PoSI: A New Consensus Protocol Based on Storage Age and Data Integrity Verification

Chang Wang, Jun Shen, Shichong Tan

State Key Laboratory of Integrated Service Networks (ISN), Xidian University, China
changw97@163.com, demon_sj@126.com, sctan@mail.xidian.edu.cn

Abstract

The blockchain is getting increasingly popular nowadays because of its decentralization and distributed consistency. The backbone protocol of blockchain is the consensus protocol, making it execute stably. The commonly used consensus protocols in blockchain, such as Proof of Work (PoW), Proof of Stake (PoS) and Proof of Space (PoSpace), have the problem of massive waste of resources. As far as we know, the existing consensus protocols fail to deal with such an issue well. In order to make full use of these resources, we propose a new consensus protocol named Proof of Storage Age and Integrity Verification (PoSI) in this paper. We introduce the definition of storage age and combine it with pairing-free data integrity verification in the proposed protocol. Numerous analyses and implementation results demonstrate the security and high efficiency of the proposed protocol.

Keywords: Consensus protocol, Storage age, Integrity verification, Pairing-free, High efficiency

1 Introduction

With the popularity of digital cryptocurrencies, the underlying blockchain technology has attracted great attention from researchers. The blockchain has the characteristics of decentralization, immutability and transparency, which is considered to be one of the most disruptive and revolutionary technological innovations in recent years. The combination of blockchain technology and the real economy has released great momentum. The application of blockchain has been extended to many areas of society, such as education, medical and logistics. However, there is still a long way to go for the future development of blockchain industry. Only when we make scientific use of it, can we give play to the potential of blockchain.

As we all know, the consensus protocol is the core of blockchain technology, and its existence ensures the stable work of the blockchain. PoW, the consensus protocol used in the most popular Bitcoin system [1], is simple and easy to understand. It requires that a block can be added to the blockchain only if the node successfully solves a hash puzzle. However, it has a serious problem of resource waste brought by numerous hash calculations. To solve this problem, PoS [2] is proposed, which is used in well-known Ethereum [3]. It introduces the concept of the coin age, which is seen as a stake. The hybrid consensus protocol of PoW and PoS in Ethereum determines the difficulty of finding the hash primitive image according to the size of the stake owned by miners, which reduces the use of a small part of computing resources. However, the resources used to hash operations are still wasteful. In PoSpace [4] and Proofs of Space-Time (PoST) [5], storage resources are used to replace computing resources to achieve the purpose of reducing the waste of computing resources. PoSpace is a way to indicate that a person has a legitimate interest in service by allocating a nontrivial amount of storage space. PoST defines a "space-time" resource as a trade-off between CPU work and space-time. It requires participants to submit proof of space-time before they add blocks and receive a mining reward. However, in order to prove the existing space, the nodes in these two protocols need to store a large and meaningless Directed Acyclic Graph (DAG) in space, which reduces the waste of computing resources to a certain extent. However, PoSpace and PoST waste storage resources. Above of these consensus protocols are wasteful of computing resources or storage resources.

To solve the problem of resource waste, Permacoin [6] and Audita [7] try to achieve a consensus by storing valid data and verifying integrity, which is committed to using node resources for integrity verification. Miller *et al*. proposed a modification to Bitcoin in Permacoin that repurposed its mining resources to achieve a more broadly useful goal, distributed storage of archival data. Audita combines integrity verification and block generation in the blockchain network. It builds the network via incentives and solving for issues such as auditing, outsourcing and malicious users. However, Permacoin still needs to execute a large number of hash operations before verifying integrity, which does not fundamentally solve the problem of

*Corresponding Author: Jun Shen; E-mail: demon_sj@126.com
DOI: 10.53106/160792642021092205004

waste of computing resources. In addition, the integrity verification methods used in these two protocols are both based on bilinear pairings, so the efficiency is relatively low. As far as we know, there is no better way to further achieve the efficient use of resources in the blockchain consensus protocol.

1.1 Our Contribution

In this paper, we aim to design a new efficient consensus protocol based on storage age and data integrity verification in order to realize the efficient use of resources. Our contributions are summarized as follows.

We propose the definition of storage age, introducing the difference of decision rights between nodes, which is combined with Practical Byzantine Fault Tolerant (PBFT) to realize the weighted voting mechanism.

We present a new consensus protocol based on storage age and data integrity verification, which enables the resources to be reused efficiently. Note that, the pairing-free data integrity verification employed here is lightweight and efficient.

We demonstrate the security and high efficiency of our PoSI protocol through security analysis and experimental results. The protocol achieves properties of unforgeability and privacy preservation.

1.2 Related Work

Extensive researches have been conducted on the consensus protocol in blockchain and integrity verification for cloud storage.

(1) The consensus protocol in blockchain. Nakamoto [1] initially proposed the architecture model of the Bitcoin system, and used the PoW consensus protocol for its design. The main idea of PoW is to select a specific node for minting through computational competition, and the new block generated by the node is added to the blockchain, so as to ensure the consistency of distributed accounting of the Bitcoin network. Mechanic [2] proposed PoS for the first time in Bitcointalk, which rules for determining the accounting rights of the next moment are basically the same as those of PoW. The difference between the two is that in PoW, the probability of a miner's success in finding a hash is proportional to its computational force, while in PoS, the probability is proportional to its stake. However, there are many defects in PoS. For example, participants with stakes have no intention to participate in the competition of accounting rights. Therefore, in 2014, Larimer *et al.* [8] proposed Delegated Proof of Stake (DPoS) on the basis of PoS, which solved the above problems and improved the enthusiasm and responsibility of coin holders to participate in mining. In the same year, Bentov *et al*. [9] designed the Proof of Activity (PoA) consensus protocol in order to reduce the network attacks caused by the increase of transaction returns. In 2015, Dziembowski *et al.* [4] proposed Proof of Space, in which the computing power resources in PoW were replaced with disk space resources, and miners competed for the accounting right by comparing the size of disk space. In 2018, in view of the phenomenon of currency accumulation in PoS, Proof of Stake Velocity (PoSV) was proposed by Ren [10]. PoSV modified the formula for calculating the coin age in PoS by changing the original formula for calculating the coin age about the linear relationship of time into an exponential decay function about time.

