Information Sciences 519 (2020) 348–362

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Blockchain-based fair payment smart contract for public cloud storage auditing

Hao Wang[a,b,*], Hong Qin[a], Minghao Zhao[c], Xiaochao Wei[a], Hua Shen[d], Willy Susilo[b]

[a] School of Information Science and Engineering, Shandong Normal University, China
[b] School of Computing and Information Technology, University of Wollongong, Australia
[c] School of Software, Tsinghua University, China
[d] School of Computers, Hubei University of Technology, China

ARTICLE INFO

Article history:
Received 28 August 2019
Revised 27 January 2020
Accepted 29 January 2020
Available online 30 January 2020

Keywords:
Blockchain
Smart contract
Public auditing
Fair payment
Cloud storage

ABSTRACT

Cloud storage plays an important role in today's cloud ecosystem. Increasingly clients tend to outsource their data to the cloud. In spite of its copious advantages, integrity has always been a significant issue. The audit method is commonly used to ensure integrity in cloud scenarios. However, traditional auditing schemes expect a third-party auditor (TPA), which is not always available in the real world. Also, the former scheme implies a limited pay-as-you-go service, as it requires the client to pay for the service in advance.

In this paper, we aim to address the aforementioned drawback by adopting blockchain to replace TPA and designing a blockchain-based fair payment smart contract for public cloud storage auditing. In our system, data owner and cloud service provider (CSP) will run a blockchain-based smart contract. The contract ensures that the CSP is required to submit data possession proof regularly. The CSP gets paid only if the verification is passed; otherwise, it gets no remuneration but has to pay the penalties. To reduce the number of interactions in the execution of contract, we present the notion of non-interactive public provable data possession and design a blockchain-based smart contract for public cloud storage auditing based on this primitive.

1. Introduction

In cloud storage services, the cloud service provider (CSP) offers the clients with on-demand storage services, either in the form of IssS (*e.g.,* Amazon AWS S3 and Google Cloud Storage) or SaaS (*e.g.,* iCloud and Dropbox). Until now, many pieces of research have been devoted to efficiency, reliability, and user-friendliness aspects of cloud storage services (*e.g.,* [37,42,44]). Recently, cloud storage service has become a big industry, and it is estimated that by 2022, the size of the cloud storage market will grow from $23.48 billion in 2016 to $88.91 billion [1]. Also, with the rapid prevalence of Internet-of-Things (IoT) Devices, growth on the diversity of computation-intensive service (*e.g.,* serverless computing and machine learning services), and the increasing need for enterprise mobility, the popularity trend of cloud storage is still thriving.

* Corresponding author at: School of Information Science and Engineering, Shandong Normal University, No. 88 East Wenhua Road, Jinan 250014, China.
E-mail address: wanghao@sdnu.edu.cn (H. Wang).

https://doi.org/10.1016/j.ins.2020.01.051
0020-0255/

In spite of its numerous advantages, security, reliability and privacy of cloud storage has always been a severe issue [16,17,43]. For the cloud service providers, their storage data centers are normally built as distributed systems with commodity hardware. Accordingly, they are susceptible to both independent and correlated failures [20,22]. Although many hardware (*e.g.,* RAID facilities and ECC Memory) and software (*e.g.,* replication and erasure codes) techniques have been used to prevent data corruption and ensure security and reliability, data corruption accidents still happen occasionally (*e.g.,* [5–8]). In this case, the clients hope to be ensured that, their data is safe, reliable and unmodified stored on the cloud.

However, traditional integrate insurance methods, such as hash function and signature, cannot be applied to or is not effective in cloud storage scenarios, as these tools require the client to have full copies of data. Accordingly, it is required to construct specific methods to check the data reliability and integrity in cloud storage cases [10,13,23]. Cloud audit protocols, especially the Provable Data Possession (PDP) schemes, have been proposed to fulfill this requirement.

In a typical cloud audit scheme, there exists an auditor (normally referred to as the *third-party auditor, TPA*), which uses a spot-checking technique (instead of accessing the whole data stored on the cloud) to check the data integrity. This is also known as *public auditing*. Recently, public auditing systems have been widely studied [10,25,31,33–35]. A secure public-auditing system should be able to resist a forge attack, replacing attack, and replay attack from CSP. Moreover, an ideal public-auditing scheme should also take some desirable proprieties, such as privacy preservation, auditing of dynamic data, batch auditing, auditing of multiple replicas, auditing of shared data and lightweight overheads [32].

However, a suitable TPA may not always exist. In addition, in the use of a cloud storage system, data owners must pay for the cloud storage service in advance. Accordingly, once the data is lost or damaged by CSP, it is hard for data owners to protect their rights. Although the TPA can provide relevant evidence, data owners have to use legal means to defend their rights, which actually requires high additional cost. What is more worrying is that the current law system involving cloud computing is not yet sound, and some related legal issues are difficult to define. Especially, for a cloud storage provider, his storage servers are deployed and distributed around the world. This raises a lot of questions of legal governance over the data. In case a conflict arises between the cloud vendor and the customer, which country's legal system will settle the dispute is still an unresolved question. To resolve the aforementioned problem, we replace the traditional TPA with a smart contract, which is an executable code that runs on the blockchain. Using this technology, we design a fair payment smart contract for the cloud storage system. When data is lost or damaged, the user will no longer have to pay the rent of the cloud service, and will be compensated automatically.

1.1. Our contribution

Taking advantage of decentralization and automatical triggering of the blockchain, we design a blockchain-based fair payment smart contract for public cloud storage auditing. In our system, data owners and CSP will run a smart contract based on blockchain. The contract ensures that the CSP is required to submit data possession proof regularly. Only if the verification is passed, the CSP will be paid, otherwise the CSP will not only receive no remuneration, but will pay the penalties.

When using the traditional public auditing protocol, the verifier needs to interact with the CSP. In this process, a verifier usually generates a random challenge and CSP returns a data possession proof based on this challenge. This kind of interactive proof is not suitable for executing on a smart contract platform, because each consensus node (as a verifier) has to interact with the CSP, the complexity of system communications and the computing cost of CSP will be unacceptable. In order to avoid the interaction between smart contract platforms and CSP in the execution of a contract, we present the notion of non-interactive public provable data possession (NI-PPDP) and design a blockchain-based fair payment smart contract for cloud storage based on this primitive. Concretely, we construct an efficient NI-PPDP scheme by non-trivially extending the Wang et al.'s interactive public auditing scheme [33]. Specifically, the contributions of this paper mainly includes the following three aspects:

- Present the notion of non-interactive public provable data possession (NI-PPDP);
- Construct an efficient NI-PPDP scheme, and give formal proof in the random oracle model;
- Design a blockchain-based fair payment smart contract for cloud storage based on NI-PPDP.

1.2. Organization

We introduce the background and preliminaries in Section 2, and give the formal definition of non-interactive public provable data possession (NI-PPDP) in Section 3. Then, we describe the designs of blockchain-based fair payment contract for cloud storage in Section 4. In Section 5, we present a specific NI-PPDP scheme, and analyze that our NI-PPDP scheme meets all the design goals in Section 6. In Section 7, we analyse the performance of our NI-PPDP scheme. Finally, we give a conclusion in Section 8.

1.3. Related work

1.3.1. Provable data possession

Data owners will lose the physical control of their data, when they use the cloud storage service. How to ensure the integrity of remote data is the most important security issues. In 2007, Ateniese et al. [10] introduced a notion of provable

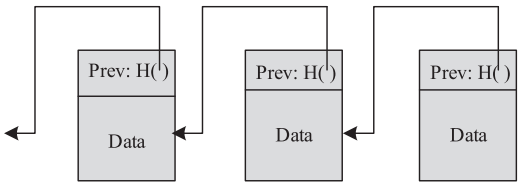

Fig. 1. Blockchain.

