficiency Comparison

We analyze the overheads of blockchain, and computation overheads of our scheme and [5, 12-13]. We set the security parameter λ to be of 80 bits, and p to be of $2\lambda = 160$ bits.

Table 8. Storage overhead of blockchain

	Blockchain
[12]	$logc + (2c + 4)logp + (c + 2)160$
[13]	$logc + (2c + 1)logp + (c + 4)320$
IPAPA	$logc + (2c + s + 4)logp + 320$
IPAPA w/batch auditing	$logc + (2cf_0 + s + 4)logp + 320$

Storage overhead on blockchain. We show a comparison of storage overhead on the blockchain including challenge and proof phase in Table 8, where $log\ p$ is the length of p, $log\ c$ is the length of c. In [5], both the TPA and DO need store the challenge information locally. While in other schemes, e.g. [12-13] and IPAPA, it is stored on the blockchain, and the storage overheads are of little difference.

Moreover, the storage overhead in the batch auditing in IPAPA does not increase when compared with the individual file auditing case, because we utilize the aggregation technique. While that of [13] grow linearly with the number of files to be audited. Notice that batch auditing is not supported in [5].

Computation overhead. We show a comparison of computation overhead of the DO and TPA in Table 9, where H_p denotes a hash mapping to Z_p, E denotes an exponentiation in G, E_T denotes an exponentiation in G_T, M_P denotes a multiplication in Z_p, M_G denotes a multiplication in G, M_T denotes a multiplication in G_T, and P denotes a bilinear pairing. Compared with [5], the computation overhead is the same on the TPA side. But it is a little bigger on the CS side. That is because the challenge information set is chosen by TPA itself instead of algorithm in [5], which cannot guarantee the randomness, we provide a random seed for challenge information generation algorithm. While in the batch auditing, the computation overhead grows linearly with the number of auditing files.

7.3 Experimental Evaluation

We used public test blockchain Kovan and Ropsten in Ethereum to demonstrate the efficiency of our scheme. Consensus mechanism of Ropsten and Kovan are PoW and PoA respectively. The smart contract language is Solidity. The transaction fee is estimated as

$Total\ cost = actual\ gas\ used \times gas\ price.$

$1\ ETH = 10^{18} wei = 10^{19} Gwei.$

We published all smart contracts (DLC, CISC, PISC, TDWC) into Kovan and Ropsten on June 7, 2021.

Table 9. Computation overhead

	TPA	CS
[5]	$4P + (2c + 5)E + (2s + 3)E_T + (c + 1)M_G + sM_T$	$(c + 1)E + cE_G + (s + 1)E_T + sM_T + (c + s + 1)M_p$
IPAPA	$4P + (2c + 5)E + (2s + 3)E_T + (c + 1)M_G + sM_T$	$2H_p + 2cM_p + (c + 1)E + cM_G + (s + 1)E_T + sM_T$ $(c + s + 1)M_p$
IPAPA /batch*	$4P + (2f_0c + 5)E + (2s + 3)E_T + f_0(c + 1)M_G +$ $(s + 1)M_p + sM_T$	$2H_p + 2cf_0M_p + (2cf_0 + 1)E + cf_0M_G + (sf_0 +$ $1)E_T + sM_T + (cf_0 + s + 1)M_p$

batch*: batch auditing

Table 10. Gas consumption estimation of smart contracts

Contract	DLC	CISC	PISC	TDWC
PoW (June 7, 2021)		gas price=1 Gwei		
A*	10389493	10389507	10389513	10389523
B*	13	33	13	52
C*	431870	291068	313559	1195525
D*	0.000432	0.000291	0.000314	0.001196
PoA (June 7, 2021)		gas price=5 Gwei		
A*	25321889	25321939	25322000	25322032
B*	2	0	1	2
C*	431870	291068	313559	1195525
D*	0.002159	0.001455	0.001568	0.005978