(2) Data integrity verification. In outsourced storage scenarios, an important problem is how to verify data integrity [11-17]. In 2007, Ateniese *et al.* [18] proposed Provable Data Possession (PDP) for the first time. In the same year, Juels *et al.* [19] also proposed Proof of Retrievability (PoR) for the first time, which is slightly different from PDP but has a similar purpose in that it examines data stored remotely. In 2008, Shacham and Waters [20] first combined PDP scheme and redundant coding technology. PDP scheme and PoR scheme are two basic schemes to research data integrity verification. In 2017, Shen *et al.* [21] proposed an efficient public integrity verification scheme with global and blockless sampling validation and batch auditing, where data dynamic support is much more effective than in the case of the prior art. In 2018, Han *et al.* [22] proposed a pairing-free integrity verification scheme based on Schnorr signature [23]. However, the scheme has computation errors in the domain and requires a third-party. In 2019, Zhang *et al.* [24] proposed a general construction method of PoR based on Linearly Homomorphic Structure-Preserving Signatures (LHSPS). The unforgeability of LHSPS ensured the authenticity and tractability of the PoR scheme. In 2020, Yu *et al.* [25] designed a more efficient pairing-free scheme based on blockchain. However, this scheme could only be applied to private auditing.

(3) The combination of consensus protocols and integrity verification. Miller *et al.* [6] presented Permacoin in 2014. In Permacoin, a modification to Bitcoin was proposed that repurposing its mining resources achieves a more broadly useful goal: distributed storage of archival data. Permacoin requires clients to invest not just computational resources, but also storage. Francati *et al.* [7] presented Audita in 2019. Permacoin and Audita combined storage, integrity verification and consensus protocol to try to solve the problem of resource waste in the consensus protocol. Unfortunately, these two protocols only solve part of the waste problem and unsatisfying efficiency. In contrast, we explore how to design a consensus protocol combining integrity verification to achieve an efficient and resource reusable protocol.

1.3 Organization

The rest of this paper is organized as follows. Section 2 reviews some of preliminaries. Section 3 describes the formal definitions of PoSI protocol. The proposed consensus protocol based on storage age and data integrity verification is presented in Section 4. Section 5 shows the implementation of PoSI protocol in the blockchain. Finally, the conclusion is drawn in Section 6.

2 Preliminaries

We review some preliminaries used in our protocol, including bilinear pairings and the corresponding PoR protocol.

2.1 Bilinear Pairings

Let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of prime order p, and g is a generator of $\mathbb{G}$. The pairing $e:\mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$ is a bilinear one iff the following conditions are satisfied:

- Bilinear: $e(u^a, v^b)=e(u,v)^{ab}$ always holds for $\forall_{a,b}\in\mathbb{Z}_p$ and $\forall_{u,v}\in\mathbb{G}$;
- Non-degenerate: $e(g,g)\neq 1_{\mathbb{G}_T}$;
- Computable: $e(u,v)$ is efficiently computable for $\forall_{u,v}\in\mathbb{G}$.

The following is an intractable problem in $\mathbb{G}$.

Definition 1 (Discrete Logarithm Problem (DLP)). The *DLP* in $\mathbb{G}$ is described as follows: given a tuple (g, g^x) for any $x\in_R\mathbb{Z}_p$ as input, output x. Define that the *DLP* assumption holds in $\mathbb{G}$ if for any PPT adversary $\mathcal{A}$,

$$\Pr[\mathcal{A}(1^\lambda, g, g^x)\to x]\le \mathrm{negl}(\lambda)$$

holds for arbitrary security parameter λ, where negl(·) is a negligible function.

2.2 The Pairing-Based PoR

The basic PoR protocol based on bilinear pairings [20] consists of six algorithms. Given a file $M=\{m_{ij}\}_{1\le i\le n,1\le j\le s}$ named *name*, the protocol $\mathcal{BP}=<Setup, Kg, St, Chal, Proof, Verify>$ is defined as follows.

- $\mathcal{BP}.\mathrm{Setup}(1^\lambda)\to(param)$: Input the security parameter λ. Output a set of security parameters $param=(\mathbb{G}, p, g, e)$.
- $\mathcal{BP}.\mathrm{Kg}(param)\to(sk, pk)$: Input *param*. Output the secret key $sk=\alpha\leftarrow_R\mathbb{Z}_p$ and the public key $pk=g^\alpha$.
- $\mathcal{BP}.\mathrm{St}(sk, M)\to(M^*)$: Choose s random elements $u_1,...,u_s\leftarrow_R\mathbb{G}$. Compute

$$\sigma_i\leftarrow(H(name\|i)\cdot\prod_{j=1}^{s}u_j^{m_{ij}})^\alpha$$

Output the file

$$M^*=(\{m_{ij}\}_{1\le i\le n,1\le j\le s},\{\sigma_{i\le i\le n}\}).$$

- $\mathcal{BP}.\mathrm{Chal}(param)\to(Q)$: Pick a random c-element subset I. Select a random element $v_i\leftarrow_R\mathbb{Z}_p$ for $\forall_i\in I$. Output the challenge set $Q=\{(i,v_i)\}$.
- $\mathcal{BP}.\mathrm{Proof}(M^*, Q)\to(\sigma,\mu_1,...,\mu_s)$: Compute

$$\mu_j\leftarrow\Sigma_{(i,v_i)\in Q}v_i m_{ij}\in\mathbb{Z}_p \text{ for } 1\le j\le s,$$

$$\sigma\leftarrow\prod_{(i,v_i)\in Q}\sigma_i^{v_i}\in\mathbb{G}.$$

Output the response $\mu_1,...,\mu_s$ and σ.

- $\mathcal{BP}.\mathrm{Verify}(\sigma,\mu_1,\mu_s)\to(0/1)$: Check whether the equation holds or not,

$$e(\sigma,g)\overset{?}{=}e(\prod_{(i,v_i)\in Q}H(name\|i)^{v_i}\cdot\prod_{j=1}^{s}u_j^{\mu_j},v).$$

3 Formal Definitions of PoSI

In this section, we describe some formal definitions of the proposed PoSI protocol.

3.1 Definition of Entities

Definition 2. The PoSI protocol involves four entities: the common user node (*UN*), the storage node (*SN*), the consensus node (*CN*) and the minting node (*MN*).

- *UN* is an entity that wants a file to be stored by other *SN*s.
- *SN* is an entity that has extra space to help *UN* store files.
- *CN* is an entity that verifies integrity and pins block in the blockchain. It may be *UN* or other nodes.
- *MN* is an entity that has passed integrity verification in *SN*s first. It is responsible for the generation of blocks.

We describe the relationships between entities in brief, as shown in Figure 1. *UN* stores his file in *SN*. *SN* provides the proof of integrity to *CN*. *CN* verifies the proof and elects *MN* from *SN*s.