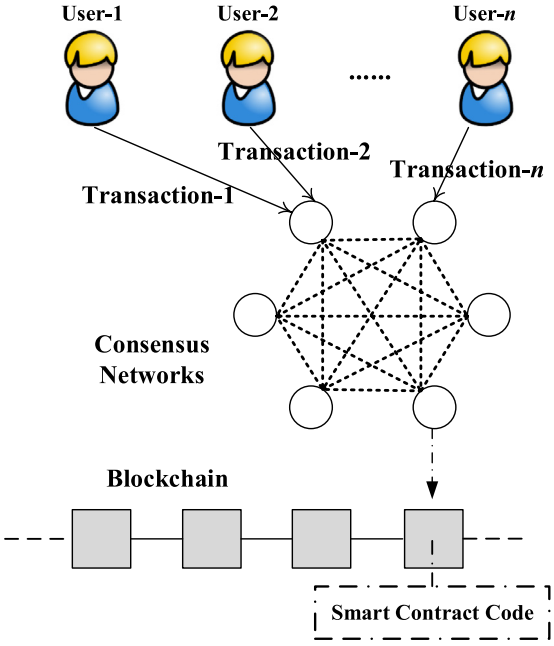

Fig. 2. Smart contract.

data possession (PDP) that allows users to check the remote data without downloading it. Then, Erway et al. [19] proposed the concept of dynamic PDP, which supports data updating. Almost in the same period, Sebé et al. [30] did similar work. In 2008, Shacham and Waters [31] introduced the first fully secure proof-of-retrievability scheme in the Juels-Kaliski model. Since then, a lot of research has been done in this area [14,26,36,39]. However, most of schemes of that period suffer from efficiency problems. In 2015, Liu et al. [27] gave an efficient public auditing scheme based on the Merkle hash tree. This scheme greatly reduces the communication overhead and improves the verification efficiency. Furthermore, considering the case of identity-based cryptosystem and certificateless cryptosystem, Wang [34] proposed an identity-based PDP scheme, and He et al. [24] proposed a certificateless PDP scheme.

1.4. Blockchain and smart contract

In 2008, Satoshi Nakamoto [29] proposed the concept of bitcoin. By combining decentralized consensus technique and append-only data structure, a type of cryptocurrency without the existence of a trusted party is designed. The framework used by bitcoin is generally called *blockchain*. Inspired by bitcoin, some cryptocurrencies and intelligent applications based on blockchain have been proposed one after another, such as Litecoin[3], Zcash[11], Monero[4], Ethereum[2]. These systems have common features: (1) using a consensus protocol to achieve data consensus; (2) using hash chain structure to store data. In general, blockchains are considered to be a linked list of data blocks, where each block is linked by a hash pointer. As shown in Fig. 1, the hash value of the previous block is recorded on the head of the next block. Each block includes a collection of data. Once a block is appended to the blockchain, any change to that block will cause a series of changes in the subsequent blocks. In a blockchain system, the distributed nodes update the hash chain synchronously by running a consensus protocol.

A smart contract is an executable code that runs automatically on the blockchain by consensus nodes without any trusted third party. A smart contract can perform specified operations once pre-defined rules have been met [12]. For instance, using a smart contract, Bob cloud receive x currency units from Alice, if he sends correct calculation results to Alice.

In the smart contract system (Figure 2), each contract has a unique address and cannot be changed after being deployed into the blockchain. When users execute a contract, they only need to send the transaction to the address stated on that

contract. Then, every active consensus node will execute this transaction in the smart contract system to get a consensual result. At present, researchers are trying to use smart contracts to solve various problems in a variety of areas, such as Insurance [21], Medical Care [38], e-Voting [28], Cloud Computing [18] and IoT [15].

Recently, Zhang et al. [40,41] studied the fair payment issues for cloud storage based on blockchain. In their works, PDP is still implemented in a traditional challenge-response way between users and servers. They used bitcoin-based timed commitment technology [9] to achieve fair payment. Our work uses a different approach. After deploying smart contracts, there is no need for challenge-response interaction among users, CSP and any smart contract platform in our system. This will facilitate the implementation of smart contracts by consensus nodes in the public blockchain.

2. Preliminaries

2.1. Bilinear pairings

Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups with prime order p, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a function from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$. $\mathbb{G}$ and $\mathbb{G}_T$ are called bilinear groups, if

- (Bilinear) $\forall g_1, g_2 \in \mathbb{G}, x, y \in Z_p,\ e(g_1{}^x, g_2{}^y) = e(g_1, g_2)^{xy}$.
- (Non-degenerate) $\exists h_1, h_2 \in \mathbb{G}_1,\ e(h_1, h_2)$ is a generator of $\mathbb{G}_T$.

Furthermore, all group operations and function e should be computable.

2.2. Computational Diffie-Hellman (CDH) assumption

Definition 1. Suppose $\mathbb{G}$ is a q-order cyclic group, g is a random generator of $\mathbb{G}$. For $\forall x, y \in \{0, \ldots, q-1\}$, given (g, g^x, g^y), it is computationally intractable to compute g^{xy}.

2.3. Review Wang et al's interactive public auditing scheme [33]

We only review the basic construction of Wang et al's scheme. For a description of the system model and security model, please refer to [33]. In their scheme, there are two phases, (1) Setup and (2) Audit:

- Setup Phase:
  KeyGen $(1^\lambda) \to pk, sk$: Let g be a generator of bilinear group $\mathbb{G}$. The data owner chooses two hash functions, $H(\cdot)$: $\{0,1\}^* \to \mathbb{G}$, $h(\cdot) : \mathbb{G}_T \to \mathbb{Z}_p$, and generates public/private key pairs (spk, ssk) for a digital signature algorithm. Then, it chooses $x \leftarrow \mathbb{Z}_p$, $u \leftarrow \mathbb{G}$ randomly, and calculates $v \leftarrow g^x$. The secret key is $sk = (ssk, x)$ and the public key is $pk = (spk, g, u, v, e(u, v), H(\cdot), h(\cdot))$.
  TagGen $(F, pk, sk) \to \Phi$: We suppose that the data file can be expressed as $F = \{m_i\}_{1 \le i \le n}$, where $m_i \in \mathbb{Z}_p$. The data owner computes authenticators $\sigma_i \leftarrow (H(W_i) \cdot u^{m_i})^x \in \mathbb{G}$ for $i \in [1, n]$, where $W_i = name||i$, and $name \leftarrow \mathbb{Z}_p$ is chosen by the data owner randomly as the identifier of file F. Let $\Psi = \{\sigma_i\}_{1 \le i \le n}$.
  To ensure the correctness of the file identifier $name$, it runs a signing algorithm Sig on the $name$ under ssk, and sets $t = name \,||\, Sig_{ssk}(name)$ as the file identifier for F. The data owner then uploads data file F and corresponding data tags $\Phi = (\Psi, t)$ to CSP.
- Audit Phase:
  The verifier first retrieves the file identifier t, and verifies the signature $Sig_{ssk}(name)$ via spk, and aborts by outputing $\perp$ if verification fails. Otherwise, the verifier recovers the $name$. Then, the verifier picks a random c-element subset $I = \{s_1, \ldots, s_c\}$ of set $[1, n]$. For each index $i \in I$, the verifier chooses $v_j \leftarrow Z_p^*$ randomly. The verifier sends $chal = \{(i, v_i)\}_{i \in I}$ to the server.
  ProofGen $(pk, \Phi, F, chal) \to \Sigma$: It computes

$$\sigma = \prod_{j \in I} \sigma_j^{v_j},$$

and

$$\mu' = \sum_{j \in I} v_j \cdot m_j.$$

The CSP chooses $s \leftarrow \mathbb{Z}_p$ randomly, and calculates $T = e(u, v)^s \in \mathbb{G}_T$. Then, it computes: $\mu = s + \gamma \mu'\ mod\ p$, where $\gamma = h(T) \in Z_p$. It sends $\Sigma = \{\mu, \sigma, T\}$ as the data possession proof to the verifier.
  Verify (pk, Σ): The verifier computes $\gamma = h(T)$ and outputs 1 or 0, according to the correctness of equation below

$$T \cdot e(\sigma^\gamma, g) \stackrel{?}{=} e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^\gamma \cdot u^\mu, v).$$