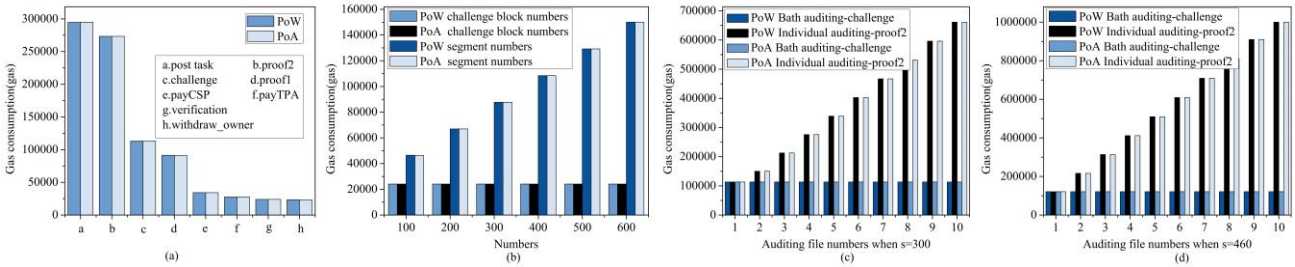
A*: Blockcheight B*: TxIndex C*: Actual gas used (wei)
D*: Total cost (ETH)

Currently, the price configuration in Ropsten is $gas\ price = 1Gwei$ and $gas\ price = 5Gwei$ in Kovan. The costs include two parts: issuing smart contracts and calling functions in each smart contract. The blockheight, transaction id, actual gas used and total cost are showed in Table 10.

In addition, we tested each function transaction fees. Figure 6 shows the gas cost estimation of each smart contract and all the functions. Figure 6(a) is gas consumption of each function into PoW and PoA respectively, there is little difference of the cost under PoW and PoA. Figure 6(b) shows the gas consumption, gas cost does not grow with the increase challenge block numbers, because IPAPA only stores challenge information. While it grows linearly with the increase of segment numbers. Figure 6(c) shows the gas cost

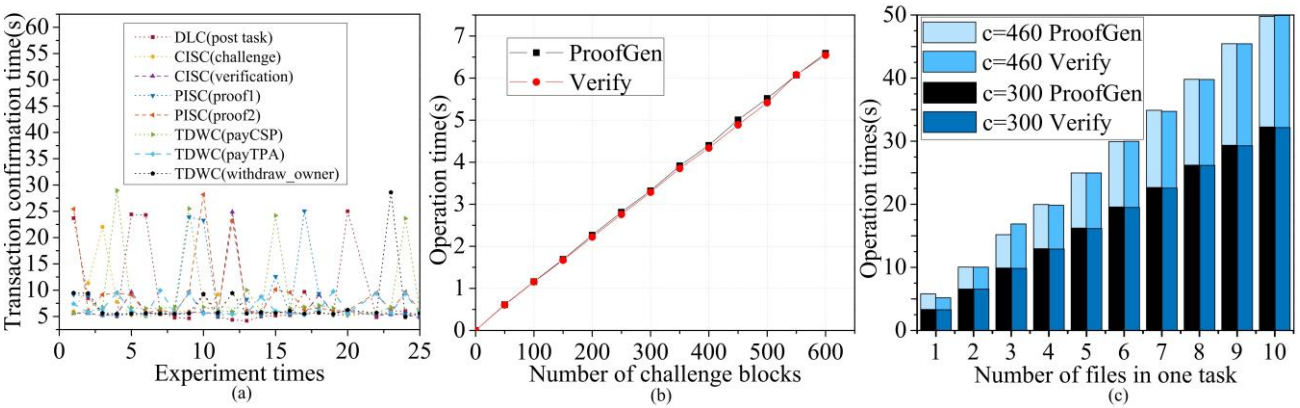
of individual file auditing and batch files auditing. The challenge function cost does not grow with file numbers no matter in individual file auditing or batch files auditing. While the $proof2$ function cost grows linearly in individual file auditing. In Figure 6(d), the gas cost has similar trend.

