Proof of Continuous Work for Reliable Data Storage Over Permissionless Blockchain

Hao Yin, Zijian Zhang, Jialing He, Liran Ma, *Member, IEEE*, Liehuang Zhu, *Member, IEEE*, Meng Li, *Member, IEEE*, and Bakh Khoussainov

***Abstract*—Bitcoin first proposed the Nakamoto consensus that applies proof of work into the blockchain structure to build a trustless append-only ledger. The Nakamoto consensus solves the distributed consistency problem in the public network but wastes too much computing power. Instead of consuming computing resources, many improved consensus schemes address this problem by leveraging miners' storage resources. However, these schemes fail to let miners store data constantly and usually rely on a dealer to assign data, which is hard to build a reliable decentralized storage system. In this article, we first design a variant consensus algorithm named Proof of Continuous Work (PoCW) with a storage-related incentive mechanism. Miners can accumulate mining advantage by continuously submitting proofs of storage. Then, we present a hash ring-based data allocation algorithm using the blockchain's state. Combined with both of them, we build a reliable blockchain-based storage system without relying on any third parties. The theoretical analysis and simulation results demonstrate that the proposed system has higher reliability than those existing systems, and we also give practical suggestions about system parameters. Finally, we discuss additional benefits that our system brings.**

***Index Terms*—Blockchain, data allocation, decentralized storage, Proof of Continuous Work (PoCW), reliability.**

I. Introduction

BLOCKCHAIN [1] is known as decentralized database technology, and it can build a unique ledger among the untrusted distributed network. Such technology can effectively cut down on trust costs in distributed systems and thus, be studied by many scholars. Bitcoin [2] first proposes this blockchain structure to design a cryptocurrency, which applies Proof-of-Work (PoW) to construct the Nakamoto consensus. In the consensus protocol, participants compete for appending a new block to the current blockchain by solving a difficult puzzle. Generally, only the fastest participant wins the competition and gets the reward. Others have to drop the current puzzle for computing the next one, which wastes massive computing power.

Researchers proposed some PoW-variant schemes to figure out the mentioned problems. For example, Proof of Stake (PoS) [3] extends the framework of PoW and uses stakes to influence block mining difficulty, where the stake can be the cryptocurrency on the blockchain. The more the stake a participant holds, the easier the participant mines a valid block and gets the reward. Although no longer wasting lots of computing power, it faces the risk of instability. An adversary may create multiple identities (i.e., Sybil attack [4]) to increase its opportunity of successful mining. Another attack named nothing at stake [5] can easily cause blockchain forks, which ruins the consistency. Nevertheless, PoS shows a possible idea that we can substitute the consumption of computing resources with others.

Inspired by PoS, many state-of-art schemes utilize storage contribution to substitute computing power. On the one hand, storage is not a virtual resource so that it can tolerate the Sybil attack. On the other hand, storage provides a useful service that can be used to earn income. Fortunately, there are cryptographic primitives, such as proof of retrievability (POR) [6] and proof of data possession (PDP) [7]. Based on these techniques, Permacoin [8] proposes a local POR to construct a useful consensus. It assumes a dealer to assign data to miners who are encouraged to submit proof of storage for higher success in block mining. Filecoin [9] commercializes the data storage activities via introducing a distributed storage market. It also proposes a variant Nakamoto consensus that biases the mining advantage to the miners who provide more storage space.

However, the existing schemes are hard to build a reliable storage system. The first reason is that those consensus algorithms only concern storage capacity. They lack an incentive mechanism to encourage miners to constantly store the data. Second, these systems usually depend on a centralized party to allocate data for miners. Such mode might bring extra trust

Manuscript received June 2, 2021; revised July 19, 2021 and August 29, 2021; accepted September 15, 2021. Date of publication September 27, 2021; date of current version May 9, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62172040, Grant U1836212, and Grant 61872041; in part by the Ministry of Education—China Mobile Research Fund Project under Grant MCM20180401; and in part by the National Science Foundation of the U.S. under Grant CNS-1912755. *(Corresponding authors: Zijian Zhang; Liehuang Zhu.)*

Hao Yin is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: yinhao@bit.edu.cn).

Zijian Zhang, Jialing He, and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: zhangzijian@bit.edu.cn; hejialing@bit.edu.cn; liehuangz@bit.edu.cn).

Liran Ma is with the Department of Computer Science, Texas Christian University, Fort Worth, TX 76129 USA (e-mail: l.ma@tcu.edu).

Meng Li is with the Key Laboratory of Knowledge Engineering with Big Data, Ministry of Education, and the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China (e-mail: mengli@hfut.edu.cn).

Bakh Khoussainov is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: bmk@uestc.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3115568

cost and somewhat harm storage reliability. It may cause that the blockchain-based storage system weakens to a centralized cloud storage system. To address the challenges, we propose a variant Nakamoto consensus with a storage-related incentive mechanism to build a blockchain-based storage system, which allocates data to multiple miners without any trusted third parties. The contribution of our work is as follows.

1) We design a Proof-of-Continuous-Work (PoCW) algorithm to encourage miner participation in storage contribution. In this variant Nakamoto consensus, miners accumulate the mining advantages by continuously storing data and submitting proofs.
2) We propose a hash ring-based data allocation algorithm to assign data in a decentralized fashion. This method leverages consistent hashing to allocate data to storage nodes. The allocation result is used to obtain the assigned data in the PoCW computation.
3) Theoretical analysis proves that our allocation scheme gets higher reliability than the centralized allocation scheme. According to the results, we also investigate the practical situation of node failure and give useful suggestions about system parameters.

The remainder of this article is organized as follows. We present related work in Section II. We then introduce the preliminary notations and definitions in Section III. Section IV gives an overview of the proposed blockchain-based storage system. Section V presents proof-of-continuous-work and hash ring-based data allocation. After that, we give the reliability analysis and simulation verification in Section VI. Finally, we conclude this article in Section VIII.

II. Related Work

In this section, we briefly show some research work about blockchain-based storage systems in various domains ranging from the Internet of Things (IoT) to Healthcare. Wang *et al.* [10] presented ForkBase that built an efficient storage engine with collaborative analytics and fork semantics. Li *et al.* [11] designed a secure and accountable IoT storage system based on blockchain. Xu *et al.* [12] leveraged blockchain to execute data analytics with IoT devices. Zhou *et al.* [13] proposed MIStore to build a blockchain-based medical insurance storage system. Wang *et al.* [14] proposed BlockZone, a blockchain-based DNS system with an improved PBFT consensus algorithm. Chen *et al.* [15] designed a blockchain-based medical service framework for personal data management.

To address the issues of storage limitation, a few mainstream blockchain-based storage systems relieve capacity restrictions by storing raw data off-chain. An earlier work Factom [16] created a data layer on top of Bitcoin, where data are retrieved using distributed hash table (DHT) and metadata are stored on-chain. Later, Zyskind *et al.* [17] used blockchain to manage user permissions and message delivery. Inspired by Namecoin [18], Blockstack [19] constructed a global storage system by embedding zone files into Bitcoin. Recently, Li *et al.* [20] designed a blockchain-based distributed cloud storage architecture, where transactions contain the URLs of file replicas that can be retrieved from a distributed storage network (DSN).