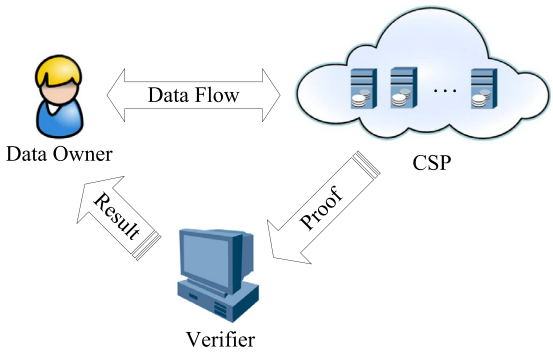

Fig. 3. System model.

3. Non-Interactive public provable data possession scheme

3.1. System model

In a non-interactive public provable data possession (NI-PPDP) scheme, there are three types of entities, that is, data owner, CSP and verifier (as shown in Fig. 3). Different from the traditional interactive public provable data possession scheme, CSP and the verifier do not need to interact during auditing. An NI-PPDP scheme consists of 4 algorithms, which are divided into two phases:

- Setup Phase: In this phase, data owners generate the data tags Φ corresponding to their data file F and store F along with Φ on the cloud storage service. They will run the key generation algorithm and tag generation algorithm as follows.
  KeyGen $(1^{\lambda}) \rightarrow pk, sk$: The key generation algorithm is run by data owner. It takes security parameter λ as input, and outputs public key pk, secret key sk.
  TagGen $(F, pk, sk) \rightarrow \Phi$: The tag generation algorithm is run by the data owner. It takes data file F, public key pk, secret key sk as input, and outputs the corresponding data tags Φ. The data owner then uploads F and Φ to the CSP.
- Audit Phase: In this phase, CSP will prove that it stores complete data. It gives proof based on data tags Φ, data file F and current state τ, by running a proof generation algorithm. Everybody could verify the proof publicly, by running a verifying algorithm. Usually, the verifier is a third-party auditor (TPA), who has a higher computing capability than the data owner, and can check data integrity for data users.
  ProofGen $(pk, \Phi, F, \tau) \rightarrow \Sigma$: The proof generation algorithm is run by CSP. It takes public key pk, data tags Φ, data file F and current state τ as input, and outputs a proof Σ. We suppose the current state τ is some time-varying public information, which cannot be controlled by CSP.
  Verify $(pk, \Sigma) \rightarrow 0, 1$: This is a publicly verifiable algorithm, that can be executed by anyone in this system. It takes public key pk, proof Σ as input and outputs 1 or 0 based on the correctness of Σ.

3.2. Threat model

- We assume that CSP has no incentives to reveal its hosted data to external parties and also has no incentives to drop its hosted data. However, due to some uncontrollable factors, such as, software bugs, hardware failures, bugs in the network path, economically motivated hackers, malicious or accidental management errors, the integrity of users' data might be destroyed. Moreover, for its own benefits, CSP might even decide to hide this data corruption incident to data owners.
- The verifier can verify the integrity of data for data owners according to the proof provided by CSP. However, it may harm the data owners if the verifier could learn related information of the outsourced data from the proof.
- We assume that CSP will not collude with any verifier.

3.3. Design goals

The NI-PPDP scheme should achieve:

- Correctness: For all keypairs $(pk, sk) \leftarrow \textbf{KeyGen}(1^{\lambda})$, for all data files F, and for all states τ, the verification algorithm always outputs

$$1 \leftarrow \textbf{Verify}(pk, \textbf{ProofGen}(\textbf{TagGen}(F, sk), F, \tau)).$$

- Soundness(Data integrity): to ensure that the verification can be passed only if the integrity of data is achieved.
- Privacy preserving: to ensure that auditing process does not disclose any information data.
- Non-interactive: to ensure that the CSP and verifier do not need to interact during auditing.

- Public auditability: to ensure that anyone can verify the integrity of the remote data only depending on the data possession proof given by the cloud storage provider and the public key of data owners.

3.4. Formal security definition

Among the above design goals, data integrity and privacy preserving are the key security features of NI-PPDP. Therefore, we give the formal definition as follows.

3.4.1. Data integrity (soundness)

We use the following game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ to define the soundness of data integrity:

1. $\mathcal{C}$ calls key generation algorithm **KeyGen**(1^{λ}) to generate keypair (pk, sk), and gives pk to $\mathcal{A}$.
2. $\mathcal{A}$ can interact with $\mathcal{C}$ repeatedly and make queries for some file F. Then, $\mathcal{C}$ returns $\Phi \leftarrow$ **TagGen**(F, pk, sk) to $\mathcal{A}$.
3. Finally, $\mathcal{A}$ outputs Σ for some data file F and data tag Φ on state τ.

Define the advantage of $\mathcal{A}$ is $Adv_{\mathcal{A}} = Pr[\textbf{Verify}(pk, \Sigma) = 1]$. We say the adversary wins the above game, if $Adv_{\mathcal{A}}$ is non-negligible.

Definition 2. A non-interactive public provable data possession scheme is sound if exists an efficient extraction algorithm **Extr** such that, for every adversary $\mathcal{A}$, who outputs Σ for some data file F and data tag Φ on state τ and wins above game, the extraction algorithm recovers file F from Φ and Σ, i.e., $\textbf{Extr}(pk, \Phi, \Sigma) = F$.

3.4.2. Privacy preserving

We use the following game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ to define privacy preserving:

1. $\mathcal{C}$ calls key generation algorithm **KeyGen**(1^{λ}) to generate keypair (pk, sk), and gives pk to $\mathcal{A}$.
2. A can interact with $\mathcal{C}$ repeatedly and make queries for some file F. Then, $\mathcal{C}$ returns $\Phi \leftarrow$ **TagGen**(F, pk, sk) to $\mathcal{A}$.
3. At some point, $\mathcal{A}$ submits two files F_0^* and F_1^* to $\mathcal{C}$. Then, C selects $b \in \{0, 1\}$ randomly, and returns **ProofGen**$(\textbf{TagGen}(F_b^*, sk), F_b^*, \tau)$ to A.
4. Finally, $\mathcal{A}$ outputs a guess bit b'.

Defining the advantage of $\mathcal{A}$ as $Adv_{\mathcal{A}} = Pr[b' = b] - 1/2$. We say the adversary wins the above game, if $Adv_{\mathcal{A}}$ is non-negligible.