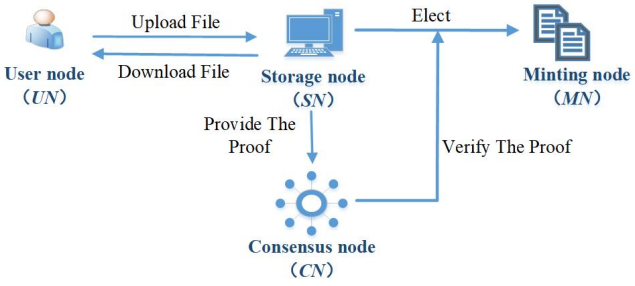

Figure 1. Relationships between entities

3.2 Definition of The Storage Age

Definition 3 (The Storage Age). The storage age is the quantification of how long a file or a part of files has been stored in a fixed location. The size of the storage age depends on the size of storage and storage time. Specifically,

$$StorageAge(GBday) = Content(GB) \times Time(day).$$

For example, if one node stored 3 GB of files for 100 days, its storage age is 300 GBdays. There are three ways in which a node may obtain storage age.

- Getting the most storage age is to store files for *UN* and send proof on time.
- When *SN* contends for the minting right to become *MN*, it may obtain a few storage age and transaction fees by creating blocks.
- Any nodes that join the consensus group and participate in the verification of integrity proof and block pinned may gain a certain storage age.

3.3 Definition of The PoSI Protocol

The formal definition of the PoSI protocol is given as follow:

Definition 4. The PoSI protocol consists of six algorithms defined below.

- $\text{Setup}(1^{\lambda}) \to (param)$. This algorithm is a probabilistic one run by the system. With the security parameter λ as input, it outputs the system parameter *param*.
- $\text{KeyGen}(param) \to (sk_U, pk_U)$. This algorithm is a probabilistic one run by *UN*. Input *param*, the *UN* generates private and public key pair (sk_U, pk_U), where sk_U is kept secret for tag generation and pk_U is public.
- $\text{TagGen}(M, sk_U) \to (\Phi)$. This algorithm is a probabilistic one run by *UN*. Input the file *M* and the private key sk_U, *UN* generates the tag set Φ.
- $\text{Chal}(param) \to Q$. This algorithm is a probabilistic one run by *SN*. Input *param*, *SN* generates the challenge set *Q*.
- $\text{Proof}(Q, M, \Phi) \to proof$. This algorithm is a probabilistic one run by *SN*. Input the challenge set *Q*, the file *M* and the tag set Φ, *SN* generates the proof *proof*.
- $\text{Verify}(proof, Q, pk_U) \to \{0/1\}$. This algorithm is a deterministic one run by *CN*. Input the proof *proof*, the challenge set *Q* and the public key pk_U, *CN* generates verification results.

3.4 Security Goals

The consensus protocol based on storage age and data integrity verification should satisfy properties of validity, unforgeability, privacy preservation and substitution resistance, which are defined as follows.

Definition 5 (Validity). Validity means that all proofs generated by honest *SN* must pass verification, while proofs generated by malicious *SN* cannot be discovered, i.e., for any security parameter λ and negligible function $\text{negl}(\cdot)$,

$$\begin{aligned}&\Pr[\text{Verify}(proof, Q, pk_U) \to 1 : proof \leftarrow \text{proof}\\ &(Q, M, \Phi), Q \leftarrow \text{Chal}(param), \Phi \leftarrow \text{TagGen}\\ &(sk_U, param), (sk_U, pk_U) \leftarrow \text{KeyGen}(param),\\ &param \leftarrow \text{Setup}(1^{\lambda})] = 1.\end{aligned}$$

Definition 6 (Unforgeability). The PoSI protocol is unforgeable if a valid proof is infeasible to be forged when the malicious *SN* doesn't store *UN*'s data completely but a partial file *M'*, i.e., for any security parameter λ and negligible function $\text{negl}(\cdot)$,

$$\begin{aligned}&\Pr[\mathcal{A}^{O_{sign}(sk_U,\cdot)}(pk_U, param) \to (\Phi') \wedge \Phi' \neq \Phi\\ &\wedge \text{Proof}(Q, M', \Phi') \leftarrow \text{Proof}' \wedge \text{Verify}\\ &(proof', Q, pk_U) \to 1 : (sk_U, pk_U) \leftarrow\\ &\text{KeyGen}(param), param \leftarrow \text{Setup}(1^{\lambda})] \leq \text{negl}(\lambda).\end{aligned}$$

where $O_{sign}(\cdot,\cdot)$ is the oracle for signature queries.

Definition 7 (Privacy preservation). The PoSI protocol is with privacy preserving if the following conditions are satisfied:

- If it is infeasible that the curious *SN* obtain any beneficial data when he stores *UN*'s encoded data blocks and the tag set, i.e., for any security parameter λ and negligible function $\text{negl}(\cdot)$,

$$\begin{aligned}&\Pr[\mathcal{A}(M, \Phi) \to (message) : (\Phi) \leftarrow \text{TagGen}\\ &(sk_U, param), (sk_U, pk_U) \leftarrow \text{KeyGen}(param)\\ &(param) \leftarrow \text{Setup}(1^{\lambda})] \leq \text{negl}(\lambda),\\ &\text{KeyGen}(param), param \leftarrow \text{Setup}(1^{\lambda}) \leq \text{negl}(\lambda).\end{aligned}$$

  where *message* is some beneficial data.
- If it is infeasible that the curious *CN* obtain any beneficial data when he receives the proof from *SN*, i.e., for any λ and negligible function $\text{negl}(\cdot)$,

$$\Pr[\mathcal{A}(proof) \to (message) : (proof) \leftarrow \text{proof}$$
$$(Q, M, \Phi), Q \leftarrow \text{Chal}(param), (\Phi) \leftarrow$$
$$\text{TagGen}(sk_U, param), (sk_U, pk_U) \leftarrow \text{KeyGen}$$
$$(param), param \leftarrow \text{Setup}(1^{\lambda})] \le \text{negl}(\lambda).$$

Definition 8 (Substitution resistance). The PoSI protocol is resistible to substitution attacks if it is infeasible that the existing data block m_t is substituted for the lost challenge block m_{i_k} and a valid proof is generated when a part of the challenge blocks in *SN* is lost, i.e., for any security parameter λ and negligible function $\text{negl}(\cdot)$,

$$\Pr[\mathcal{A}^{O_{sign}(sk_U,\cdot)}(pk_U, param) \to (\Phi) \wedge$$
$$m_{i_k} \notin M \wedge \sigma_{i_k} \notin \Phi \wedge m_t \in M \wedge \sigma_t \in \Phi$$
$$\text{Proof}(Q, M, \Phi) \to proof' \wedge \text{Verify}$$
$$(proof', Q, pk_U) \to 1 : (sk_U, pk_U) \leftarrow \text{KeyGen}$$
$$(param), param \leftarrow \text{Setup}(1^{\lambda})] \le \text{negl}(\lambda).$$

4 The Proposed PoSI Protocol

We first overview the proposed PoSI protocol based on storage age and data integrity verification. Then, we give the concrete construction of PoSI protocol. Finally, we analyze the security and performance.