Some systems adopt proof protocols to synchronize off-chain data with blockchain for high reliability. Based on various proofs of storage, Miller *et al.* [8] proposed Permacoin to make consensus work for data preservation, where miners need to generate local POR to get more mining rewards. Storj [21] enabled users to sell and buy storage space from providers, which can be audited by any peers via a challenge-based POR protocol. This also introduced sharding, an extension of Kademlia [22], to further improve system reliability. Kopp *et al.* [23], [24] used a POR protocol to build a distributed file storage but paid more attention to financial incentives to ensure fairness. Filecoin [9] built a distributed storage market, where clients and miners send *bid* and *ask* orders, respectively. This work also proposed a noninteractive proof of space time to reach reliability. Ateniese *et al.* [25] proposed a similar notion named proof of storage time. It designs a challenge–response protocol to efficiently verify that data are continuously available and retrievable for a range of time.

One important assumption in these works is the honesty of the third party to allocate data. Without it, the systems' reliability can significantly be degraded. One novelty of our work thus lies in building decentralized data allocation in blockchain-based storage systems.

III. Preliminaries

A. Blockchain in Bitcoin

Bitcoin was proposed by Nakamoto [2] in 2009, which is the first practical application of blockchain. The system organizes transactions via a hash chain and relies on PoW to build the Nakamoto consensus.

Blockchain structure is a chain of blocks that pack transactions within an interval of time. Each block includes hash value pointing to the previous block, except for the first one (i.e., genesis block). In Bitcoin, a block consists of a list of transactions, a hash value, a version number, a timestamp, and a random nonce. Let $H(\cdot)$ be a cryptographic hash function. For simplicity, we formalize the ith block as

$$B_i := (T_i||H(B_{i-1})||N_i)$$

where B_i is the ith block, T_i is the list of transactions in this block, and N_i is the random nonce selected by miners. Thus, the blockchain with height $h \geq 0$ can be viewed as the sequence of blocks $L_h :=< B_0, B_1, \ldots, B_h >$, where B_0 is the genesis block.

Nakamoto consensus is an essential component that makes all participants (also called miners) hold the same ledger. After verifying all the transactions by certain specified rules, miners compete with each other to solve a computational puzzle. The solution of this puzzle is to find an appropriate random nonce that makes the hash of the block less than a target value of Z, named PoW. We express it as follows:

$$H(B_i) < Z.$$

Multiple solutions for the current puzzle may cause the blockchain fork. To address it, all miners follow the longest chain to solve puzzles, and thus, the blockchain can eventually reach global consistency with high probability.

B. Consistent Hashing

Consistent hashing [26] forms a ring structure to "smoothly absorb" network changes, which allows servers to join or leave in an arbitrary order with limited costs. It connects the output of a hash function from the beginning to the end via a modulo operation to construct the hash ring, where requests and servers are mapped into the same space. Requests find the closest server on the space or the first server encountered in a certain direction on the ring.

Due to high efficiency in resource allocation and localization, the consistent hashing concept (used in designing DHT) has been applied to many peer-to-peer storage systems, such as chord [27], Pastry [28], etc. Typically, the previous works use this technology to locate nodes for data retrieval without assigning data storage. Abe's thesis [29] aims to alleviate miner's overhead for storing the ledger, where each miner only stores part of blocks according to the DHT. But the paper, in some respects, failed to build a blockchain-based storage system because that paper did not consider to synchronize DHT with data off-chain. In our work, we directly adopt consistent hashing to construct a global hash ring structure rather than DHT. Such a hash ring not only can locate nodes for data retrieval but it can also manage distribution for data storage.

C. Local Proof of Retrievability

In 2007, Juels and Kaliski, Jr [6] proposed a POR mechanism that is a challenge–response protocol, where the prover convinces the verifier that the file in its possession can be correctly retrieved. Miller *et al.* [8] presented a local POR version and applied it to the blockchain for repurposing POW computation to data preservation. The local POR scheme embeds a secret key of miners in the proofs. Miners have to store the data locally for generating proofs on time to get a reward, and thus, restricting miners' behavior of data outsourcing.

Here, we formally define the local POR as follows.

1) *Gen*$(1^{\lambda}) \rightarrow (pk, sk)$*:* The Gen algorithm executed by a miner takes a security parameter λ as input and outputs a public and secret key pair (pk, sk).
2) *Encode*$(F) \rightarrow (rt, \hat{F})$*:* The Encode algorithm encodes a large file F to a file $\hat{F}$ with erasure code. It computes the root rt of a Merkle tree whose leaves are segments of the encoded file.
3) *Prove*$(sk, R, \hat{F}) \rightarrow \pi$*:* The Prove algorithm inputs a secret key sk, a random challenge R, and an encoded file $\hat{F}$. It outputs a proof π, containing the file segments, signatures, and the Merkle tree paths.
4) *Verify*$(pk, rt, R, \pi) \rightarrow \{0,1\}$*:* The Verify algorithm takes a public key pk, a Merkle tree's root rt, a random challenge R, and a proof π. If the signatures and the proof are valid, it outputs 1, otherwise, 0.

The concrete construction of this scheme is omitted. We recommend the readers to refer the original paper for details.

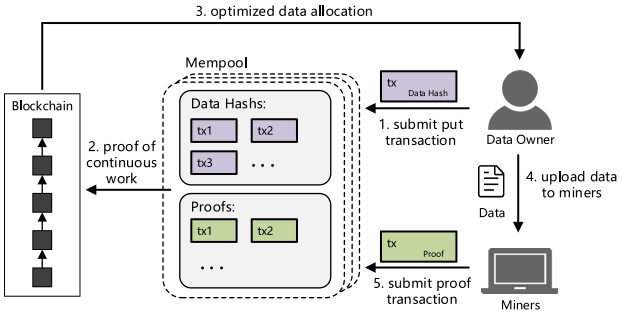

Fig. 1. System model.

In our system, we treat the local POR as a cryptographic primitive. Miners use it to compute the proof of storage.

IV. Blockchain-Based Storage System

In this section, we give an overview of system participants and make some assumptions for them. Our system can rely on PoW computation to providing reliable decentralized data storage. In the rest of this article, we focus on building such a system on the permissionless blockchain.

A. System Overview

We have following three entities in the system.

1) *Data Owner:* A user who uploads and retrieves data in the system.
2) *Miner:* A peer in the blockchain network who participates in mining and provides storage.
3) *Blockchain:* A public distributed ledger that accepts transactions and updates the global state.

The key idea is to let the miners introduce storage contribution in mining a new block. We modify the Nakamoto consensus and lean the advantage in mining to those miners who submit the proof of storage to the blockchain. Then, we use consistent hashing to organize data distribution, building a decentralized and autonomous data storage system.