Definition 3. A non-interactive public provable data possession scheme is privacy preserved if for any probability polynomial time adversary $\mathcal{A}$, advantage of $\mathcal{A}$ in above game is negligible.

4. Blockchain-based fair payment smart contract for cloud storage

In a traditional cloud storage system, data owners must pay the rent before using cloud storage. Once the data is lost or damaged by cloud storage service provider, it is hard for data owners to restore economic losses. To solve this problem, we introduce a novel cloud storage payment model, in which the data owners will pay the fees according to the service quality after enjoying the service. In order to protect the rights of both the data owners and cloud storage service provider, we use a blockchain-based smart contract platform and non-interactive public provable data possession scheme in this system.

As shown in Fig. 4, the data owner runs key generation algorithms and tag generation algorithms of the NI-PPDP scheme, uploads file F and the data tags Φ to the CSP. At the same time, it submits the contract T_0 (Fig. 5) to the smart contract platform. T_0 includes file name, file size, file hash, upload time, storage period, service charges, data owner account (cryptocurrency), cloud storage account (cryptocurrency), data owner's public key, data owner's signature and code of smart contract. This contract ensures that if cloud storage service provider could submit correct data possession proof (using NI-PPDP scheme) on time, the data owner will pay the service fees on time.

After receiving the file F and data tags Φ, the cloud storage provider will check the integrity of data, and the authentication of source. If all verification checks are passed, cloud storage server will submit the contract T_1 (Fig. 6) to the smart contract platform. T_1 confirms that file F has been received by CSP, and ensures that if the data possession proof is not passed, CPS will pay the compensation to the data owner. That is T_1 has two functions: (1) to confirm the receipt of F, (2) to make a promise of compensation. Note that compensation is not necessary, but compensation reflects the reputation of the cloud storage provider.

The specific workflow can be described as:

1. Data owner employs the NI-PPDP scheme, and runs its **TagGen** algorithms on file F to obtain the corresponding data tags Φ.
2. Date owner uploads the file F and the data tags Φ to CSP.
3. Date owner submits the contract T_0 (Fig. 5) to the smart contract platform.
4. CSP checks the integrity of data and the authentication of source.
5. CSP submits the contract T_1 (Fig. 6) to the smart contract platform.

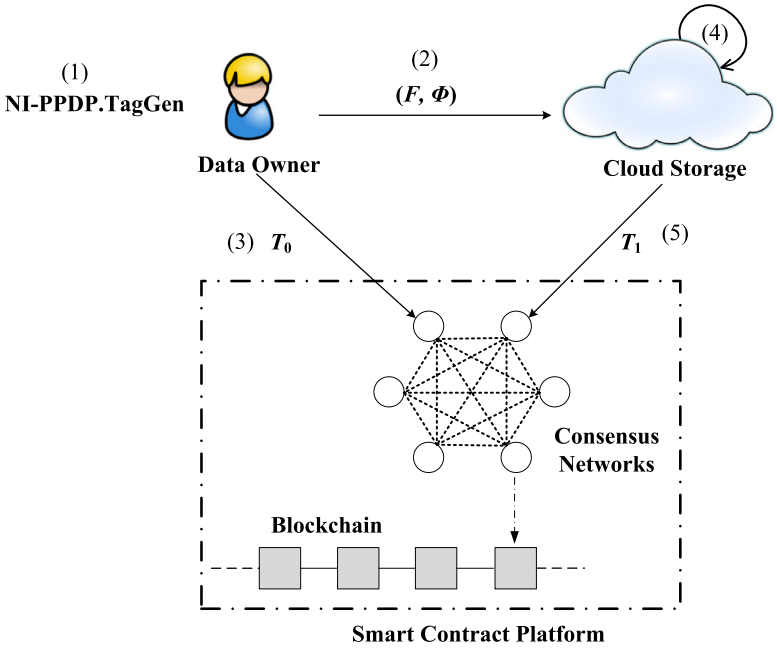

Fig. 4. Data storage process.

Contract T_0
File Name: FN File Size: FS File Hash: FH Upload time: UT Storage Period: CP Service Charges: SC Data owner Account: DOA Cloud Storage Account: CSA Data owner's public key: pk_D Data owner's signature: sig_D

Contract Content:
promise
{
if (NI-PPDP.Verify(pk_D, Σ)==1)
pay SC from DOA to CSA;
}

Fig. 5. Contract T_0.

As shown in Fig. 7, in order to get service charges, the cloud server periodically submits a contract T_2 (Fig. 8), which contains a non-interactive data possession proof Σ. The consensus network will verify this data possession proof in the T_2 and activate T_0 or T_1 based on the validation result. If validation is successful, the T_0 will be activated and the data owner pays the service fees to the cloud server, else the T_1 will be activated, and the CPS will pay compensation to the data owner.

Contract T_1
File Name: FN File Size: FS File Hash: FH Receiving Time: RT Storage Period: CP Penalty: Pen Data owner Account: DOA Cloud Storage Account: CSA CSP's public key: pk_C CSP's signature: sig_C

Contract Content:
confirm F; /*T_0 takes effect*/
promise
{
if (NI-PPDP.Verify(pk_D, Σ)==0)
pay Pen from CSA to DOA;
}

Fig. 6. Contract T_1.

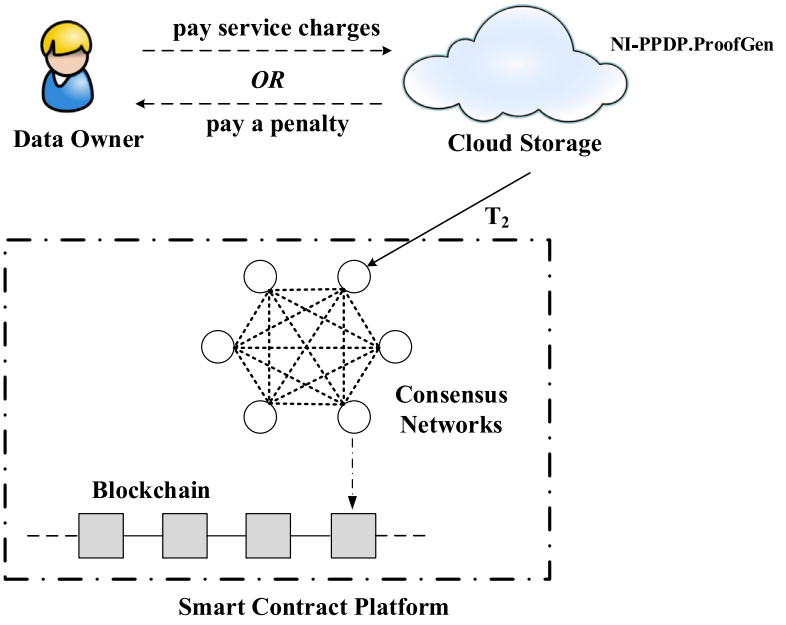

Fig. 7. Data validation process.

5. A specific NI-PPDP scheme

In the following, we present a specific construction of non-interactive public provable data possession scheme. Our construction is achieved by extending interactive public auditing schemes introduced by Shacham and Waters [31] and Wang et al. [33].

Contract T_2
Payment contract: T_0 Compensation contract: T_1 Data possession proof: Σ
Contract Content: activate T_0 or T_1 depended on the validation result of Σ;

Fig. 8. Contract T_2.

5.1. Our construction

Our NI-PPDP scheme is constructed, based on Wang et al.'s public auditing scheme [33]. The main difference is that there is no interaction between the verifier and CSP in the Audit Phase. In order to simulate the challenge process, we use the pseudorandom function on the input of current state. There are also two phases in our scheme.