Furthermore, we implemented our scheme and did the test on a host machine with Windows 10, Intel i7 2.5GHz CPU and 8GB memory. Figure 7 shows the computational efficiency of our scheme. The time complexity of each function is shown in Figure 7(a), which depends on the time needed for the next new block generation and the confirmation numbers by other nodes. In our experiment, we counted the confirmation time after a transaction being confirmed by 500 nodes. From Figure 7(a) we know that the confirmation of each function needs about 5 to 30 seconds. The computation time of executing an auditing task on the TPA side and CS side are shown in Figure 7(b). The computation time grows linearly with the number of challenge blocks. The computation time of batch auditing is shown in Figure 7(c). The time complexity grows linearly with the number of files to be audited.



(a) Gas consumption of each function (b) Gas consumption with different challenge block numbers and segment numbers based on PoW and PoA (c) Gas consumption with batch auditing and individual auditing based on PoW and PoA when s=300 (d) Gas consumption with batch auditing and individual auditing based on PoW and PoA when s=460

Figure 6. Gas cost estimation



(a) The time complexity of each function in smart contract posting into Kovan (b) The operation time with the change of different challenge block numbers (c) The operation time of batch auditing when c=300 and c=460 respectively

Figure 7. Time complexity

8 Conclusion

Aiming to improve the auditing process transparency and resisting the procrastinating TPA and CS, we proposed a public auditing scheme with incentive based on blockchain and smart contracts. Specifically, our scheme utilizes smart contracts to implement a time-locked incentive mechanism to reward or amerce the entities according to the status of an auditing task after the deadline. The TPA would be rewarded if it finishes the auditing task timely and the CS would be rewarded if it maintains the DO's data intactly; otherwise, they would be amerced. Moreover, the auditing transcript and result are uploaded to the blockchain to achieve the auditing process transparency. If there is a dispute, the information stored on the blockchain could serve as a witness. Our scheme achieves privacy preservation property so that an auditing task would not leak information about the DO's data. Our scheme achieves timely auditing and resists the procrastinating TPA and CS, and supports batch auditing so that multiple files could be audited simultaneously without increasing the transaction fees. Experimental results show that our scheme is efficient.

Acknowledgement

This work is supported by the Major Program of Guangdong Basic and Applied Research (2019B030302008), National Natural Science Foundation of China (62272174), and Science and Technology Program of Guangzhou (201902010081).

References

[1] P. Yang, N. Xiong, J. Ren, Data security and privacy protection for cloud storage: A survey, *IEEE Access*, Vol. 8, pp. 131723-131740, July, 2020.

[2] A. Juels, B. S. Kaliski Jr, Pors: Proofs of retrievability for large files, *Proceedings of the 14th ACM conference on Computer and communications security*, Alexandria, Virginia, USA, 2007, pp. 584-597.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, Provable data possession at untrusted stores, *Proceedings of the 14th ACM conference on Computer and communications security*, Alexandria, Virginia, USA, 2007, pp. 598-609.

[4] K. Zhao, D. Sun, G. Ren, Y. Zhang, Public auditing scheme with identity privacy preserving based on certificateless ring signature for wireless body area networks, *IEEE Access*, Vol. 8, pp. 41975-41984, February, 2020.

[5] Y. Yu, Y. Li, M. H. Au, W. Susilo, K.-K. R. Choo, X. Zhang, Public cloud data auditing with practical key update and zero knowledge privacy, *Australasian Conference on Information Security and Privacy*, Melbourne, Australia, 2016, pp. 389-405.

[6] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, G. Min, Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage, *IEEE Transactions on Information Forensics and Security*, Vol. 12, No. 4, pp. 767-778, April, 2017.

[7] H. Tian, F. Nan, C.-C. Chang, Y. Huang, J. Lu, Y. Du, Privacy-preserving public auditing for secure data storage in fog-to-cloud computing, *Journal of Network and Computer Applications*, Vol. 127, pp. 59-69, February, 2019.