4.1 Overview

The PoSI consensus protocol we proposed is based on storage age and data integrity verification. To improve the verification efficiency, the data integrity verification we employed is pairing-free. The main idea of PoSI is shown in Figure 2.

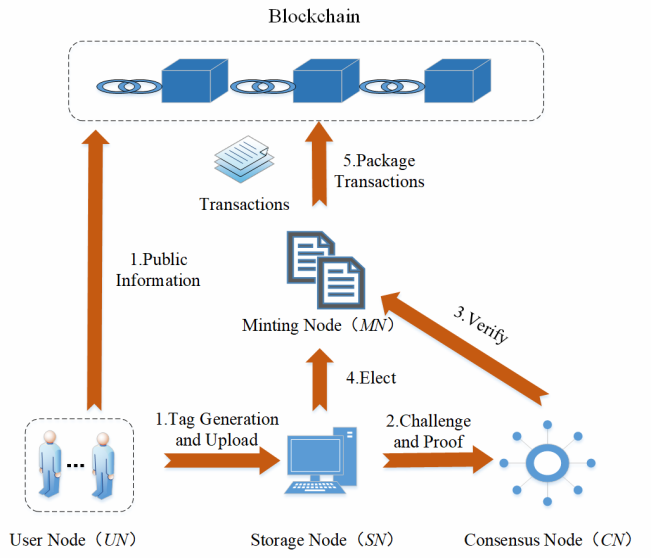

Figure 2. The execution of PoSI protocol

UN preprocesses the file and generates the data file tags using its private key. Then, the file blocks and tags are stored in *SN*. At per period, *SN*s according to the timestamp in the latest block in the blockchain generate challenges and the corresponding proof. *CN* verifies whether the proof is valid or not. If $f+1$ *CN*s verifies the proof is valid, the first verified *SN* becomes *MN*, where f is the maximum number of malicious nodes that the protocol can tolerate. *MN* completes the transactions of the packaging and generates a block in the blockchain. After each period of the consensus protocol, the storage age will be updated. The larger the storage age, the fewer blocks need to be verified in the generated challenge, the faster the proof will be generated, and the easier it is to be selected as *MN*. For an honest *SN*, the storage age is the key to whether it may become *MN*.

4.2 The Concrete Construction of PoSI protocol

The PoSI protocol is based on storage age and pairing-free data integrity verification. Given an encoded file $M = \{m_{ij}\}_{1 \le i \le n, 1 \le j \le s}$, the procedure of the protocol execution is as follows.

- $\text{Setup}(1^{\lambda}) \to (param)$. On input the security parameter λ, the system executes as follows:
  - Choose two large primes p and q, where $q \mid p-1$. $\mathbb{G}$ is a multiplicative cyclic group of order q, and g is generator of $\mathbb{G}$.
  - Choose a pseudorandom function and a secure hash function:

    $$F(\cdot) : \{0,1\}^* \to [1, n], H(\cdot) : \{0,1\}^* \to \mathbb{Z}_q.$$

  - Output a set of system parameters $param = (p, q, g, \mathbb{G}, F(\cdot), H(\cdot))$.
- $\text{KeyGen}(param) \to (sk, pk)$. On input the system parameter *param*, UN generates the key pair as follows:
  - Randomly select a private key $sk_U \in_R \mathbb{Z}_q^*$, and compute the public key $pk_U = g^{sk_U} \mathbb{G}$.
  - Output the *UN*'s public and private key pair (sk_U, pk_U).
- $\text{TagGen}(M, sk_U) \to (\Phi)$. On input the file M and the private key sk_U, UN generates the tag set as follows:
  - Randomly select $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_s)$, $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ where $\alpha_l, \beta_i \in_R \mathbb{Z}_q$ and compute

    $$\rho_i = \sum_{l=1}^{s} \alpha_l m_{il} + \beta_i, \text{ for } i \in [1, n].$$

  - Randomly select $\eta_i \in_R \mathbb{Z}_q^*$, and compute

    $$\gamma_i = g^{\eta_i} \bmod q, s_i = \eta_i + \rho_i \cdot sk_U \bmod q,$$

— Output the tag set $\Phi = \{\sigma_i\}_{1 \le i \le n}$, where $\sigma_i = (\gamma_i, s_i)$.

• Chal($param$) $\rightarrow Q$. On input the system parameter $param$, SN generates the challenge set as follows:

— Get the number of the challenge blocks c according to the storage age.

— Compute $Chal_j = F(t \| j), j \in [1, c]$ and the set $I = \{Chal_1, Chal_2, ..., Chal_c\}$, $\forall Chal_j \in [1, n]$. Compute $v_i = H(t \| i), i \in I$.

— Output the challenge set $Q = \{(i, v_i)\}_{i \in I}$.

• Proof$(Q, M, \Phi) \rightarrow proof$. On input the challenge set Q, the file M and the tag set Φ, SN generates the proof as follows:

— Compute R and S according to the block tag, then compute μ_l and τ_0 according to the file block,

$$R = \prod_{i \in I} \gamma_i^{v_i} \bmod p,$$

$$S = \sum_{i \in I} v_i s_i \bmod q,$$

$$\mu_l = \sum_{i \in I} v_i m_{il} \bmod q,$$

$$\tau_0 = \sum_{l=1}^{s} \alpha_l \mu_l \bmod q.$$

— Output the proof $Proof = (R, S, \tau_0)$.

• Verify($proof, Q, pk_U$) $\rightarrow \{0/1\}$. On input the proof $proof$, the challenge set Q and the public key pk_U, CN generates verification results as follows:

— Compute $\tau = \tau_0 + \sum_{i \in I} v_i \alpha_i \bmod q$.

— Check whether the equation holds or not,

$$g^S \stackrel{?}{=} R \cdot pk_U{}^{\tau} \bmod p \qquad \textbf{(1)}$$

of which the correctness is derived as follows.

$$\begin{aligned} g^S &= \prod_{i \in I} g^{v_i s_i} \bmod p \\ &= \prod_{i \in I} g^{v_i \eta_i + v_i p_i s k_U} \bmod p \\ &= R \cdot \prod_{i \in I} pk_U{}^{\sum_{l=1}^{s} \alpha_l m_{il} v_i + v_i \beta_i} \bmod p \\ &= R \cdot pk_U{}^{\tau} \bmod p \end{aligned}$$

4.3 Security Analysis

We give proofs of the following several theorems to demonstrate the achievement of unforgeability, privacy preservation and substitution resistance in the proposed PoSI protocol.