- Setup Phase:
  KeyGen $(1^\lambda) \rightarrow$ *pk, sk*: Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, select g as a generator of group $\mathbb{G}$, select two cryptographic hash functions $H(\cdot) : \{0, 1\}^* \rightarrow \mathbb{G}$, $h(\cdot) : \mathbb{G}_T \rightarrow \mathbb{Z}_p$, and select pseudorandom function $\mathcal{F}(\cdot)$: $\{0, 1\}^* \rightarrow [1, n]$, which maps arbitrary values uniformly to an integer range $[1, n]$.
  The data owner generates public/private key pairs (*spk, ssk*) for a digital signature algorithm. Then, it chooses $x \leftarrow \mathbb{Z}_p$, $u \leftarrow \mathbb{G}$ randomly, and calculates $v \leftarrow g^x$. The secret key is $sk = (ssk, x)$ and the public key is $pk =$ (*spk, g, u, v, e,* $H(\cdot)$, $h(\cdot)$, $\mathcal{F}(\cdot)$).
  TagGen $(F, pk, sk) \rightarrow \Phi$: Suppose that the data file can be expressed as $F = \{m_i\}_{1 \le i \le n}$, where $m_i \in \mathbb{Z}_p$. The data owner computes authenticators $\sigma_i \leftarrow (H(W_i) \cdot u^{m_i})^x$ for $i \in [1, n]$, where $W_i = name||i$, and $name \leftarrow \mathbb{Z}_p$ is chosen by the data owner randomly as the identifier of file F. Let $\Psi = \{\sigma_i\}_{1 \le i \le n}$.
  To ensure the correctness of the file identifier *name*, it runs signature algorithm *Sig* on *name* under *ssk*, and sets $t =$ *name* $||$ $Sig_{ssk}(name)$ as the file identifier for F. The data owner then uploads the data file F and corresponding data tags $\Phi = (\Psi, t)$ to CSP.
- Audit Phase:
  In this phase, the verifier does not need to choose the challenge set. The CSP uses the current state as the input of pseudorandom function $\mathcal{F}(\cdot)$, to simulate the challenge process.
  ProofGen $(pk, \Phi, F, \tau) \rightarrow \Sigma$: In input, τ represents the current public status information, which contains current time and some other public information and cannot be controlled by CSP. We assume that τ will change in each running of **ProofGen** algorithm. In our blockchain-based fair payment model, τ should also include the header information of the current block of blockchain, which can not be controlled by CSP.
  First of all, CSP choose an appropriate number $c < n$. For $i \in [1, c]$, CSP computes

$$s_i \leftarrow \mathcal{F}(\tau||i).$$

  $I = \{s_1, s_2, \ldots, s_c\}$ is a c-element multiset, $\forall s_i \in [1, n]$. Note, the multiset I is allowed to contain repeated elements.
  For $j \in I$, the CSP computes

$$v_j \leftarrow h(\tau||j).$$

  Then, it computes

$$\sigma = \prod_{j \in I} \sigma_j^{v_j},$$

  and

$$\mu' = \sum_{j \in I} v_j \cdot m_j.$$

  The CSP chooses $s \leftarrow \mathbb{Z}_p$ randomly, and calculates $T = e(u, v)^s \in \mathbb{G}_T$. Then, it computes: $\mu = s + \gamma\mu' \ mod \ p$, where $\gamma = h(T) \in Z_p$. It sends $\Sigma = \{\mu, \sigma, T, \tau, c\}$ as the data possession proof to the verifier.
  Verify (pk, Σ): The verifier runs the verification algorithm of $Ver(spk, id, Sig_{ssk}(name))$ to verify integrity of *id* and verifies the authenticity of state information τ via blockchain. It aborts, if any verification fails. Otherwise, the verifier recovers the *name*. Then, the verifier computes $I = \{\mathcal{F}(\tau||1), \mathcal{F}(\tau||2), \ldots, \mathcal{F}(\tau||c)\}$, $\{v_j = h(\tau||j)\}_{j \in I}$, $\{h(N_j)\}_{j \in I}$, $\gamma = h(T)$ and then outputs 1 or 0, according to the correctness of equation below

$$T \cdot e(\sigma^\gamma, g) \stackrel{?}{=} e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^\gamma \cdot u^\mu, v).$$

5.2. Correctness

$$\begin{aligned}
T \cdot e(\sigma^{\gamma}, g) &= e(u, v)^{s} \cdot e((\prod_{i=s_1}^{s_c}(H(W_i) \cdot u^{m_i})^{x \cdot v_i})^{\gamma}, g) \\
&= e(u^{s}, v) \cdot e((\prod_{i=s_1}^{s_c}(H(W_i)^{v_i} \cdot u^{m_i v_i}))^{\gamma}, g)^{x} \\
&= e(u^{s}, v) \cdot e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu' \gamma}, v) \\
&= e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu' \gamma + s}, v) \\
&= e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu}, v)
\end{aligned}$$

6. Design goals analysis

6.1. Data integrity (soundness)

We use the hybrid argument technique to prove soundness as in [31]. First of all, we define the following games:

Game-0. Game-0 is the original game defined in Section 3.4.1.

Game-1. Game-1 is the same as Game-0, except that the challenger $\mathcal{C}$ records all the tags it signed in a local list. If adversary $\mathcal{A}$ ever submits a tag Φ, that (1) has a valid signature under *ssk* but (2) is not signed by $\mathcal{C}$, then $\mathcal{C}$ announces failure and aborts.

Game-2. Game-2 is the same as Game-1, except that $\mathcal{C}$ records all the responses to **TagGen** queries from $\mathcal{A}$. If $\mathcal{A}$ is successful (i.e., **Verify** output 1) but $\mathcal{A}$'s aggregate signature σ is not equal to $\prod_{j \in I} \sigma_j^{v_j}$, then the challenger $\mathcal{C}$ announces failure and aborts.

Game-3. Game-3 is the same as Game-2, except that challenger $\mathcal{C}$ announces failure and aborts, if at least one of the aggregate messages μ' is not equal to $= \sum_{j \in I} v_j \cdot m_j$.

Lemma 1. *If there is an algorithm $\mathcal{A}$ can distinguish between **Game-0** and **Game-1** with the non-negligible probability, then we can construct an algorithm $\mathcal{B}$ that has a non-negligible advantage to break the existentially unforgeability.*

Analysis. If $\mathcal{A}$ causes $\mathcal{C}$ to abort in Game-1, then we can use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ against the existentially unforgeability of the signature scheme.