We illustrate the system model in Fig. 1. A data owner constructs a Put transaction and submits it to the blockchain. Miners compute the corresponding proofs of storage, continuously sending Proof transactions for showing their work. Note that the two types of transactions (Put and Proof) determine the blockchain's state, which consists of a data set and a miner set, respectively. We build a mapping relation between the two sets to allocate data for miners. Once new data are registered on the blockchain, it triggers the state changes and conducts data allocation, and then the data owner will send raw data to the designated miners. Iteratively, the miners update local storage and continue to generate proofs.

Assumptions: We consider a permissionless setting in which participants can join to be miners or leave at any time. Miners have a certain storage capacity and use their public keys as pseudonyms identity. We assume that miners are rational to get financial rewards in mining and transaction fees. Adversaries can only control partial miners, where the sum of storage is less than 50% of total storage. We also assume that data

are split into segments with the same size and distributes to multiple replicas for retrieval reliability.

B. Transaction Definition

The blockchain accepts two special transactions: 1) the Put transaction and 2) the Proof transaction. We use these transactions to register data and present proofs of storage, respectively. At a high level, we treat the blockchain as an arbitrary state machine where the sequence of transactions is taken as inputs to trigger state changes.

For the Put transaction, it requests leasing storage and decides the length of the lease. The blockchain keeps data in the state until the data expire at some block height. We let *did* denote the identity of data. The proofs contain a Merkle tree's root, which is denoted by *drt*. Also, we let *tts* be a block-based counter that determines the valid duration of data. The transaction has the following form:

$$\mathsf{tx}_{\text{put}} := (did, drt, tts).$$

For the Proof transaction, we let *pk* denote the public key of the miners to represent their identity. Theoretically, the miner can arbitrarily generate key pairs, choosing different public key as its identity. We use the Prove algorithm in the local POR scheme to construct a proof of storage π, where it involves a random challenge R. Once the miner broadcasts this transaction, others execute the Verify algorithm to check if it is valid. We formulate the transaction as follows:

$$\mathsf{tx}_{\text{proof}} := (pk, R, \pi).$$

To provide storage, a miner first submits a transaction without any proofs and random challenges. The default transaction declares that the miner is ready to contribute storage. Then, the miner gets the results of data allocation, which is presented in Section V-C, selects a random challenge, and uses its secret key to compute a proof. We explain how to select the random challenge in Section V-A. To ensure an available storage service, we require each miner to submit proofs constantly. This process just like a heartbeat, which means that a miner turns to fail if the blockchain cannot receive any valid proof within certain blocks. Thus, miners continuously work for constructing valid Proof transactions and refresh the blockchain's state to be active.

V. Proof of Continuous Work for Storage

In this section, we design a variant Nakamoto consensus with the corresponding incentive mechanism in storage contribution. After that, we present a decentralized data allocation algorithm based on consistent hashing.

A. Chain the Proof Transactions

Here, we consider a chain structure to organize those proofs. Each Proof transaction includes a reference to the last transaction owned by the same miner, making the first one be a register action. By doing so, we can address the following two problems. First, the transaction refers to a certain block, pointing to a determined state. We can use the state to conduct data allocation for miners. Second, the block hash and the transaction index in that block are unpredictable when submitting a transaction. Miners must wait for the previous transaction packed into the blockchain before construing the next one. It effectively prevents miners from generating a sequence of proofs in advance so that it can drop or outsource the data.

We modify the original format of the Proof transaction by adding two reference fields as follows:

$$\mathsf{tx}_{\text{proof}} := (bid, idx, pk, R, \pi)$$

where the previous transaction is located by the block hash *bid* and the index *idx* in the transaction lists. We use the PoW-like technique to constraint the transaction that its hash value must be less than a target value of V. Thus, miners need to attempt several random challenges.

Based on the observation that miners with longer transaction chains and more stored data are more reliable, we consider the following two parameters to adjust the mining difficulty. First, we let *ctr* denote the length of the Proof transactions chain to evaluate the miner's workload. Second, we let *num* denote the number of assigned data. To protect the miners who do not participate in storage contribution, we reserve the base difficulty in PoW for them. We have the following formula as the mining difficulty:

$$H(B_i) < (ctr * num + 1) * Z.$$

The pseudocode of PoCW is shown in Algorithm 1. Miners who do not provide storage run *mining* function in lines 6–14 with the same effect as PoW. However, they have an optional task that defines in the *proving* function in lines 15–31. Miners can create a thread to continuously submit proof, which changes the variable of the counter *ctr* and the number *num* to affect the mining difficulty in line 11. To avoid the accumulative *ctr* increase without limitation, making the blockchain always controlled by the oldest miner. We add a rule in line 14, i.e., the counter will be clear and recalculate when a miner appends a new block to the blockchain.

Algorithm 2 describes the process of blockchain when receiving a new block. Miners use this same procedure to validate the block and append it to the blockchain. At a high level, the blockchain keeps the state of data set and miner set at every block height. Lines 7–10 refresh the data set and the miner set, removing those overdue data and inactive miners. We use the symbol # to denote iterating all elements in the set. Lines 11–15 process Put transactions while lines 16–29 process Proof process transactions, which update the state in the previous block. Similarly, once the new block from a miner is successful appended, the blockchain will clear the miner's counter in line 5.

As shown in Fig. 2, the middle row is a blockchain in a simplified version. At each block, the upper square presents the current blockchain's state, and the square below lists the transactions packed in the block. For example, we set the maximum interval blocks between two consecutive Proof transactions to 2, which simulates the heartbeat of miners. Data 1 (denoted as d1) is loaded by a Put transaction in block N within three blocks. Thus, the corresponding state keeps d1 until block N + 2. Similarly, the blockchain has data 2 (denoted as d2)

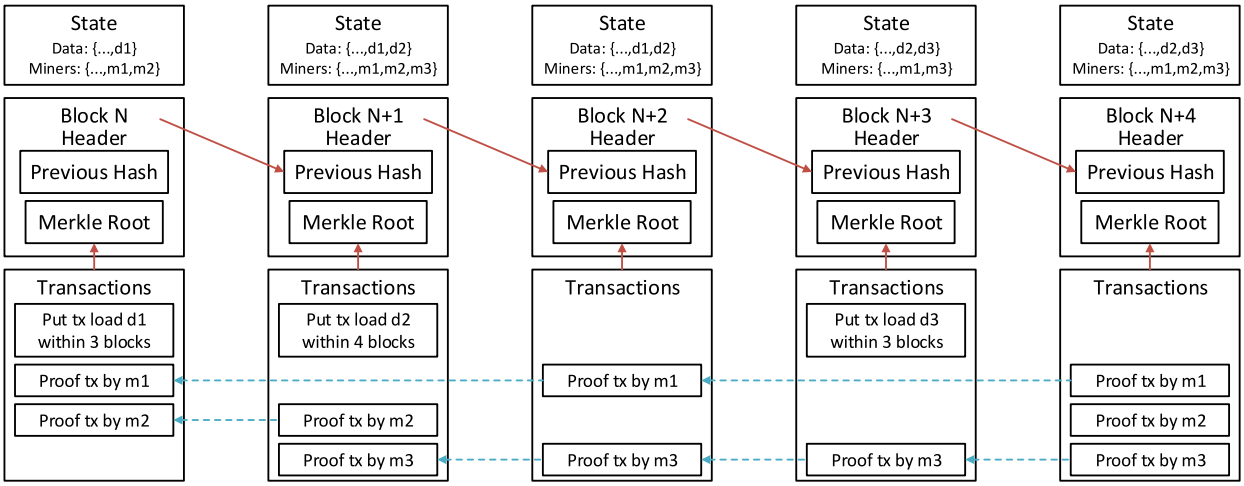

Fig. 2. Chains of proof transactions.