Theorem 1. *The PoSI protocol is unforgeable under the DLP assumption.*

Proof. By reference to the security game defined in [20], we may demonstrate that if the game, named **Game 1**, is won by *SN*, through using incorrect date to forge a valid proof, then we will discover a solution to work out a *DLP* instance in $\mathbb{G}$ with a high probability of $1 - \frac{1}{q}$, which goes against the difficulty of *DLP* in $\mathbb{G}$. Next, we give the definition of **Game 1**.

Game 1: On getting a challenge with the timestamp of the previous block, *SN* should use the file M to generate a proof *proof* and this proof must make the verification (1) hold. Assuming that *SN* stores the corrupted file M, he will use the improper file $\hat{M}$ to forge a proof $\widehat{proof} = \{(R, S, \widehat{\tau_0})\}$, where $\widehat{\tau_0} = \sum_{l=1}^{s} \alpha_l \widehat{\mu_l} \bmod q$. and $\widehat{\mu_l} = \sum_{i \in I} v_i \widehat{m_{il}} \bmod q$, $l = 1, 2, ..., s$.

If *CN*'s verification is passed by the forged proof, *SN* will win the **Game 1**. Otherwise, he fails.

We define as following:

$$\Delta\mu_l = \mu_l - \widehat{\mu_l} = R \cdot pk_U{}^{\sum_{l=1}^{s} \alpha_l \widehat{\mu_l} + \sum_{i \in I} v_i \beta_i} \bmod p$$

Where $l = 1, 2, ..., s$. Apparently, here is at least one element of $\{\Delta\mu_l\}$ is nonzero. We suppose that *SN* has won the **Game 1**. Equation (2) holds because of the verification equation (1).

$$g^S = R \cdot pk_U{}^{\hat{\tau}} = R \cdot pk_U{}^{\sum_{l=1}^{s} \alpha_l \widehat{\mu_l} + \sum_{i \in I} v_i \beta_i} \bmod p \qquad \textbf{(2)}$$

Equation (3) also holds due to a valid proof *proof*.

$$g^S = R \cdot pk_U{}^{\tau} = R \cdot pk_U{}^{\sum_{l=1}^{s} \alpha_l \widehat{\mu_l} + \sum_{i \in I} v_i \beta_i} \bmod p \qquad \textbf{(3)}$$

In combination with equation (2) and equation (3), we see that

$$pk_U{}^{\sum_{l=1}^{s} \alpha_l \widehat{\mu_l} + \alpha_l \mu_l} = 1 \bmod p, \prod_{l=1}^{s} pk_U{}^{\alpha_l \Delta\mu_l} = 1 \bmod p.$$

Because $\mathbb{G}$ is a cyclic group with order q and $pk_U = g^{sk_U} \in \mathbb{G}$, then we have

$$1 = \prod_{l=1}^{s} pk_U{}^{\alpha_l \Delta\mu_l} = g^{\sum_{l=1}^{s} sk_U \alpha_l \Delta\mu_l} \bmod p.$$

Because $g^q = 1$, we may find a solution to the *DLP* in $\mathbb{G}$.

$$sk_U = q / \sum_{l=1}^{s} \alpha_l \Delta\mu_l \bmod q \qquad \textbf{(4)}$$

The equation (4) holds as long as the denominator is not zero. Nevertheless, according to the definition in **Game 1**, we know that at least one element is nonzero in $\{\mu_l\}, l=1,2,...,s$. Because α_l is a random value of $\mathbb{Z}_q$, the denominator is zero with a probability of $1/q$, which is negligible because q is a big prime. That is, as soon as the malicious *SN* wins **Game 1**, we may discover a solution to work out the *DLP* in $\mathbb{G}$ with a high probability of $1-1/q$, which goes against that *DLP* is computationally infeasible in $\mathbb{G}$ in the security assumption. □

Theorem 2. *The PoSI protocol is privacy preserved.*

Proof. In our protocol, the *UN* makes use of redundant encoding technology to encode the file to ensure the confidentiality, privacy and recoverability of the file *M*. Besides, the generation of the authentication meta set is primarily based on the encoding of the file block. As a result, the curious *SN* cannot attain any beneficial information from the encoded data and authentication element set.

The curious *CN* may additionally desire to derive privacy data when he receives the proof $proof=(R,S,\tau_0)$, where

$$\tau_0=\sum_{l=1}^{s}\alpha_l\mu_l \bmod q,\ \mu_l=\sum_{i\in I}v_i m_{il} \bmod q.$$

However, each data block is redundantly encoded data and only someone with a decoding matrix can decode the data block and get useful information. Therefore, the curious *CN* cannot obtain any secretive information from the proof.

In conclusion, the protocol is able to protect *CN*'s data privacy from attacks by the curious *CN* and *SN*. □

Theorem 3. *The PoSI protocol is able to resist substitution attack.*

Proof. Assume that the number of challenge blocks based on the timestamp of the current block of the blockchain is *c*. The challenge blocks' index is $I=\{i_1,...,i_c\}$ and the challenge block set is $\{m_{i1},...,m_{ic}\}$. Assume file block $m_{i_k}, i_k\in I$ is lost or corrupted, and *SN* replaces m_{i_k} with file blocks $m_t, t\notin I$. At this point, the proof is $\overline{proof}=\{(\overline{R},\overline{S},\overline{\tau_0})\}$, where

$$\overline{R}=\prod_{i\in I/\{i_k\}}\gamma_i^{v_i}\cdot\gamma_t^{v_{i_k}} \bmod p$$

$$\overline{S}=\sum_{i\in I/\{i_k\}}v_i s_i+v_{i_k}s_t \bmod p$$

$$\overline{\mu_l}=\sum_{i\in I/\{i_k\}}v_i m_{il}+v_{i_k}m_{tl} \bmod q$$

$$\overline{\tau_0}=\sum_{l=1}^{s}\alpha_l\overline{\mu_l} \bmod q$$

CN calculates $\overline{\tau}$ according to the corresponding $\beta_i, i\in I$ if the file is not lost or corrupted. We know,

$$\overline{\tau}=\overline{\tau_0}+\sum_{i\in I}v_i\beta_i \bmod q.$$

Assuming that the verification equation (1) holds, then we have

$$g^{\overline{S}}=\overline{R}\cdot pk_U^{\overline{\tau}} \bmod p.$$

Then, we derive that

$$s_t=\eta_t+sk_U(\sum_{l=1}^{s}\alpha_l m_{tl}+\beta_{i_k}) \bmod p.$$

Because $s_t=\eta_t+sk_U(\sum_{l=1}^{s}\alpha_l m_{tl}+\beta_{i_k}) \bmod p$, the proof after the substitution of the block makes the verification equation hold, if and only if $\beta_t=\beta_{i_k}$. The probability of $\beta_t=\beta_{i_k}$ is $1/q$, and since q is a large prime, the probability $1/q$ is negligible. Therefore, if some data blocks stored on the server are corrupted or lost, a malicious *SN* is able to make verification equation hold by forged proof using substitution attack with negligible probability. □

4.4 Performance Evaluation

The numerical analysis and experimental results compared with protocol [20] are present to evaluate the performance of the PoSI protocol. The corresponding notations are listed in Table 1.

