Hindawi
Security and Communication Networks
Volume 2023, Article ID 3412200, 14 pages
https://doi.org/10.1155/2023/3412200

Research Article

HLOChain: A Hierarchical Blockchain Framework with Lightweight Consensus and Optimized Storage for IoT

Qingqing Xie,[1] **Fan Dong**,[1] **and Xia Feng**[2]

[1]*School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang 212013, China*
[2]*School of Automotive and Traffic Engineering, Jiangsu University, Zhenjiang 212013, China*

Correspondence should be addressed to Qingqing Xie; xieqq@ujs.edu.cn

Received 6 January 2023; Revised 4 February 2023; Accepted 20 February 2023; Published 11 April 2023

Academic Editor: Jie Cui

Blockchain can effectively deal with the security and trust issues in Internet of Things (IoT) due to its salient features including decentralization, immutability, traceability, openness, and transparency. However, most IoT devices have too limited computing, storage, and bandwidth resources to maintain the complete operation of a blockchain system. To this end, we propose a hierarchical blockchain framework called HLOChain for IoT scenarios. First, according to computing and storage capabilities, the IoT devices are classified into three levels, i.e., high, medium, and low. They are deployed on different layers. In this way, a hierarchical blockchain architecture is designed. Second, we propose a lightweight proof of random (PoR) consensus mechanism to provide low-energy block mining, so that even the medium nodes can participate in the consensus task. Third, in order to reduce the ledger storage overhead, we design a blockchain storage optimization strategy based on the account model. Finally, the security analysis demonstrates that our HLOChain is secure against double-spend attack, Sybil attack, and so on. The experimental evaluation shows that our HLOChain achieves better performance in ledger storage cost, consensus computing cost, throughput, and transaction confirmation latency.

1. Introduction

The Internet of Things (IoT) [1] is a network where objects can connect to the Internet through information sensing devices. The objects exchange information and communicate to realize intelligent recognition, tracking, positioning, and so on. At present, the IoT is widely used in smart home, smart city, medical care, logistics, transportation, and other fields to provide users with convenient services, efficient and reliable management control. However, the IoT is a trustless environment. Considering that the mass data generated in IoT are exposed in the network, some security and privacy concerns are rising. In traditional security protocols, network nodes communicate with each other through a centralized entity. This kind of centralized protocol not only increases network delay and interaction times but is also prone to single-point failures. Furthermore, most IoT devices are resource-constrained. Most of the resources of IoT devices [2] are used to perform core functions such as collecting, processing, and transmitting data. They cannot afford the resource consumption of existing security techniques [3]. A low-energy and decentralized approach is needed to realize security and privacy preservation in the IoT.

Blockchain is a new type of distributed ledger technology that combines the P2P network, smart contract, consensus mechanism, and cryptography. Its decentralization feature makes a trust foundation come true under a trustless scenario. It can provide a new idea for solving the security and trust problems existing in the traditional service architecture. As we know, the traditional blockchain relies on computation-intensive consensus algorithms and highly redundant storage to provide transaction security. A typical application field is cryptocurrency [4]. With the development of blockchain, people try to apply the blockchain into financial economy [5], Internet of Things (IoT) [6, 7], supply chain [8], healthcare [9], and other fields. However, there are many resource-constrained devices in these application areas, such as sensors, smart home devices, and so on. They cannot afford the computation-intensive consensus algorithms and highly redundant storage of traditional blockchain.

Edge computing is decentralized and owns rich computation, storage, and communication resources. Thus, it can provide a natural solution to overcome the aforementioned resource-poor situation [10, 11]. Edge nodes with rich computing, storage, and communication capabilities can take on the role of blockchain miners. Integrated edge computing is expected to make blockchain technology widely used in resource-constrained scenarios.