Algorithm 1 PoCW for Miners

function init()
  Generate a key pair (sk, pk) as a pseudonyms
  Set a counter $ctr = 0$ for the consecutive proofs
  Set a number $num = 0$ for the amount of data
  Set a pointer $ltx = null$ for the last proof transaction
function mining()
  for true **do**
    Sync the longest chain and get the last block B_h
    Collect transactions txs from memory pool
    Randomly select a number $nonce$ and pack a new block $B_{h+1} = (txs||H(B_h)||nonce)$
    if $H(B_{h+1}) < (ctr * num + 1) * Z$ **then**
      Broadcast this new block to network
      if the blockchain accepted this block **then**
        clear the counter $ctr = 0$
function proving()
  Broadcast $tx_{proof} = (pk)$ to network
  wait for this transaction write to blockchain
  Set the last proof transaction $ltx = H(tx_{proof})$
  for true **do**
    Locate the hash bid of the block and the index idx of transaction ltx from blockchain
    Query the state of data set $\mathcal{D}$ and miner set $\mathcal{M}$
    Call $\mathcal{L} = allocation(pk, \mathcal{D}, \mathcal{M})$ for assigned data
    for true **do**
      Randomly select a number R as a challenge
      Call $\pi = \mathsf{Prove}(sk, R, \mathcal{L})$ to compute a proof
      Construct $tx_{proof} = (bid, idx, pk, R, \pi)$
      if $H(tx_{proof}) < V$ **then** broadcast it and break
    wait for this transaction write to blockchain
    Set the last proof transaction $ltx = H(tx_{proof})$
    Set the amount of data $num = len(\mathcal{L})$
    Increase the counter $ctr = ctr + 1$
function main()
  Call $init()$ to initialize the node
  Create the main thread to run $mining()$
  Create a thread to run $proving()$ in the background

Algorithm 2 Protocol for Blockchain

On receive a new block B from miner m **do**
  Check the format of blocks and transactions
  The current state $S = \mathcal{M}_h[m.pk]$ with block height h
  if $H(B) < (S.ctr * S.num + 1) * Z$ **then**
    clear the counter $\mathcal{M}_h[m.pk].ctr = 0$
  else Drop this block and exit
  Let $\mathcal{D}_{h+1} = \mathcal{D}_h$, $\mathcal{D}_{h+1}[\#].tick = \mathcal{D}_h[\#].tick - 1$
  Delete $\mathcal{D}_{h+1}[\#]$ where $\mathcal{D}_{h+1}[\#].tick == 0$
  Let $\mathcal{M}_{h+1} = \mathcal{M}_h$, $\mathcal{M}_{h+1}[\#].tick = \mathcal{M}_h[\#].tick - 1$
  Delete $\mathcal{M}_{h+1}[\#]$ where $\mathcal{M}_{h+1}[\#].tick == 0$
  Process Put transactions $tx_{put} = (did, drt, tts)$
  if $\mathcal{D}_h[did]$ is empty **then**
    $\mathcal{D}_{h+1}[did].root = drt$, $\mathcal{D}_{h+1}[did].tick = tts$
  else if $\mathcal{D}_h[did].root == drt$ **then**
    $\mathcal{D}_{h+1}[did].tick = \mathcal{D}_h[did].tick + tts$
  Process Proof transactions tx_{proof}
  if $tx_{proof} = (pk)$ **then**
    $\mathcal{M}_{h+1}[pk].ctr = 0$, $\mathcal{M}_{h+1}[pk].num = 0$
    $\mathcal{M}_{h+1}[pk].tick = interval$
  else if $tx_{proof} = (bid, idx, pk, R, \pi)$ **then**
    if $\mathcal{M}_{h+1}[pk]$ is empty **or** $H(tx_{proof}) \geq V$ **then**
      Drop this block and exit
    Get the block height i of block bid
    Get assigned data $\mathcal{L}_i = allocation(pk, \mathcal{D}_i, \mathcal{M}_i)$
    Verify the proof $res = \mathsf{Verify}(pk, \mathcal{L}_i, R, \pi)$
    if $res == 1$ **then**
      $\mathcal{M}_{h+1}[pk].ctr = \mathcal{M}_h[pk].ctr + 1$
      $\mathcal{M}_{h+1}[pk].num = len(\mathcal{L}_i)$
    else Drop this block and exit
  Append a new block and hold the state $\mathcal{D}_{h+1}, \mathcal{M}_{h+1}$

in the data set until block N + 4. Miner 1 (denoted as m1) in the miner set from Block N and generates a Proof transaction every other block. It does not exceed the maximum interval blocks so that m1 is in the state all the time. Miner 2 (denoted as m2) submits the last Proof transaction in Block N + 4, but the previous one is in Block N + 1, exceeding the maximum interval blocks. Thus, the chain owned by miner 2 is broken off so that m2 is not in the state of block N + 3.

B. Incentives for Data Storage

In our system, we explain how it makes a positive effect on PoCW. There are some notations listed in Table I.

We consider the miners' profit from their income and expenditure, respectively. For a single miner, the income consists of a rental payment from the data owner and block rewards from

TABLE I
NOTATION USED

Notation	Description
δ	Miner's income from every single proof transaction
γ	Reward of mining a new valid block
ω	Fee from every single proof transaction
τ	Power cost in one block time
κ	A constant value of storage cost
μ	A time unit counted by blocks

successful mining. Each proof transaction earns a piece of fees denoted as δ. Once mining a valid block, the miner gets a considerable reward γ that contains a created token from Coinbase and packed transaction fees. Let the unit time be a block period, we denote ω and τ as one Proof transaction fee and computing cost in one block time, respectively. The storage cost is a fixed value of κ.