Table 1. Notations

Notation	Description
n	The number of data blocks in a file
s	The number of sectors of a data block
p	The prime order of $\mathbb{G}_1$ in [20]
q	The prime order of $\mathbb{G}_2$ in our protocol
c	The number of the challenge blocks
M	The multiplication
E	The modular exponentiation
P	The bilinear operation

The communication comparison with [20] is shown in Table 2. In tag uploading, the communication cost is just linear with the number of data blocks both in [20] and our protocol. In proof responding, the communication cost of our protocol is a const, smaller than that of [20] which is linear with the number of sectors of a data block. In general, the communication cost of our protocol is less than that of protocol [20].

Table 2. Comparison of communications and computations

	Protocols	Protocol [20]	Our protocol
Communications	Tag uploading	$n\|\mathbb{G}_1\|$	$n(\|\mathbb{G}_2\|+\|q\|)$
	Proof responding	$\|\mathbb{G}_1\|+s\|p\|$	$\|\mathbb{G}_2\|+2\|q\|$
Computations	Tag generation	$snE+sM$	$nE+(s+n)M$
	Proof generation	$cE+(cs+c-1)M$	$cE+(3c+s-1)M$
	Proof verification	$2P+(c+s)E+(c+s-1)M$	$2E+(c+1)M$

The computational comparison is presented in Table 2. In tag generation, protocol [20] needs sn modular exponentiations and our protocol only needs n modular exponentiations. Hence, the computational cost of our protocol is smaller than [20]. In proof generation, the computational cost is linear with the number of challenge blocks and it is easy to see that our protocol has a lower computational cost. In proof verification, protocol [20] needs two bilinear operations and $c+s$ modular exponentiations while our protocol only needs two modular exponentiations. Therefore, the computational cost of our protocol is much less than that of [20]. The same conclusion is shown in experimental results.

On Linux operating system, Intel(R) Xeon CPU E5-2682 v4 @ 2.50GHz processor and 2GB of RAM are used to execute the following experiments. The GMP libr-ary of GNU multi-precision algorithms and PBC library are written in Python. We set $|p|=1024\,bits$, $|q|=160\,bits$ and the 1024-bit MODP Group with 160-bit Prime Order Subgroup in literature [26] was used in the Group $\mathbb{G}$. Each experiment was run several times and its average was calculated.

The change of tag generation computational cost of the two protocols with the change of n is shown in Figure 3. The computation cost is linear with the number of data blocks n in tag generation. However, in tag generation, thecomputational cost in our protocol is significantly lower than that in [20]. The change of proof generation and proof verification cost of the two protocols with the change of the number of the challenged data blocks is showed in Figure 4 and Figure 5.

As shown in Figure 4, we know that as the number of challenge blocks increases, so does the cost for proof generation. In our PoSI protocol, such a changing trend may make *SN* calculate the proof in different time, which is able to elect *MN* more effectively.

Figure 5 shows that our protocol takes significantly less time than protocol [20] when verifying proof. As shown in Figure 6, the proof verification in our protocol obviously takes much less time than the proof generation. In particular, our PoSI protocol only takes 2ms~6ms to verify proof, which is very suitable for the blockchain system.

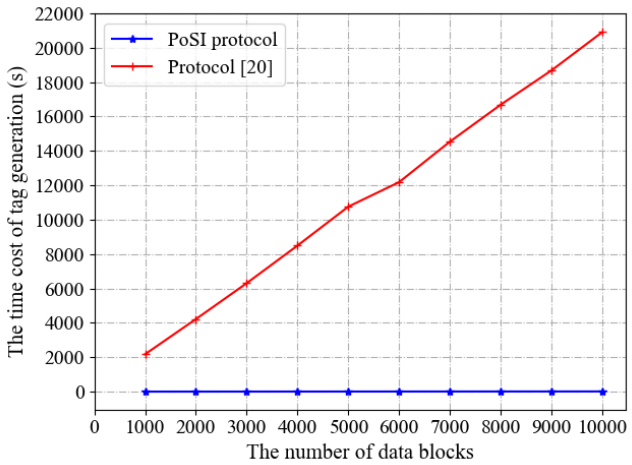

Figure 3. Computational cost in tag generation

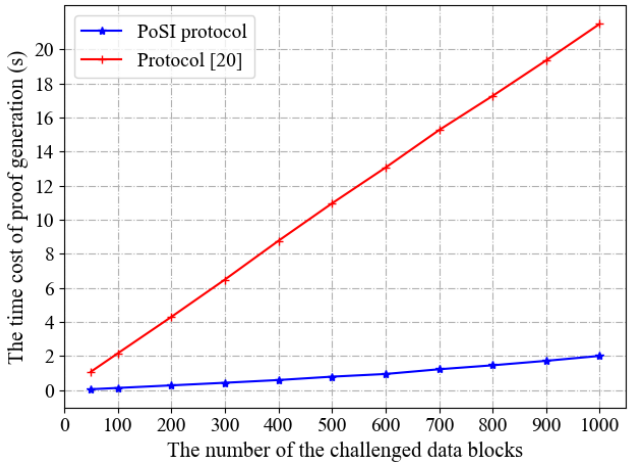

Figure 4. Computational cost in proof generation

5 Implementation of PoSI Protocol in the Blockchain

The PoSI protocol is a consensus protocol that leverages the storage ability of *SN* and makes them provide a storage-age-based proof of integrity for election *MN*. In this section, we introduce the implementation of PoSI protocol in the blockchain.

5.1 The Specific Work Process

The specific work process of PoSI in the blockchain is divided into four phases: Initialization, Storage, Consensus, Transaction.