Lemma 2. *If there is an algorithm $\mathcal{A}$ can distinguish between **Game-1** and **Game-2** with the non-negligible probability, then we can construct an algorithm $\mathcal{B}$ that has non-negligible advantage to break the computation Diffie-Hellman assumption.*

Analysis. Suppose g^x and g^y are the elements of CDH problem, we set $v = g^x$, $u = g^y$. Suppose $\mathcal{A}$ can respond a signature σ', which is different from the expected signature σ. We can calculate

$$e(\sigma'/\sigma, g) = e(\prod_{j \in I} u^{\Delta \mu_j}, v) = e(g^{\sum_{j \in I} \Delta \mu_j \cdot x \cdot y}, g)$$

Therefore, we can calculate $g^{x \cdot y} = (\sigma'/\sigma)^{\frac{1}{\sum_{j \in I} \Delta \mu_j}}$

Lemma 3. *If there is an algorithm $\mathcal{A}$ that can distinguish between **Game-2** and **Game-3** with the non-negligible probability, then we can construct an algorithm $\mathcal{B}$ that has a non-negligible advantage to break the computation Diffie-Hellman assumption.*

Analysis. We only introduce the main ideas. We suppose that $h(\cdot)$ is a random oracle controlled by an extractor, who answers the hash query asked by the adversary (CSP). For $\gamma = h(T)$ from extractor, the adversary outputs $\{\mu, \sigma, T, t, \tau, c\}$ makes:

$$T \cdot e(\sigma^{\gamma}, g) = e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu}, v).$$

Then, the extractor rewinds $h(T)$ to be $\gamma^* \neq \gamma$. The adversary outputs $\{\mu^*, \sigma, T, t, \tau, c\}$ makes:

$$T \cdot e(\sigma^{\gamma^*}, g) = e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma^*} \cdot u^{\mu^*}, v).$$

Divide above two equations, we have

$$\begin{aligned}
e(\sigma^{\gamma-\gamma^*}, g) &= e(u^{\sum_{j\in I}(h(N_j)v_j)(\gamma-\gamma^*)} \cdot u^{\mu-\mu^*}, v)\\
e(\sigma^{\gamma-\gamma^*}, g) &= e(u^{\sum_{j\in I}(h(N_j)v_j)(\gamma-\gamma^*)} \cdot u^{\mu-\mu^*}, g^x)\\
\sigma^{\gamma-\gamma^*} &= u^{\sum_{j\in I}(h(N_j)v_j)x(\gamma-\gamma^*)} \cdot u^{x(\mu-\mu^*)}\\
(\Pi_{j\in I}\sigma_j^{v_j})^{\gamma-\gamma^*} &= (\Pi_{j\in I}u^{h(N_j)v_jx(\gamma-\gamma^*)} \cdot u^{x(\mu-\mu^*)}\\
u^{x(\mu-\mu^*)} &= (\Pi_{j\in I}(\sigma_j/u^{h(N_j)x})^{v_j})^{\gamma-\gamma^*}\\
u^{x(\mu-\mu^*)} &= (\Pi_{j\in I}(u^{xm_j})^{v_j})^{\gamma-\gamma^*}\\
\mu-\mu^* &= (\sum_{j\in I} m_jv_j)\cdot(\gamma-\gamma^*)\\
\sum_{j\in I} m_jv_j &= (\gamma-\gamma^*)/(\mu-\mu^*)
\end{aligned}$$

Finally, $\{\sigma, \mu' = (\mu-\mu^*)/(\gamma-\gamma^*)\}$ can be treated as a response for the extractor.

Theorem 1. *If the signature scheme is existentially unforgeable and computational Diffie-Hellman assumption holds in bilinear groups, then no probabilistic polynomial time adversary can break the soundness of our NI-PPDP scheme with non-negligible probability.*

Proof. Any adversary's advantage in **Game 3** must be 0 since the challenger always announces failure and aborts if there is no integral file F, i.e. at least one of the aggregate messages μ' is not equal to $= \sum_{j\in I} v_j \cdot m_j$. By the games sequence and Lemmas 1–3, an adversary's advantage in the original game **Game 0** must be negligibly close to 0. □

6.2. Privacy preserving

This theorem shows that the verification process do not reveal any information of the users' data.

Theorem 2. *Our NI-PPDP scheme is privacy preserved.*

Proof. This proof follows from [31,33]. We only introduce the main ideas, i.e. the data possession proof $\Sigma = \{\mu, \sigma, T, t, \tau, c\}$ can not reveal any information about μ'. In the random oracle model, the simulator can construct the response without knowing μ'. It randomly chooses γ, μ from Z_p, and sets

$$T \leftarrow e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu}, v)/e(\sigma^{\gamma}, g).$$

Then, the simulator sets random oracle $h(\cdot)$, makes $\gamma = h(T)$. □

6.3. Non-Interactive

Compared with the traditional interactive public provable data possession scheme, there is no challenge stage in our scheme. CSP does not have to interact with the verifier when it makes proof. To achieve this goal, we use the pseudorandom function, $\mathcal{F}(\cdot)$, to generate the challenge set. Since the input of the $\mathcal{F}(\cdot)$ contains the current state information, this information is related to the current time and the current block chain state, so it cannot be controlled by the CPS. Due to the pseudo randomness, the challenge set generated in this way is indistinguishable to the random choice of verifier.

6.4. Public auditability

It is clear that our scheme has achieved public auditability. The validation algorithm does not depend on any secret inputs.

7. Efficiency analysis

7.1. Theoretical analysis

In the pairing-based cryptography scheme, the computation overhead mainly comes from pairing, as well as exponentiation and multiplication in the group $\mathbb{G}$.

The specific computational analysis is given in Table 1. In the TagGen phase, the main computation overhead comes from the calculation of $\{\sigma_i \leftarrow (H(W_i)\cdot u^{m_i})^x\}_{1\le i\le n}$, which contains c multiplications and n exponentiations on the group $\mathbb{G}$. In the ProofGen phase, the main computation overhead comes from the calculation of $\sigma = \prod_{j\in I}\sigma_j^{v_j}$, which contains c exponentiations and c multiplications on group $\mathbb{G}$. In the Verify phase, the main computation overhead comes from the

Table 1
Computational analysis.

	Multiplication	Exponentiation	Pairing
TagGen	n	$n+1$	0
ProofGen	c	c	0
Verify	$c+1$	$c+3$	2

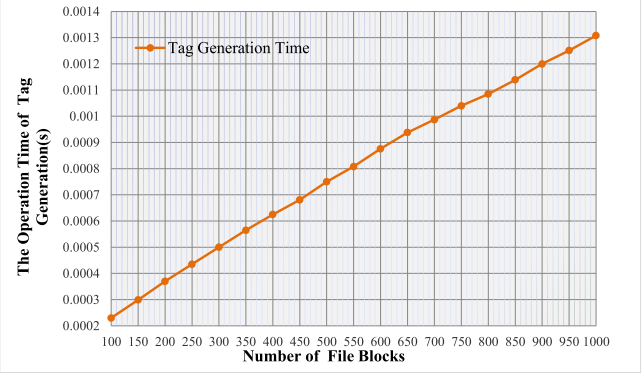

Fig. 9. Operation time for tag generation with file blocks from 100 to 1000.

calculation of $T \cdot e(\sigma^{\gamma}, g) \stackrel{?}{=} e((\prod_{i=s_1}^{s_c} H(W_i)^{v_i})^{\gamma} \cdot u^{\mu}, v)$, which contains $c+1$ multiplications on $\mathbb{Z}_p$, $c+3$ exponentiations on $\mathbb{G}$, and 2 pairings.

7.2. Experimental evacuation

In order to get an in-depth evaluation for the performance of our scheme, we conducted an experiment by implementing a prototype of our scheme with the C Programming language. We adopted the GMP[1] and PBC[2] library for big integer and pairing operation, and adopted OpenSSL[3] for basis cryptographic primitives (e.g., pseudorandom function). We choose the Type-A elliptic curve with order of 160-bit. Both of the programs for the Cloud Service Provider and the Client side are compiled with clang of vision 900.0.39.2, and run on MacBook Pro-with 2.7 GHz Intel Core i5 CPU and 8 GB 1867 MHz DDR3 memory.