Assuming a miner spends μ block time in the system, submits the valid proof transaction in every block, and gets a reward because of successful mining, the miner will get the profit $\rho = (\mu\delta + \gamma) - (\mu(\omega + \tau) + \kappa)$. Two requirements must be satisfied to reach an incentive mechanism. First, $\delta > \omega + \tau$ can let the miner have an initial motivation to participate in providing storage. Second, $\gamma >> \kappa$ gives a large interest to keep the miner stays on making proofs. The existing consensus algorithm biases mining advantages to the miner who provides more storage so that γ here is a probabilistic function positively related to κ. In our new consensus algorithm, we introduce an extra accumulated variable that counts the consecutive proof transactions. Thus, γ is also positively related to μ.

We let $\gamma \sim \kappa$ and $\gamma' \sim (\kappa, \mu)$ denote the profit function of block reward in the existing consensus and the proposed consensus, respectively. The transactions under both consensus algorithms have the same income and expenditure, which means that in our consensus, the longer the miner participates, the more the miner gains than before. If a miner quits from storage contribution halfway, it will lose a massive accumulated mining advantage. Thus, our scheme gets more stable participants under the rational assumption of miners. Moreover, the Proof transaction chain leaves a trick to adjust the interval time of submitting a valid proof. We can increase the difficulty of the transaction chain to relax the interval time of two consecutive proof transactions. For example, if an average time of generating a valid proof transaction is 3, the interval time can be set to 4. That is, to say, we cut down the transaction fee on item $\mu\omega$ by the interval times but keep other costs no changes.

C. Hash Ring-Based Data Allocation

Given a block height, we have the determined blockchain's state, including a data set and miner set. We apply consistent hashing to the state to design a data allocation algorithm as follows. Initially, the algorithm maps data and miners into points in the hash ring structure, and then defines a mapping function that assigns data to miners. Inspired by Chord [27], we require that a certain number of successor nodes (counting by clockwise) on the hash ring store data, where this number is a crucial system parameter that represents the amount of data backup.

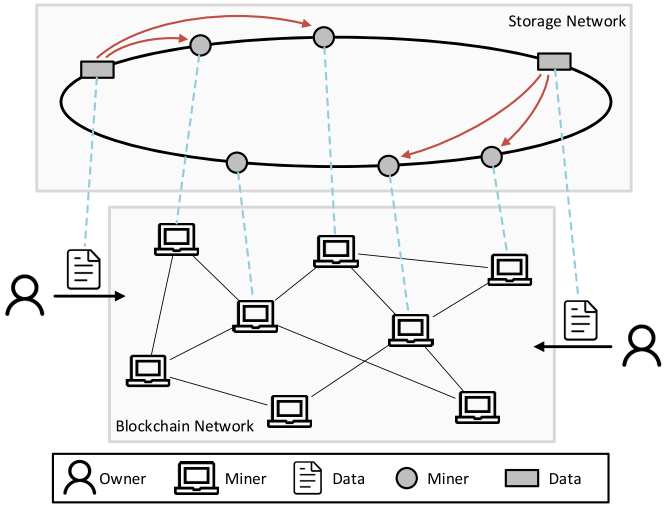

Fig. 3. Hash ring-based data allocation using consistent hashing.

Algorithm 3 Getting Assigned Data

Input:
A miner's public key, pk;
A set of data' identities, $\mathcal{D}$;
A set of miners' identities, $\mathcal{M}$;
Output:
The list of assigned data' identities, $\mathcal{L}$;
Initialize an empty list $\mathcal{L} := \emptyset$ for assigned data;
Assume the set $\mathcal{D}$ and $\mathcal{M}$ are sorted, the size of hash ring is $rSize$, let $\mathcal{D} := \{d_1, ..., d_u\}$, $\mathcal{M} := \{m_1, ..., m_v\}$, where u and v are the size of $\mathcal{D}$ and $\mathcal{M}$ respectively;
Get system parameter k, the amount of data backup;
Compute the identity of miner, i.e. $m_j = H(pk)$;
Find the index of two miners, j and $j-k$;
if $j - k > 0$ **then**
  $\mathcal{L} = \{f_i : f_i \in (m_{j-k}, m_j)\}$;
else
  $\mathcal{L} = \{f_i : f_i \in (m_{j-k+v}, rSize)\} \cup \{f_i : f_i \in (0, m_j)\}$;
end if
return $\mathcal{L}$

Fig. 3 illustrates the procedure of data allocation, where we abstractly divide the system into the blockchain network and the storage network (consisting of all miners who provide storage). Note that anyone can join the blockchain network to become a miner, and even upgrade to a storage contributor. The blockchain network accepts the two types of transactions to determine the data set and the miner set, while the storage network holds all raw data in the miners. We map the identities of data by H(data) and the identities of miners by $H(pk)$, where $H(\cdot)$ is a hash function. As shown in Fig. 3, the blue dash lines represent the mapping relationship, while the solid square and circle represent the identities of data and miner, respectively. Data are assigned to multiple successor nodes on the hash ring, which is represented by the lines with a directed arrow in red.

Algorithm 3 is a pseudocode of the data allocation procedure. Note that each block has its state of the data and miner sets. Thus, for this algorithm, we first need to specify

the block height and then invoke the algorithm. The algorithm takes the two sets as inputs and also enters a miner's public key *pk*. The output is a list of assigned data for the miner. At line 4, the algorithm computes the hash value of the miner's public key as the identity. Due to the ring structure, assigning data to k successor nodes is equivalent to that storing all data in the interval starting from the kth miner up to the miner itself. Lines 5–10 provide data distribution operations to miners.

VI. Evaluation

In this section, we analyze the data reliability of the proposed system built from PoCW. To verify our theoretical analysis, we make some simulations to compare with the existing schemes. Based on the experimental results, we give some practical suggestions about system parameters.

A. Security Analysis

The system relies on the proposed variant Nakamoto consensus to build the blockchain. We follow the difficulty setting in Bitcoin and lean some advantages to miners who continuously contribute local storage. On one hand, miners who do not involve in storage contributions still generate new blocks just like in Bitcoin. Our system keeps the same security when no miners participate in providing local storage. On the other hand, miners get mining advantages by accumulating storage contribution value, which will be reset once the miner has used it in mining a new block.

The mining advantage cannot be accumulated all the time, and it does not widen the gap from the normal miners in mining new blocks. If all miners have participated in storage contribution, the effect of the competition is similar to that of nobody participated. Besides, the security of our system is nearly equivalent to proof of stake, where the storage contribution represents the stake. Miners compute the stake via a deterministic algorithm using the historical state of the blockchain. The result is efficiently verifiable, and thus, it does not affect the stability of the consensus.

Overall, the proposed variant Nakamoto consensus is secure in the current blockchain network.

B. Reliability Analysis

We analyze the factors that affect system reliability and try to find the correlations between them. Besides, we also compare the hash ring-based data allocation with the dealer manipulated data allocation. By doing so, we conduct evaluations and give suggestions to build such a system. For convenience, we list our parameters in Table II.