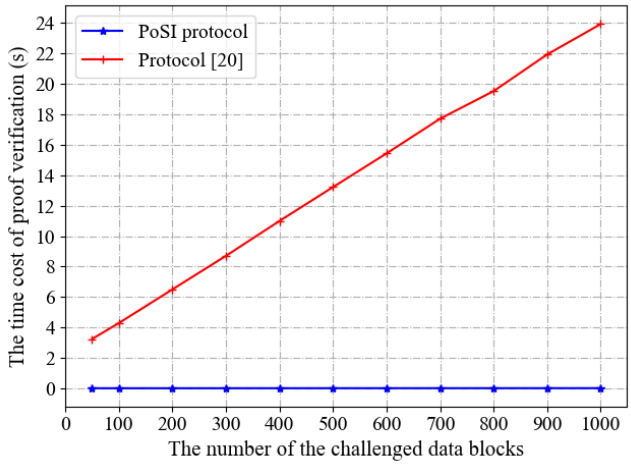

Figure 5. Computational cost in proof verification

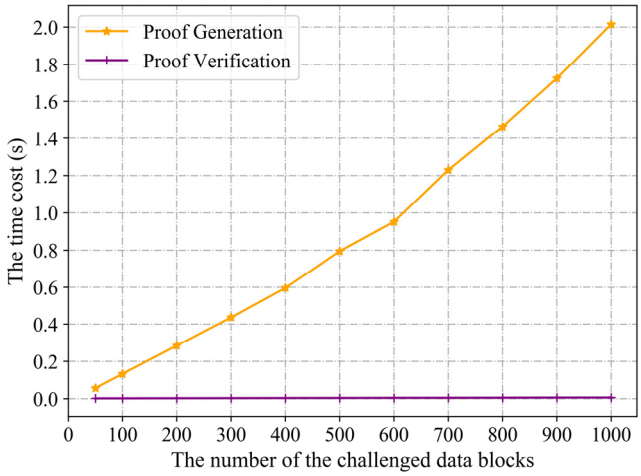

Figure 6. Comparison of computational cost

(1) Initialization

- The system runs the Setup to generate public security parameters.
- Nodes are free to join the network and then run the KeyGen to generate their public and private keys.

(2) Storage

- *UN* publishes a storage transaction in the blockchain when he wants to store his own large file to other nodes.
- It runs TagGen. Then the file blocks and tags are stored in *SN*, and some public parameters are stored in blocks.

(3) Consensus

- At the beginning of each period, *SN* who wants to be *MN* runs Chal and Proof, and sends *proof* to the consensus group.
- The members of the consensus group run Verify.
- When $f+1$ members validate the proof of *SN*, the consensus group broadcasts this *SN* becomes *MN*.

(4) Transaction

- *MN* packages transactions in the system into blocks.
- The consensus group verifies whether the transaction in the block is valid or not, just like the system [27].
- If valid, this block was pinned in the blockchain. Otherwise, *MN* is penalized by reducing its storage age to zero.

The consensus group verifies whether transactions in the block are valid according to a consensus protocol similar to PBFT [28], which we call a storage-age-based weighted voting mechanism. The only difference between the voting mechanism and PBFT is that each node in PBFT makes equal decisions, while the decision right of each node in the voting mechanism depends on the proportion of its storage age in the whole. The consensus group is a subset of miners who may control the operation of the system. The size is defined as the minimum number of miners with sufficient decision-making power, which is not predetermined.

5.2 Security Analysis

We demonstrate the security of the blockchain using PoSI from two aspects: the safety and liveness of the system and some existing attacks that the blockchain is able to resist.

5.2.1 Safety and Liveness of The Blockchain

Unlike PoW-based blockchains, when using PoSI, an attacker cannot break the system by merely relying on its mining ability, that is, its computing power. An attacker rather needs to gain storage age by storing files for the system, contributing to the blockchain by providing proof and generating blocks. The storage age of a miner with correct behavior, in the system used PoSI (hereinafter referred to as the PoSI blockchain), builds essentially on its continued contribution to the entire blockchain.

For simplicity, we assume that each miner runs virtually for a period of time, allowing the attacker enough storage age to attack the system.

Let $\mathbb{X} = \{x_1, \ldots, x_{|\mathbb{X}|}\}$ be the consensus group. We note A_i is the storage age of miner x_i, which represents it decision power. The PoSI blockchain rely on underlying secure consensus protocol PBFT to guarantee safety and liveness.

The PoSI blockchain guarantees consensus *safety*, if: (i) the attacker controls no more than f miners in the consensus group; or (ii) the consensus group members compromised by the attacker have a total storage age A_a such that

$$A_a < \frac{\sum_{i=1}^{|\mathbb{X}|} A_i}{3}.$$

In other words, unless both (i) and (ii) fail, an attacker cannot break the safety.

In addition, if either of the above two conditions is not true, then an attacker is able to deactivate the *liveness* of the PoSI blockchain, that is, no agreement

can be reached on any block.

5.2.2 Defense Against Specific Attacks

(1) We discuss the defenses of our PoSI blockchain against some common attacks. Table 3 compares the defenses of the Bitcoin system, the Byzantine system, and our PoSI system against some of the attacks.

Table 3. Attack resistance comparison

Attacks	Bitcoin [17]	ByzCoin [29]	PoSI Blockchain
Flash Attack	×	×	√
Selfish Mining Attack	×	×	√
Double Spending Attack	×	√	√
Eclipse Attack	×	√	√

(1) *Flash Attack*. In a flash attack, an attacker is able to temporarily gain most of the computing power by renting enough mining power. Nevertheless, our PoSI blockchain is resilient to flash attacks. When an attacker with a lot of computing and storage power joins in the system, it still needs to take a long time to gain enough storage age to damage the system.

(2) *Selfish Mining Attack*. The selfish mining pool has more than half of the computing power. The main idea of the selfish mining attack is that the selfish mining pool intentionally delays releasing the new block calculated by it, and constructs a private branch under its own control, causing the chain to fork. In the PoSI blockchain, the consensus group pins each created block, and *MN* only creates new blocks based on pinned blocks. Given that the PoSI blockchain relies on the consensus protocol based on the safety integrity verification scheme and storage age, the attacker cannot predict who may control at least f+1 *CN*s. Hence selfish miners will not by hiding it in the PoSI blockchain created to gain any advantage.

(3) *Double Spending Attack*. A Double Spending Attack is when the same amount of money is consumed multiple times in a blockchain system. The attacker has more than half of the computing power. Our PoSI blockchain addresses the double-spending attack by accelerating the commit process to less than a few seconds. In addition, the PoSI blockchain provides a consistent guarantee of certainty. Determinism uses a consensus protocol based on the voting mechanism of storage age to pin blocks. The pinned block is irreversible, which is the guarantee we provide using consensus protocol.