Some related works on the combination of blockchain and edge computing have been carried out so far. Liu et al. [12] proposed a wireless blockchain framework supporting mobile edge computing. In this framework, the computationally intensive block mining work is undertaken by the edge nodes, and the ledger is stored on the edge servers. Chen et al. [13] proposed a collaborative and distributed computing offloading algorithm for the blockchain-driven Industrial Internet of Things (IIoT), where both data processing tasks and mining tasks are taken into consideration. Groupchain [14] is a novel double-chain structure blockchain. It improves the scalability of blockchain and fog computing integration. Lang et al. [15] proposed a blockchain-based data sharing architecture for vehicular edge computing networks. It uses the collaborative computing offloading method to offload some computing and storage tasks to edge nodes. In this way, it enables efficient data sharing between smart vehicles and service providers. Xu et al. [16] proposed a blockchain-based edge computing architecture by integrating edge computing and blockchain into the IoT. It not only realizes secure and scalable IoT transactions but also provides ample data storage space. Guo et al. [17] combined blockchain edge computing and blockchain and proposed a distributed and trusted authentication system. To improve authentication efficiency, the smart contracts are deployed on the edge nodes to provide name resolution and distribute authentication services. In these aforementioned schemes, the resource-constrained devices do not directly participate in consensus but operate as light nodes. The maintenance of the blockchain depends on the edge nodes.

In this study, we also introduce edge computing into deploying the blockchain system. The first motivation is that the edge computing is distributed, which is consistent with the blockchain's decentralized characteristic. The second motivation is that edge computing can provide abundant computation, storage, and communication resources for maintaining a blockchain system, especially for some resource-constrained IoT scenarios. Nevertheless, combining edge computing and blockchain is not straightforward. It faces the following challenge.

1.1. Challenge. The direct integration of blockchain and edge computing cannot completely solve the problem of resource-constrained devices participating in the blockchain. Those low-resource devices do not always work as light nodes. They also wish to participate in consensus as full nodes and maintain the blockchain. In addition, existing consensus mechanisms, such as PoW, PoS, and PBFT, cannot be directly applied to the IoT. In order to enable resource-constrained devices to participate in consensus, it is necessary to design a more efficient and low-energy consensus mechanism. The conflict between relieving storage pressure and protecting ledger data integrity also urgently needs to be solved.

1.2. Main Contributions. To address the previous challenges, this study proposes a hierarchical blockchain framework with lightweight consensus and optimized storage for resource-constrained scenarios. The main contributions are summarized as follows:

(1) We design a novel hierarchical system architecture, in which all the nodes are divided into three levels, i.e., high, mid, and low, according to their capability. With the support of lightweight consensus protocol and data storage optimization, our HLOChain enables mid-level nodes with limited resources to maintain the blockchain such as full nodes.

(2) In order to reduce the resource consumption of the consensus mechanism, we propose a lightweight consensus mechanism, Proof of Random. The miner node is selected in a random way rather than relying on computing power or stake. The consensus process only consumes a few resources and is friendly to resource-constrained devices.

(3) Facing the problem of ever-increasing ledger data, this study designs an account model-based storage optimization strategy combined with the hierarchical node architecture. The blockchain ledger is divided into a full ledger and a state ledger. In the case of ensuring data integrity and normal operation of the system, resource-constrained nodes can participate in the consensus by only storing the state ledger. It is needed noting that the size of the state ledger is much smaller than the full ledger.

(4) The security analysis shows that our HLOChain could resist some common blockchain attacks well (such as double-spend attack, Sybil attack, and so on.). The experimental results show that our HLOChain can achieve better performance in ledger storage cost, consensus computing cost, throughput, and transaction confirmation latency.

1.3. Organization. The rest of this paper is organized as follows: Section 2 reviews the related work. Our system architecture is introduced in Section 3. Sections 4 and 5 present our HLOChain design in detail. Section 6 analyzes the security, and Section 7 experimentally evaluates the performance. Finally, Section 8 concludes this paper.

2. Related Works

2.1. Consensus Mechanism Optimization. The computing resource consumption of the blockchain is caused mainly by the consensus mechanism. Resource-constrained devices cannot afford computing-intensive consensus mechanisms. Therefore, optimizing the consensus mechanism is an important way to achieve lightweight computing. Some scholars have improved the consensus mechanism. Li et al. [18] proposed an improved PBFT blockchain consensus mechanism for federated blockchain. In this work, the strategy of reward and punishment is utilized, and the primary node is selected by voting. It reduces the communication complexity of PBFT. Puthal et al. [19] designed a consensus protocol, proof-of-authentication, to replace the PoW consensus. It can greatly reduce computing consumption in the consensus process. However, its application scope is limited to the permissioned blockchain.