In the hash ring-based data allocation, there is no third trusted party for distributing data. However, the existing systems generally designate a miner as a dealer to conduct data allocation. Here, we assume this dealer may incur similar node failures just like a miner. We consider that nodes failed randomly and simultaneously in a short interval time. First, if the interval between two failures is too long, the system can adjust itself to a new reliable state. Second, data loss means that all backup nodes are failed. Otherwise, the system can always recover the lost backup from the storage network. Recall that the node randomly selects a public key to compute the hash value as its identity. These backup nodes are essentially in random positions.

TABLE II
Notations and Descriptions

Notation	Description
n	Number of miners who provide storage in the system
k	Amount of data backup assigned to different miners
d	Real amount of data stored in the system
r	Number of miners that occur failures simultaneously in a short interval time
s_r	Amount of data loss that lose all backups when incurring r miners failure (less than d)

In the proposed scheme, we let l_r denote the ratio of data loss with r random node failures, i.e., $l_r = s_r/d$, where d is the current number of stored data and s_r is the number of data that lost all backup when r nodes incur failure. In our data allocation algorithm, n nodes split the hash ring into n adjacent segments, where consecutive k successors have the data backups located at the segment. The data loss means that a segment fails. Let p be the probability of losing one single segment. Then, we use permutations and combinations, which are denoted as A_n^r and C_n^r, respectively, to compute the probability. Recall that we have $A_n^r = (n!/[(n-r)!])$ and $C_n^r = (n!/[r!(n-r)!])$. Then, probability p that a segment fails can be computed as follows:

$$\begin{aligned} p &= \frac{C_{n-k}^{r-k}}{C_n^r} = \frac{r! \cdot A_{n-k}^{n-r}}{(n-r)! \cdot A_n^r} = \frac{r! \cdot (n-k)!}{n! \cdot (r-k)!} \\ &= \frac{r(r-1)(r-2)\cdots(r-k+1)}{n(n-1)(n-2)\cdots(n-k+1)} \\ &\approx \left(\frac{r}{n}\right)^k. \end{aligned} \tag{1}$$

Let X be the number of failed segments in the hash ring containing n segments, which is a random variable. The hash ring contains n segments where each of them has the same fail probability p. Ignoring the tiny correlation between adjacent segments, X satisfies binomial distribution, i.e., $\mathrm{X} \sim B(n, p)$. We can compute the expectation $E(\mathrm{X}) = np$. This implies that the system might lose np segments on average. Now, we assume that the system has n nodes and d data mapping to the hash ring, which rises a similar distribution between the two identities after adjustment in nodes. To compute the ratio of data loss l_r, we roughly consider that data are distributed uniformly over the segments split by nodes because of the similar distribution, and each segment contains data of proportion d/n. According to the expectation of segments failure, the expected amount of data loss is $s_r = np \cdot d/n = pd$. Thus, we can get the ratio of data loss with random fault as follows:

$$l_r = \frac{s_r}{d} = p \approx \left(\frac{r}{n}\right)^k. \tag{2}$$

The existing schemes have at least one single dealer or miner to allocate data (e.g., Permacoin or Filecoin). This party assigns data to specific locations for balanced distribution. Similarly, we consider the situation of simultaneously nodes

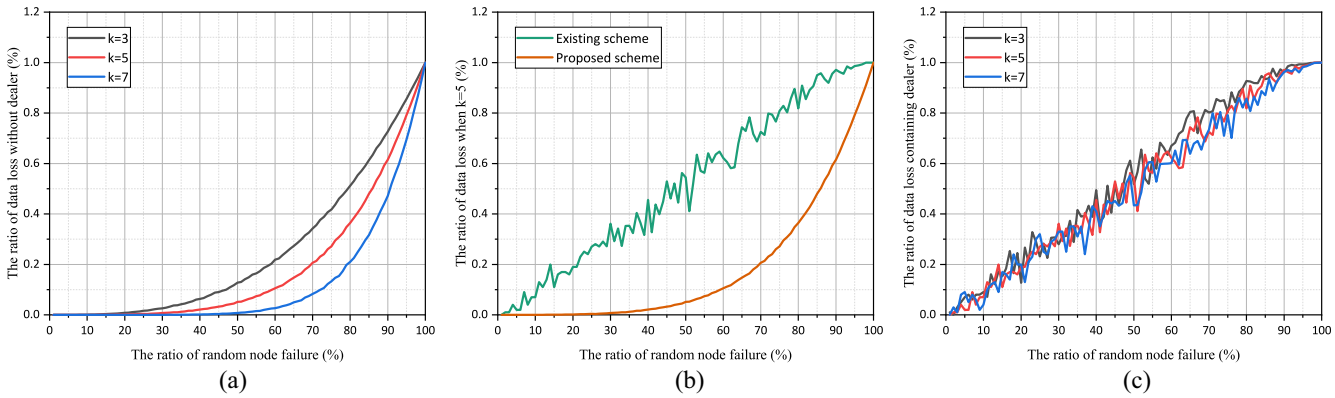

Fig. 4. Experiment results for evaluating system reliability. (a) Data loss in the proposed allocation. (b) Comparison of data loss. (c) Data loss in the existing allocation.

failure. The data loss happens only if all the k specific nodes encounter failure. We treat $\sigma = r/n$ as the probability of one node failure. In the dealer manipulated data allocation, we assume that the dealer has the same probability of node failure σ. Let l'_σ denote the probability of data loss with random node failures, and the dealer assigns data to arbitrary k nodes. The probability of data loss includes two parts below. First, the dealer failure leads to data loss. Second, if the dealer does not encounter failure, only all the specific nodes failed causes data loss. Thus, we have the probability as follows:

$$l'_\sigma = \sigma + (1-\sigma) * \sigma^k = \sigma^k + \sigma - \sigma^{k+1}. \tag{3}$$

Due to the existence of dealer role, the probability of data loss increases a lot, that is, $l'_\sigma - l_r = \sigma - \sigma^{k+1}$. Compared to it, our data allocation algorithm gets higher reliability.

C. Simulation Verification

We implement the simulation in Java 1.8 on a Desktop PC that runs Windows 10 with Intel Core i5-8250U CPU and 8-GB RAM. In the simulation, we fix the number of miners $n = 1000$ and the real amount of stored data $d = 5000$. The failed miners are randomly selected to simulate the random failure, setting the number r from 0 to n. We make a statistic from the union of the data stored in these miners.

First, we evaluate the ratio of data loss in the hash ring-based data allocation. Let the amount of data backup $k = 3, 5, 7$, getting the average results from 100 experiments. As shown in Fig. 4(a), the larger the value of k, the more concave the curve that represents the ratio of data loss l_r is. The experimental results imply that we can reduce the risk of data loss by increasing the amount of data backup. It matches the theoretical analysis (2) that the ratio of data loss is positively correlated with the ratio of random node failure.

Second, we adopt the same system parameter to evaluate the ratio of data loss in the dealer manipulated data allocation. Similarly, we let the amount of data backup $k = 3, 5, 7$, and get the average results from 100 experiments. As shown in Fig. 4(c), the ratio of data loss l_σ represents approximately linear growth with respect to the ratio of random node failure. From the experimental results, the amount of data backup has a very limited impact on the risk of data loss. It matches the theoretical analysis (3) that σ takes primary effect.