[8] L. Chen, S. Zhou, X. Huang, L. Xu, Data dynamics for remote data possession checking in cloud storage, *Computers & Electrical Engineering*, Vol. 39, No. 7, pp. 2413-2424, October, 2013.

[9] H. Tian, Y. Chen, C.-C. Chang, H. Jiang, Y. Huang, Y. Chen, J. Liu, Dynamic-hash-table based public auditing for secure cloud storage, *IEEE Transactions on Services Computing*, Vol. 10, No. 5, pp. 701-714, September-October, 2017.

[10] J. Shen, J. Shen, X. Chen, X. Huang, W. Susilo, An efficient public auditing protocol with novel dynamic structure for cloud data, *IEEE Transactions on Information Forensics and Security*, Vol. 12, No. 10, pp. 2402-2415, October, 2017.

[11] H. Yu, X. Lu, Z. Pan, An authorized public auditing scheme for dynamic big data storage in cloud computing, *IEEE Access*, Vol. 8, pp. 151465-151473, August, 2020.

[12] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, W. Susilo, Blockchain-based fair payment smart contract for public cloud storage auditing, *Information Sciences*, Vol. 519, pp. 348-362, May, 2020.

[13] H. Yuan, X. Chen, J. Wang, J. Yuan, H. Yan, W. Susilo, Blockchain-based public auditing and secure deduplication with fair arbitration, *Information Sciences*, Vol. 541, pp. 409-425, December, 2020.

[14] H. Li, Y. Yang, Y. Dai, S. Yu, Y. Xiang, Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data, *IEEE Transactions on Cloud Computing*, Vol. 8, No. 2, pp. 484–494, April-June, 2020.

[15] Y. Xiang, J. Xu, Y. Si, Z. Li, L. Rasmy, Y. Zhou, F. Tiryaki, F. Li, Y. Zhang, Y. Wu, X. Jiang, W. J. Zheng, D. Zhi, C. Tao, H. Xu, Time-sensitive clinical concept embeddings learned from large electronic health records, *BMC medical informatics and decision making*, Vol. 19, No. Suppl 2, pp. 139-148, April, 2019.

[16] Y. Zhang, C. Xu, X. Lin, X. S. Shen, Blockchain-based public integrity verification for cloud storage against procrastinating auditors, *IEEE Transactions on Cloud Computing*, Vol. 9, No. 3, pp. 923-937, July-September, 2021.

[17] S. Lins, S. Thiebes, S. Schneider, A. Sunyaev, What is really going on at your cloud service provider? creating trustworthy certifications by continuous auditing, *2015 48th Hawaii International Conference on System Sciences. IEEE*, Kauai, HI, USA, 2015, pp. 5352-5361.

[18] J. Li, J. Wu, G. Jiang, T. Srikanthan, Blockchain-based public auditing for big data in cloud storage, *Information Processing & Management*, Vol. 57, No. 6, Article No. 102382, November, 2020.

[19] H. Shacham, B. Waters, Compact proofs of retrievability, *International Conference on the Theory and Application of Cryptology and Information Security*, Melbourne, Australia, 2008, pp. 90-107.

[20] H. Tian, F. Nan, H. Jiang, C.-C. Chang, J. Ning, Y. Huang, Public auditing for shared cloud data with efficient and secure group management, *Information Sciences*, Vol. 472, pp. 107-125, January, 2019.

[21] B. Wang, B. Li, H. Li, Panda: Public auditing for shared data with efficient user revocation in the cloud, *IEEE Transactions on Services Computing*, Vol. 8, No. 1, pp. 92-106, January-February, 2015.

[22] F. Armknecht, J.-M. Bohli, G. O. Karame, Z. Liu, C. A. Reuter, Outsourced proofs of retrievability, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, Scottsdale, Arizona, USA, 2014, pp. 831-843.