In our experiment, we mainly focus on the computational overhead (i.e., The operation time of TagGen, ProofGen and Verify), whereas do not take the Round-Trip Time (RTT) into consideration. This is mainly because the RTT is heavily dependent on the network condition, instead of the proposed scheme and sometimes the RTT may dwarf the operation times, which will make the evaluation unaccurate.Similarly, we also do not account the I/O latency, because the I/O latency is generally determined by type of external storage facilities (e.g, SSD or HDD), the disk scheduling algorithms used in the operating system, data transmission rate and disk interface type. These factors are independent of the proposed scheme, and these uncertainties mentioned above will involve turbulence in our evaluation. Thus, the time overhead of these operations is measured as the time duration between the data is loaded to the memory until the operation is finished.

In terms of TagGen, the client generates the tags for a file, which has been split into several segment (blocks). Figs. 9 and 10 demonstrate the time overhead of TagGen (i.e., the Y-ray) with different number of file blocks. Specifically, in order to observe the variation details, in Fig. 9 we depict the operation time of TagGen as the number of file blocks ranging from 100 to 100 (with the resolution of 100); whereas for the perpose of getting a general overview of the variation tendency, in Fig. 10 we test it with file blocks ranging from 1000 to 10,000 (with resolution of 1000). It is manifest that the operation time of TagGen grows linearly with the increase of file blocks and the stable linear-incremental tendency not only exists in small-scale file blocks, fine-grained interval settings (i.e., depicted in Fig. 9), but also appears as a general tide (i.e., Fig. 10). As is shown in these figures, when the file consists 10,000 blocks, generating the tags for it costs less than 0.02 s and when the files grow even larger, we can infer that the time overheard for tag generation increase proportionally with block numbers. Thus, if the file size extends to Gigabyte scale (which will result in $\frac{1000\times1024\times1024\times8}{160} = 52,428,800$ blocks), based on the pattern observed above, the Tag Generation can be finished within $52,428,800 \times \frac{0.02}{10000} \approx 2$ min. As this procedure is

[1] The GNU MP Bignum Library, https://gmplib.org/.
[2] PBC Library - Pairing-Based Cryptography, https://crypto.stanford.edu/pbc/.
[3] Cryptography and SSL/TLS Toolkit, https://www.openssl.org/.

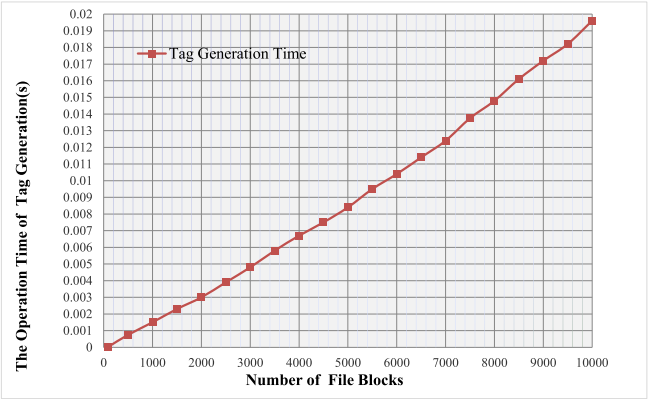

Fig. 10. Operation time for tag generation with file blocks from 1000 to 10000.

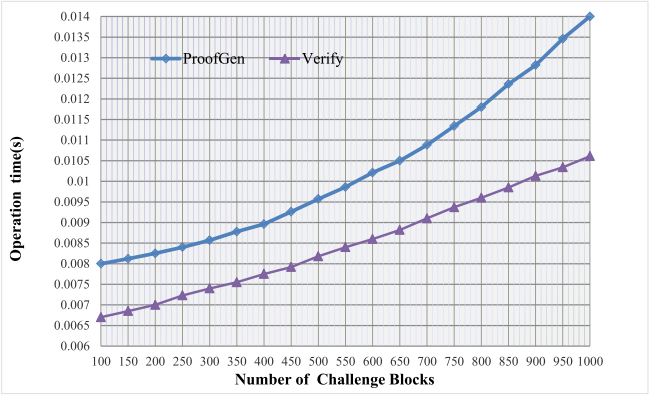

Fig. 11. Operation time for ProofGen and verify.

executed by the client in an offline manner and the file size is seldom as large as the gigabit scale, the overhead for TagGen is acceptable.

The ProofGen is executed by cloud service provider after receiving the challenge request (indicating the number of proof should be generated), and it will generate the proofs of the file under direction of the challenge. In Verify, the client checks the validity of all the proofs generated by the cloud service provider. In our experiment, we measure the operation overhead of operations ProofGen and Verify in the scene of proving and verifying a file with 10,000 blocks. As depicted in Fig. 11, when the challenge blocks increases for 100 to 1000 (i.e., the X-ray), the operation time of ProofGen and Verify also increase accordingly. Specifically the growth rate of ProofGen becomes sharper as the raise of challenge blocks; whereas the Verify, whose operation overhead is much lower than ProofGen, maintains a rigorous linear growth trend. As shown in Fig. 11, the overhead of ProofGen and Verify is approximately 14 ms and 11 ms respectively, when the number challenge blocks reaches 1000. Thus, our scheme is capable for most of the real world settings.

8. Conclusion

In this paper, we design a novel blockchain-based fair payment smart contract for cloud storage. The contract ensures that the CSP is required to submit data possession proof regularly. Only if the verification is passed, the CSP will be paid, otherwise the CSP will not only receive no remuneration, but also will be responsible to pay the penalties. In order to avoid the interaction between smart contract platform and CSP in the execution of contract, we introduce the non-interactive public provable data possession scheme and design an efficient construction.

Declaration of Competing Interest

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

CRediT authorship contribution statement

Hao Wang: Conceptualization, Methodology, Formal analysis, Writing - review & editing, Funding acquisition. **Hong Qin:** Investigation, Validation, Writing - original draft. **Minghao Zhao:** Software. **Xiaochao Wei:** Formal analysis. **Hua Shen:** Visualization. **Willy Susilo:** Supervision.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (Nos. 61602287, 61802235, 61672330, and 61702168), the Primary Research & Development Plan of Shandong Province (No. 2018GGX101037), the Major Scientific and Technological Innovation Project of Shandong Province (No. 2018CXGC0702), and the Development and Construction Funds Project of National Independent Innovation Demonstration Zone in Shandong Peninsula (No. S190101010001).