Fig. 4(b) makes a comparison between the dealer manipulated and hash ring-based data allocation, where we choose the results from which the amount of data backup $k = 5$. The proposed system performs better in any random failure. We also can increase the amount of data backup to get higher reliability. However, a large value of k not only increases the storage cost for data owners but also shrinks the total storage capacity of the system. To tradeoff the risk and cost, the system needs to choose an appropriate parameter according to a practical scenario.

D. Practical Suggestions

We introduce the Poisson distribution to simulate random process of node failure. Let Y denote the random event that is subject to $P(\mathrm{Y} = i) = ([e^{-\lambda}\lambda^i]/i!)$. In this equation, λ denotes the average number of node failures during a unit time and e denotes the natural logarithm. We multiply the probability of i nodes failure with the ratio of data loss l_i on average under the same case. Then, we can compute the probability of data loss by summing all possible situations. Let L denote this probability, we have the equation as follows:

$$L = \sum_{i=0}^{n} \frac{e^{-\lambda}\lambda^i}{i!} \cdot l_i \approx \sum_{i=0}^{n} \frac{e^{-\lambda}\lambda^i}{i!} \cdot \left(\frac{i}{n}\right)^k. \tag{4}$$

We use $1 - L$ to evaluate system reliability, the closer the value to 1, the more reliable the system is. The equation is related to three parameters of n, k, and λ, where the first two values have a positive effect on reliability while the last one makes the opposite effect. Changes in the storage network and the ranges of the average failures are less important to system reliability. Table III shows partial values about the probability of data loss as a reference. The results imply that the system should increase data backup when failures raising or miners leave the storage network.

We take the Bitcoin network as an example to compute the value of the system in reality. As of April 1st in 2019, the number of Bitcoin nodes has reached 10 000. According to the historical statistic provided by coin.dance website [30], the maximum ratio of miner changes per day does not exceed

TABLE III
REFERENCE OF L WITH PARTIAL SYSTEM PARAMETERS

λ		k=3	k=4	k=5	k=6	k=7
n=100	10	0.0013	1.67e-4	2.27e-5	3.24e-6	4.88e-7
	50	0.1325	0.0702	0.0378	0.0207	0.0116
	100	0.4208	0.3935	0.3691	0.3473	0.3277
n=1000	100	0.001	1.06e-4	1.10e-5	1.16e-6	1.22e-7
	200	0.0081	0.0016	3.36e-4	6.89e-5	1.42e-5
	500	0.1258	0.0633	0.0319	0.0161	0.0081
n=10000	500	1.26e-4	6.33e-6	3.19e-7	1.61e-8	8.2e-10
	1000	0.001	1.01e-4	**1.01e-5**	1.02e-6	1.02e-7
	2000	0.008	0.0016	3.22e-5	6.45e-5	1.29e-5

approximately 10%. We broaden the interval time for fault adjustment in the system to one day (usually no more than a few hours in a real situation). The parameters in the Bitcoin network are set to $n \approx 10000$ and $\lambda \approx 1000$. If the system wants to reach the reliability of five nines, i.e., 99.999%, we can suggest that the amount of data backup is set to five.

VII. DISCUSSION

The system we built on blockchain realizes reliable data storage, making full use of miner's extra storage space. Below, we present some additional benefits our system brings.

Collaborative Storage Service: The mining reward is the main source of profit for miners. To get more profits, miners can join the storage network to increase the opportunities for successful mining. In our system, we do not build a client-to-miner storage market, and thus, there is no competition relationship among miners to fetch data from users. The miner's revenue is related to storage contribution, which is determined by the blockchain's state. Due to the ring structure applied to the data distribution, a miner who excludes others from the storage network might increase the assigned data, which eventually harms himself because of massive storage overhead. For rational miners, the best choice is sharing data rather than obstructing others from getting data because of no gains to do this. Thus, users can parallelly download raw data from multiple miners.

Resource Exhaustive Attack Prevention: Akin to PoS, the proposed consensus uses the global state from blockchain to adjust the target hash value. Kanjalkar *et al.* [31] found resource exhaustion attacks in PoS-based consensus, where the main reason is that the current state consists of the whole blockchain from the beginning. To check a state (e.g., coinage in Peercoin [3]), miners traverse the blockchain backward for inspecting all possible transactions. Thus, miners have to temporarily store a received new block in memory before confirming it. Adversaries exploit this vulnerability to fill up a miner's memory with fake blocks to exhaust storage resources, causing the miner cannot to deal with new blocks. Fortunately, our system simply tracks recent blocks, which is defined by the maximum interval blocks, so that prevents such an attack.

Fast Healing Network: In the proposed consensus, miners can dynamically alter the node position on the hash ring to accommodate the data distribution. Considering an extreme situation, all data are mapped into the right-hand side of the hash ring. If a miner chooses to work on the right-hand side, the miner might be assigned massive data. We call this forward strategy. Conversely, the miner works on the left-hand side, called the backward strategy, to be assigned few data. When all miners adopt the backward strategy, they will be crowded on the left-hand side. Due to the ring structure, the miners around the margin of the crowd face unbearable storage overhead. Thus, the miners always have to select a new identity to join in the storage network, and frequent identity changes take no benefits for the accumulative workload. The amount of data backup can encourage miners to adopt the forward strategy, which leads to a similar distribution between miners and data on the hash ring. Thus, the system naturally enhances stability and reaches load balancing.

VIII. CONCLUSION

In this article, we proposed a variant Nakamoto consensus that uses PoCW to not only reduce the waste of computing power but also provide a reliable decentralized storage service. Besides, we utilize the blockchain's state to design a hash ring-based data allocation algorithm to assign data for miners. The theoretical analysis shows that our scheme reaches higher reliability than existing storage systems, and the simulation experiments match the analysis. Moreover, we use Poisson distribution to simulate random node failure and evaluate the probability of data loss.

REFERENCES

[1] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "SOK: Research perspectives and challenges for bitcoin and cryptocurrencies," in *Proc. IEEE Symp. Security Privacy*, 2015, pp. 104–121.

[2] S. Nakamoto. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[3] S. King and S. Nadal, "PPcoin: Peer-to-peer crypto-currency with proof-of-stake," Aug. 2012.

[4] J. R. Douceur, "The sybil attack," in *Proc. Int. Workshop Peer-to-Peer Syst.*, 2002, pp. 251–260.

[5] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Cham, Switzerland: Springer, 2017, pp. 297–315.

[6] A. Juels and B. S. Kaliski, Jr., "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 584–597.

[7] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 598–609.

[8] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *Proc. IEEE Symp. Security Privacy*, 2014, pp. 475–490.