[23] Y. Zhang, C. Xu, S. Yu, H. Li, 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, December, 2015.

[24] J. Xue, C. Xu, J. Zhao, J. Ma, Identity-based public auditing for cloud storage systems against malicious auditors via blockchain, *Science China Information Sciences*, Vol. 62, No. 3, pp. 41-56, March, 2019.

[25] Y. Miao, Q. Huang, M. Xiao, H. Li, Decentralized and privacy-preserving public auditing for cloud storage based on blockchain, *IEEE Access*, Vol. 8, pp. 139813-139826, July, 2020.

[26] H. Yu, Z. Yang, R. O. Sinnott, Decentralized big data auditing for smart city environments leveraging blockchain technology, *IEEE Access*, Vol. 7, pp. 6288-6296, December, 2018.

[27] L. Huang, G. Zhang, S. Yu, A. Fu, J. Yearwood, SeShare: Secure cloud data sharing based on blockchain and public auditing, *Concurrency and Computation: Practice and Experience*, Vol. 31, No. 22, Article No. e4359, November, 2019.

[28] R. Kumaresan, I. Bentov, How to use bitcoin to incentivize correct computations, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and*

Communications Security, Scottsdale, Arizona, USA, 2014, pp. 30-41.
[29] X. Yu, M. S. Thang, Y. Li, R. H. Deng, Collusion attacks and fair time-locked deposits for fast-payment transactions in bitcoin, *Journal of Computer Security*, Vol. 27, No. 3, pp. 375-403, June, 2019.
[30] X. Yang, M. Wang, X. Wang, G. Chen, C. Wang, Stateless cloud auditing scheme for non-manager dynamic group data with privacy preservation, *IEEE Access*, Vol. 8, pp. 212888-212903, November, 2020.
[31] J. Mao, Y. Zhang, P. Li, T. Li, Q. Wu, J. Liu, A position-aware merkle tree for dynamic cloud data integrity verification, *Soft Computing*, Vol. 21, No. 8, pp. 2151-2164, April, 2017.
[32] L. Rao, H. Zhang, T. Tu, Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated merkle hash tree, *IEEE Transactions on Services Computing*, Vol. 13, No. 3, pp. 451-463, May-June, 2020.
[33] X. Zhang, J. Zhao, C. Xu, H. Li, H. Wang, Y. Zhang, CIPPPA: Conditional Identity Privacy-Preserving Public Auditing for Cloud-Based WBANs Against Malicious Auditors, *IEEE Transactions on Cloud Computing*, Vol. 9, No. 4, pp. 1362-1375, October-December, 2021.
[34] A. Yang, J. Xu, J. Weng, J. Zhou, D. S. Wong, Lightweight and Privacy-Preserving Delegatable Proofs of Storage with Data Dynamics in Cloud Storage, *IEEE Transactions on Cloud Computing*, Vol. 9, No. 1, pp. 212-225, January-March, 2021.

Biographies

Ying Miao received her M.S. degree from South China Agricultural University. She is currently working as a research assistant at College of Mathematics and Informatics, South China Agricultural University. Her research interests include data security and blockchain.

Qiong Huang received his Ph.D. degree from City University of Hong Kong. Now he is a professor at College of Mathematics and Informatics, South China Agricultural University, China. His research interests include cryptography and information security, in particular, cryptographic protocols design and analysis.

Meiyan Xiao received her PhD degree from South China Agricultural University, and now is a lecturer at College of Mathematics and Informatics, South China Agricultural University, China. Her research interests include data security and blockchain.

Willy Susilo (IEEE Fellow) received the Ph.D. degree in computer science from the University of Wollongong, Australia. He is a Professor and the Head of School of Computing and Information Technology and the Director of Institute of Cybersecurity and Cryptology with the University of Wollongong. He has authored or co-authored over 300 research papers in the area of cybersecurity and cryptology. His main research interests include cybersecurity, cryptography, and information security.