In PoEWAL [20] (Proof of Elapsed Work and Luck) consensus protocol, each miner has a fixed period to solve a cryptographic puzzle similar to PoW. This mechanism relies on the amount of work carried out by nodes within a preset period. It can reduce the computational consumption in each block iteration by adjusting the period to calculate the cryptographic puzzle. Huang et al. [21] proposed a trust-based PoW mechanism to reduce the computing power consumption of honest nodes. This mechanism introduced the node credit value which is related to the behavior of the node. The credit value of nodes will be reduced if they perform malicious behavior. The higher the credit value of a node, the lower its mining difficulty. Saad et al. [22] introduced an extended PoS protocol e-PoS, where the nodes compete to propose a new block through auction. This process does not need to run complex cryptographic algorithms. In addition, each node has the same initial auction funds to ensure a fairer competition.

Khan et al. [23] proposed a lightweight authenticated encryption-based proof-of-authentication consensus mechanism. It utilizes a lightweight hash function and a hardware sharing architecture to reduce the overhead of computing resources. Microchain [24] randomly selected some nodes to form a committee according to the stake of the node. The committee members compute the cryptographic puzzles related to their credit value to compete to propose a new block. Dorri et al. [25] exploited the overlay network to organize nodes in the form of clusters. The nodes with rich resources are selected as cluster heads. The cluster heads reach a consensus through a lightweight time-based distributed algorithm. Kara et al. [26] proposed a Proof of Chance (PoCh) consensus mechanism. Nodes become candidates by chance and randomly select a miner from the candidates. Consensus relies on the chance value of nodes rather than computing power. However, there are only a small number of candidates in each block iteration. It is vulnerable to malicious attacks and reduces the security of the blockchain.

The previous work can reduce the resource consumption of the consensus process to a certain extent, but there are still several problems as follows: (1) The application scenarios are limited. Moreover, some are more suitable for federated blockchain [18, 19]. (2) PoW problem needs to be solved. Consensus relies on computing power and is not applicable to resource-constrained devices [20, 21]. (3) The proposed optimization measures in turn bring additional resource overhead, which will increase the burden on nodes [21, 22]. (4) Resource-constrained devices do not directly participate in consensus, but instead elect resource-rich cluster heads as representatives, which may go against the wishes of some users [20, 25]. (5) There are security loopholes, and the blockchain is vulnerable to security attacks [24, 26]. The PoR consensus proposed in this study reaches consensus in a random form that does not depend on computing power. The whole process consumes very few resources. The resource-constrained devices can also participate directly. Thus, it is suitable for resource-constrained IoT scenarios.

2.2. Blockchain Storage Optimization. Reducing the blockchain data storage to alleviate the storage burden of resource-constrained devices is the key to obtaining lightweight storage. Currently, a number of solutions have been put forth by some scholars. Zhao et al. [27] developed a lightweight enhanced SPV (simplified payment verification) node named ESPV. ESPV nodes store all new blocks and reserve some old blocks according to their storage capability. Furthermore, some projects [28–30] proposed various lightweight clients. These lightweight clients only store a subset of the most recently generated blocks rather than the entire blockchain ledger.

Li et al. [18] proposed an RS erasure code-based storage optimization strategy. Based on the RS erasure code, a new block is encoded into several segments. These code segments are distributed and stored in different nodes. To recover blocks, the nodes request the missing encoded segments from other nodes. However, this method increases the computing and communication overhead of the nodes. Liu et al. [31] proposed a lightweight blockchain system for the industrial IoT. Blocks that do not contain unspent transaction output (UTXO) sets are defined as unrelated blocks. To reduce the consumption of node storage resources, an unrelated block offloading filter is utilized to unload unrelated blocks. Wang et al. [32] proposed an UTXO model-based efficient storage scheme (ESS). To reduce blockchain storage consumption, ESS divides the blocks into three types according to UTXO weight. Then, it utilizes pruning strategies to reduce the size of the blockchain ledger. Ehmke et al. [33] introduced a Proof of Property (PoP) protocol based on the account model. Each transaction includes a proof of property, which proves that the input account of a transaction has enough coins to pay for this transaction. Therefore, the nodes only need the latest block header to verify the transaction. Kim et al. [34] proposed a ledger data compression scheme. By organically combining with the consensus mechanism, the historical block data are compressed to fully improve the utilization of node storage space. Biswas et al. [35] designed a novel type of local transaction to decrease the growth rate of the blockchain ledger. However, this approach is only applicable to a specific framework. In addition, Bitcoin-NG [36] and the works in [37–39] divide the entire blockchain network into smaller committees based on the sharding protocol. Each committee runs the consensus protocol independently and in parallel. Nodes store less ledger data.