(4) *Eclipse Attack and Isolated Leaders*. An Eclipse attack is when an attacker adds enough fake nodes to a set of neighbor nodes of some nodes by encroaching on their routing table, thereby isolating them from the normal blockchain network. The Eclipse Attack is not valid on our PoSI blockchain. If the block is pinned, the attacker will not be able to successfully launch a double-spending attack, as described earlier. In the extreme case, some group members may be temporarily quarantined as a result of an attack on the network. Such network attacks might delay the block pinning process. However, the PoSI blockchain will be restored immediately after the message is delivered, and an attacker neither creates a branch in the blockchain nor spends any currency multiple times.

6 Conclusion

We introduce the definition of storage age and propose a new consensus protocol by combining it with data integrity verification, which solves the problems of resource waste in blockchain consensus. In the protocol, the pairing-free integrity verification protocol is employed to improve the verification efficiency. In addition, the protocol satisfies properties of unforgeability, and privacy preservation, which also protects the original data from substitution attack while verifying integrity. Finally, we give the specific application of PoSI protocol in the blockchain.

Acknowledgments.

This work is supported by the Key Research and Development Program of Shaanxi (Program No. 2020ZDLGY08-03) and the Shandong Provincial Key Research and Development Program of China (2019JZZY020129).

References

[1] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.

[2] S. King, S. Nadal, *PPcoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*, August, 2012.

[3] G. Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger, *Ethereum project yellow paper*, Vol. 151, pp. 1-32, 2014.

[4] S. Dziembowski, S. Faust, V. Kolmogorov, K. Pietrzak, Proofs of Space, *Annual Cryptology Conference*, Santa Barbara, CA, USA, 2015, pp. 585-605.

[5] T. Moran, I. Orlov, Proofs of Space-Time and Rational Proofs of Storage, *IACR Cryptology ePrint Archive*, Vol. 2016, Article No. 35, 2016.

[6] A. Miller, A. Juels, E. Shi, B. Parno, J. Katz, Permacoin: Repurposing Bitcoin Work for Data Preservation, *2014 IEEE Symposium on Security and Privacy*, Berkeley, California, USA, 2014, pp. 475-490.

[7] D. Francati, G. Ateniese, A. Faye, A. M. Milazzo, A. M. Perillo, L. Schiatti, G. Giordano, Audita: A Blockchain-Based Auditing Framework for Off-Chain storage, *arXiv preprint arXiv:1911.08515*, November, 2019.

[8] F. Schuh, D. Larimer, Bitshares 2.0: Financial Smart Contract Platform, 2015.

[9] I. Bentov, C. Lee, A. Mizrahi, M. Rosenfeld, Proof of Activity: Extending Bitcoin's Proof of Work Via Proof of Stake, *ACM SIGMETRICS Performance Evaluation Review*, Vol. 42, No. 3, pp. 34-37, December, 2014.

[10] L. Ren, *Proof of Stake Velocity: Building the Social Currency of The Digital Age*, April, 2014.

[11] 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.

[12] G. Ateniese, L. Chen, M. Etemad, Q. Tang, Proof of Storage-Time: Efficiently Checking Continuous Data Availability, *Network and Distributed System Security Symposium*, San Diego, California, USA, 2020, pp. 1-15.

[13] 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.

[14] H. Wang, Q. Wang, D. He, Blockchain-Based Private Provable Data Possession, *IEEE Transactions on Dependable and Secure Computing*, pp. 1-1, October, 2019.

[15] D. Yue, R. Li, Y. Zhang, W. Tian, C. Peng, Blockchain Based Data Integrity Verification in P2P Cloud Storage, *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, Singapore, 2018, pp. 561-568.

[16] Y. Zhu, H. Hu, G. Ahn, M. Yu, Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage, *IEEE transactions on parallel and distributed systems*, Vol. 23, No. 12, pp. 2231-2244, December, 2012.

[17] R. Chen, Y. Li, Y. Yu, H. Li, X. Chen, W. Susilo, Blockchain-Based Dynamic Provable Data Possession for Smart Cities, *IEEE Internet of Things Journal*, Vol. 7, No. 5, pp. 4143-4154, May, 2020.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] J. Han, Y. Li, W. Chen, A Lightweight and Privacy-Preserving Public Cloud Auditing Scheme without Bilinear Pairings in Smart Cities, *Computer Standards & Interfaces*, Vol. 62, pp. 84-97, February, 2019.

[23] C. P. Schnorr, Efficient Identification and Signatures for Smart Cards, *Conference on the Theory and Application of Cryptology*, Santa Barbara, California, USA, 1989, pp. 239-252.

[24] X. Zhang, S. Liu, S. Han, Proofs of Retrievability from Linearly Homomorphic Structure-Preserving Signatures, *International Journal of Information and Computer Security*, Vol. 11, No. 2, pp. 178-202, January, 2019.

[25] Y. Li, Y. Yu, R. Chen, X. Du, M. Guizani, Integritychain: Provable Data Possession for Decentralized Storage, *IEEE Journal on Selected Areas in Communications*, Vol. 38, No. 6, pp. 1205-1217, June, 2020.

[26] M. Lepinski, S. Kent, Additional Diffie-Hellman Groups for Use with IETF Standards, RFC 5114, January, 2008.

[27] J. Yu, D. Kozhaya, J. Decouchant, P. Esteves-Verissimo, Repucoin: Your Reputation Is Your Power, *IEEE Transactions on Computers*, Vol. 68, No. 8, pp. 1225-1237, August, 2019.

[28] M. Castro, B. Liskov, Practical Byzantine Fault Tolerance and Proactive recovery, *ACM Transactions on Computer Systems*, Vol. 20, No. 4, pp. 398-461, November, 2002.

[29] P. Jovanovic, ByzCoin: Securely Scaling Blockchains, *Hacking, Distributed*, August, 2016.

Biographies

Chang Wang received the B.S. degree from Xidian University, Xi'an, China, in 2018. She is currently a M.S. candidate of the School of Cyber Engineering, Xidian University, Xi'an, China. Her research interests include blockchains and data security.

Jun Shen received the B.S. and M.S. degrees from the Nanjing University of Information Science and Technology, Nanjing, China, in 2015 and 2018, respectively. She is currently pursuing the Ph.D. degree with the School of Cyber Engineering, Xidian University, Xi'an, China. Her research interests include cloud computing security and blockchains.

Shichong Tan received the B.S. degree in Telecommunications Engineering in 2002, the M.S. and Ph.D. degrees in Cryptography from Xidian University, China, in 2005 and 2009, respectively. Since 2005, he has been with Xidian University, where he is now an Associate Professor. His research interests include cloud computing and blockchains.