References

[1] Cloud storage market report, (https://www.marketsandmarkets.com/Market-Reports/cloud-storage-market-902.html)Accessed July 16, 2019.
[2] Ethereum, (https://ethereum.org/) Accessed July 16, 2019.
[3] Litecoin, (https://litecoin.com)Accessed July 16, 2019.
[4] Monero, (https://monero.org/) Accessed July 16, 2019.
[5] Cloud Storage Often Results in Data Loss, 2011. https://www.businessnewsdaily.com/1543-cloud-data-storage-problems.html.
[6] What is the risk of losing data stored in the cloud?, 2018. https://www.quora.com/What-is-the-risk-of-losing-data-stored-in-the-cloud.
[7] OOPS: Google "loses" your cloud data (sky falling; film at 11), August 20, 2015. https://www.computerworld.com/article/2973600/cloud-computing/google-cloud-loses-data-belgium-itbwcw.html.
[8] What happens when data gets lost from the cloud?, January 26, 2015. https://www.cloudcomputing-news.net/news/2015/jan/26/what-happens-when-data-gets-lost-cloud/.
[9] M. Andrychowicz, S. Dziembowski, D. Malinowski, L. Mazurek, Secure multiparty computations on bitcoin, in: 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18–21, 2014, 2014, pp. 443–458, doi:10.1109/SP.2014.35.
[10] G. Ateniese, R.C. Burns, R. Curtmola, J. Herring, L. Kissner, Z.N.J. Peterson, D.X. Song, Provable data possession at untrusted stores, in: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28–31, 2007, 2007, pp. 598–609, doi:10.1145/1315245.1315318.
[11] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, M. Virza, Zerocash: decentralized anonymous payments from bitcoin, in: 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18–21, 2014, 2014, pp. 459–474, doi:10.1109/SP.2014.36.
[12] V. Buterin, A next-generation smart contract and decentralized application platform (2014).
[13] D. Cash, A. Küpçü, D. Wichs, Dynamic proofs of retrievability via oblivious ram, J. Cryptol. 30 (1) (2017) 22–57.
[14] E. Chang, J. Xu, Remote integrity check with dishonest storage server, in: Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6–8, 2008. Proceedings, 2008, pp. 223–237, doi:10.1007/978-3-540-88313-5_15.
[15] K. Christidis, M. Devetsikiotis, Blockchains and smart contracts for the internet of things, IEEE Access 4 (2016) 2292–2303, doi:10.1109/ACCESS.2016.2566339.
[16] C.-K. Chu, W.-T. Zhu, J. Han, J.K. Liu, J. Xu, J. Zhou, Security concerns in popular cloud storage services, IEEE Pervasive Comput. 12 (4) (2013) 50–57.
[17] H. Chung, J. Park, S. Lee, C. Kang, Digital forensic investigation of cloud storage services, Digit. Invest. 9 (2) (2012) 81–95.
[18] C. Dong, Y. Wang, A. Aldweesh, P. McCorry, A. van Moorsel, Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30, - November 03, 2017, 2017, pp. 211–227, doi:10.1145/3133956.3134032.
[19] C.C. Erway, A. Küpçü, C. Papamanthou, R. Tamassia, Dynamic provable data possession, in: Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9–13, 2009, 2009, pp. 213–222, doi:10.1145/1653662.1653688.
[20] D. Ford, F. Labelle, F.I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, S. Quinlan, Availability in globally distributed storage systems, in: Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI), USENIX, 2010, pp. 61–74.
[21] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, V. Santamaria, Blockchain and smart contracts for insurance: is the technology mature enough? Future Internet 10 (2) (2018) 20, doi:10.3390/fi10020020.
[22] P. Gill, N. Jain, N. Nagappan, Understanding network failures in data centers: measurement, analysis, and implications, in: Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), ACM, 2011, pp. 350–361.
[23] J. Han, Y. Li, J. Liu, M. Zhao, An efficient Lucas sequence-based batch auditing scheme for the internet of medical things, IEEE Access 7 (2018) 10077–10092.
[24] D. He, N. Kumar, H. Wang, L. Wang, K.R. Choo, Privacy-preserving certificateless provable data possession scheme for big data storage on cloud, Appl. Math. Comput. 314 (2017) 31–43, doi:10.1016/j.amc.2017.07.008.
[25] A. Juels, B.S.K. Jr., Pors: proofs of retrievability for large files, in: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28–31, 2007, 2007, pp. 584–597, doi:10.1145/1315245.1315317.
[26] J. Li, J. Li, D. Xie, Z. Cai, Secure auditing and deduplicating data in cloud, IEEE Trans. Comput. 65 (8) (2016) 2386–2396, doi:10.1109/TC.2015.2389960.
[27] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, J. Chen, MuR-DPA: top-down levelled multi-replica Merkle hash tree based secure public auditing for dynamic big data storage on cloud, IEEE Trans. Comput. 64 (9) (2015) 2609–2622, doi:10.1109/TC.2014.2375190.
[28] P. McCorry, S.F. Shahandashti, F. Hao, A smart contract for boardroom voting with maximum voter privacy, in: Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3–7, 2017, Revised Selected Papers, 2017, pp. 357–375, doi:10.1007/978-3-319-70972-7_20.
[29] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system(2008).
[30] F. Sebé, J. Domingo-Ferrer, A. Martínez-Ballesté, Y. Deswarte, J. Quisquater, Efficient remote data possession checking in critical information infrastructures, IEEE Trans. Knowl. Data Eng. 20 (8) (2008) 1034–1038, doi:10.1109/TKDE.2007.190647.
[31] H. Shacham, B. Waters, Compact proofs of retrievability, in: Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7–11, 2008. Proceedings, 2008, pp. 90–107, doi:10.1007/978-3-540-89255-7_7.
[32] H. Tian, Y. Chen, H. Jiang, Y. Huang, F. Nan, Y. Chen, Public auditing for trusted cloud storage services, IEEE Secur. Privacy 17 (1) (2019) 10–22, doi:10.1109/MSEC.2018.2875880.
[33] C. Wang, S.S.M. Chow, Q. Wang, K. Ren, W. Lou, Privacy-preserving public auditing for secure cloud storage, IEEE Trans. Comput. 62 (2) (2013) 362–375, doi:10.1109/TC.2011.245.
[34] H. Wang, Identity-based distributed provable data possession in multicloud storage, IEEE Trans. Serv. Comput. 8 (2) (2015) 328–340, doi:10.1109/TSC.2014.1.

[35] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, Enabling public auditability and data dynamics for storage security in cloud computing, IEEE Trans. Parallel Distrib. Syst. 22 (5) (2011) 847–859, doi:10.1109/TPDS.2010.183.
[36] L. Wei, H. Zhu, Z. Cao, W. Jia, A.V. Vasilakos, Seccloud: bridging secure storage and computation in cloud, in: 30th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2010 Workshops), 21–25 June 2010, Genova, Italy, 2010, pp. 52–61, doi:10.1109/ICDCSW.2010.36.
[37] Z. Wu, C. Yu, H.V. Madhyastha, Costlo: cost-effective redundancy for lower latency variance on cloud storage services, in: Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2015, pp. 543–557.
[38] X. Yue, H. Wang, D. Jin, M. Li, W. Jiang, Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control, J. Med. Syst. 40 (10) (2016) 218:1–218:8, doi:10.1007/s10916-016-0574-6.
[39] K. Zeng, Publicly verifiable remote data integrity, in: Information and Communications Security, 10th International Conference, ICICS 2008, Birmingham, UK, October 20–22, 2008, Proceedings, 2008, pp. 419–434, doi:10.1007/978-3-540-88625-9_28.
[40] Y. Zhang, R.H. Deng, X. Liu, D. Zheng, Blockchain based efficient and robust fair payment for outsourcing services in cloud computing, Inf. Sci. 462 (2018) 262–277, doi:10.1016/j.ins.2018.06.018.
[41] Y. Zhang, R.H. Deng, X. Liu, D. Zheng, Outsourcing service fair payment based on blockchain and its applications in cloud computing, IEEE Trans. Serv. Comput. (2018) 1, doi:10.1109/TSC.2018.2864191.
[42] Y. Zhang, C. Dragga, A.C. Arpaci-Dusseau, R.H. Arpaci-Dusseau, Viewbox: integrating local file systems with cloud storage services, in: Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST), 2014, pp. 119–132.
[43] M. Zhao, C. Hu, X. Song, C. Zhao, Towards dependable and trustworthy outsourced computing: a comprehensive survey and tutorial, J. Netw. Comput. Appl. 131 (2019) 55–65.
[44] M. Zhao, Z. Li, E. Zhai, G. Tyson, C. Qian, Z. Li, L. Zhao, H2cloud: maintaining the whole filesystem in an object storage cloud, in: Proceedings of the 47th International Conference on Parallel Processing, ACM, 2018, p. 68.