[9] P. Labs. (2017). *Filecoin: A Decentralized Storage Network*. [Online]. Available: https://filecoin.io/filecoin.pdf

[10] S. Wang *et al.*, "ForkBase: An efficient storage engine for blockchain and forkable applications," in *Proc. VLDB Endow.*, vol. 11, no. 10, pp. 1137–1150, 2018.

[11] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale Internet of Things data storage and protection," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 762–771, Sep./Oct. 2019.

[12] Q. Xu, K. M. M. Aung, Y. Zhu, and K. L. Yong, "A blockchain-based storage system for data analytics in the Internet of Things," in *New Advances in the Internet of Things*. Cham, Switzerland: Springer, 2018, pp. 119–138.

[13] L. Zhou, L. Wang, and Y. Sun, "MIStore: A blockchain-based medical insurance storage system," *J. Med. Syst.*, vol. 42, no. 8, p. 149, 2018.

[14] W. Wang, N. Hu, and X. Liu, "BlockZone: A blockchain-based DNS storage and retrieval scheme," in *Proc. Int. Conf. Artif. Intell. Security*, 2019, pp. 155–166.

[15] Y. Chen, S. Ding, Z. Xu, H. Zheng, and S. Yang, "Blockchain-based medical records secure storage and medical service framework," *J. Med. Syst.*, vol. 43, no. 1, p. 5, 2019.

[16] P. Snow, B. Deery, J. Lu, D. Johnston, and P. Kirby. (2014) *Factom: Business Processes Secured by Immutable Audit Trails on the Blockchain*. [Online]. Available: https://www.factom.com/

[17] G. Zyskind, O. Nathan, and A. Pentland, "ENIGMA: Decentralized computation platform with guaranteed privacy," 2015. [Online]. Available: arXiv:1506.03471.

[18] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of namecoin and lessons for decentralized namespace design," in *Proc. WEIS*, 2015, p. 22.

[19] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "BlockStack: A global naming and storage system secured by blockchains," in *Proc. USENIX Annu. Techn. Conf.*, 2016, pp. 181–194.

[20] J. Li, J. Wu, and L. Chen, "Block-secure: Blockchain based scheme for secure P2P cloud storage," *Inf. Sci.*, vol. 465, pp. 219–231, Oct. 2018.

[21] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin. (2014). *Storj a Peer-to-Peer Cloud Storage Network*. [Online]. Available: http://storj.io/storj.pdf

[22] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XoR metric," in *Proc. Int. Workshop Peer-to-Peer Syst.*, 2002, pp. 53–65.

[23] H. Kopp, D. Mödinger, F. Hauck, F. Kargl, and C. Bösch, "Design of a privacy-preserving decentralized file storage with financial incentives," in *Proc. IEEE Eur. Symp. Security Privacy Workshops*, 2017, pp. 14–22.

[24] H. Kopp, C. Bösch, and F. Kargl, "Koppercoin—A distributed file storage with financial incentives," in *Proc. Int. Conf. Inf. Security Practice Exp.*, 2016, pp. 79–93.

[25] G. Ateniese, L. Chen, M. Etemad, and Q. Tang, "Proof of storage-time: Efficiently checking continuous data availability," in *Proc. NDSS*, 2020, p. 2.

[26] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide Web," in *Proc. 29th Annu. ACM Symp. Theory Comput.*, vol. 97, 1997, pp. 654–663.

[27] I. Stoica *et al.*, "Chord: A scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[28] A. Rowstron and P. Druschel, "PASTRY: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM Int. Conf. Distrib. Syst. Platforms Open Distrib. Process.*, 2001, pp. 329–350.

[29] R. Abe, "Blockchain storage load balancing among dht clustered nodes," 2019. [Online]. Available: arXiv:1902.02174.

[30] D. Coin. (2019). *Bitcoin Nodes Summary*. [Online]. Available: https://coin.dance/nodes/all

[31] S. Kanjalkar, J. Kuo, Y. Li, and A. Miller, "Short paper: I can't believe it's not stake! resource exhaustion attacks on PoS," in *Proc. Int. Conf. Financ. Cryptography Data Security*, 2019, pp. 62–69.

Hao Yin received the B.E. and M.S. degrees in information security and software engineering from the University of Science and Technology Beijing, Beijing, China, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing.

He is also joint studied with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include applied cryptography, security and privacy, and blockchain.

Zijian Zhang received the Ph.D. degree from Beijing Institute of Technology, Beijing, China, in 2012.

He is an Associate Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He is also a Research Fellow with the School of Computer Science, University of Auckland, Auckland, New Zealand. He was a Visiting Scholar with the Computer Science and Engineering Department, The State University of New York at Buffalo, Buffalo, NY, USA, in 2015. His research interests include design of authentication and key agreement protocol and analysis of entity behavior and preference.

Jialing He received the B.E. and M.S. degrees from Beijing Institute of Technology, Beijing, China, in 2016 and 2018, respectively, where she is currently pursuing the Ph.D degree with the School of Cyberspace Science and Technology.

Her research interests include cloud security, data privacy, and blockchain.

Liran Ma (Member, IEEE) received the Ph.D. degree in computer science from George Washington University, Washington, DC, USA, in 2008.

He is a Professor with the Department of Computer Science, Texas Christian University, Fort Worth, TX, USA. His current research focuses on wireless, mobile, and embedded systems, including security and privacy, smart devices and health, mobile computing, data analytics, Internet of Things, artificial intelligence, and cybersecurity education. It involves building and simulating prototype systems and conducting real experiments and measurements.

Liehuang Zhu (Member, IEEE) received the Ph.D. degree from Beijing Institute of Technology, Beijing, China, in 2004.

He is a Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He is selected into the Program for New Century Excellent Talents in University from Ministry of Education, Beijing. His research interests include cryptographic algorithms and secure protocols, Internet of Things security, cloud computing security, big data privacy, mobile and Internet security, and trusted computing.

Meng Li (Member, IEEE) received the B.E. degree in information security from Hefei University of Technology, Hefei, China, in 2010, and the M.S. and Ph.D. degrees in computer science and technology from Beijing Institute of Technology, Beijing, China, in 2013 and 2019, respectively.

He is currently an Associate Researcher with the School of Computer Science and Information Engineering, Hefei University of Technology. He was sponsored by the China Scholarship Council to study as a visiting Ph.D. student with the Broadband Communications Research (BBCR) Lab, University of Waterloo, Waterloo, ON, Canada, and Wilfrid Laurier University, Waterloo, from September 2017 to August 2018. His research interests include applied cryptography, security and privacy, vehicular networks, fog computing, and blockchain.

Bakh Khoussainov received the Ph.D. degree from the Algebra and Logic Department, Novosibirsk University, Novosibirsk, Russia, in 1987.

He is currently a Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His research interests include computable algebraic systems and model theory, automata and automatic structures, games on finite graphs and complexity, abstract data types and algebraic specifications, and computably enumerable reals and randomness.

Prof. Khoussainov is an Editor of the *Journal for Symbolic Logic*.