To some extent, the previous work can relieve the storage pressure of nodes, but the following issues still exist: (1) lightweight clients allow nodes to store only a small amount of data, but these lightweight clients cannot participate in the consensus process [27–30], (2) unloading and compressing historical blocks lacks protection for the integrity of ledger data [31–35], and (3) the sharding technology is used to deal with the storage problem, but the problem of cross-sharding transaction is generated [36–39]. Under the premise of preserving data integrity, our storage optimization strategy enables those resource-constrained nodes to participate in the consensus by storing only a small amount of data.

3. System Architecture

Our system architecture is shown in Figure 1. Considering the diversity of IoT devices and the different levels of resources they have, we provide three different levels of clients. The high-level client includes the blockchain full ledger. The mid-level client includes the state ledger and a historical block unloading module. The low-level client only includes block header data. When joining the blockchain network, users select appropriate clients according to their own computing and storage resources. Different clients correspond to different node levels. High-level nodes and mid-level nodes participate in consensus, similar to full nodes. Low-level nodes are similar to light nodes, submitting transactions by connecting to high-level nodes or mid-level nodes. The detailed description of each level of node is as follows:

(i) High-level nodes ($\mathbb{H}$): This layer consists of nodes with rich computational and storage resources, such as servers, high-performance computers, and so on. In our HLOChain, edge nodes are deployed on the high-level nodes layer. They are responsible for block proposal and full ledger storage. It is also our motivation of introducing edge computing into deploying the blockchain system, i.e., utilizing the rich resources provided by edge computing to maintain the operation of a blockchain system.

(ii) Mid-level nodes ($\mathbb{M}$): This layer consists of some nodes with limited computational and storage resources, such as personal notebooks, smartphones, and tablets. These nodes are unable to run computation-intensive tasks for a long time and store hundreds of gigabytes of data. Our HLOChain optimizes the working mechanism of the consensus and the storage, so that mid-level nodes can participate in the consensus process even if their resources are limited.

(iii) Low-level nodes ($\mathbb{L}$): This layer consists of nodes that are severely resource-constrained, such as sensors, webcams, and smart home devices. These nodes cannot afford tasks other than normal operations. They propose transactions by connecting to high-level or mid-level nodes, which only need to store the block headers.

In the consensus module, inspired by [20, 26], we design a green and low-energy consumed consensus protocol, Proof of Random (PoR). The node decides whether it can become a candidate node according to the random value. According to a special comparison process, the only miner in consensus iteration is determined from the candidate nodes. The whole process consumes very little resources and is suitable for resource-constrained devices. The protocol is a strongly consistent consensus protocol, and there is no fork problem.

In terms of data storage, we learned from Ethereum's account model and optimized the block structure. The blockchain ledger is divided into a full ledger and a state ledger. The full ledger stores all data from the genesis block to the current latest block, including all transactions and all account states. The state ledger stores only the blocks containing the latest state tree. Note that the state ledger contains no any transaction data. As we all know, of all data storage, transaction data storage cost accounts for the highest proportion. Therefore, compared with the full ledger, the size of the state ledger is much smaller. The mid-level nodes only need to store the state ledger to participate in the consensus. In addition, the mid-level client also provides historical block unloading service, which is called when the storage space of the mid-level node is insufficient. The high-level nodes are encouraged to store the full ledger by increasing the block reward. They need to provide the proof of storing the full ledger to obtain the corresponding block reward.

Some major notations used in this paper are shown in Table 1. The following will introduce our lightweight consensus protocol PoR and account model-based storage optimization strategy.

4. Proof of Random

Deploying blockchains in resource-constrained IoT scenarios requires avoiding computationally intensive consensus. PBFT consensus will increase the communication burden of IoT devices. PoR is based on the concept of "Random," meaning that the node must have double luck to get block rewards. Nodes reach consensus through a two-stage random process. The first stage is the candidate test. The node calculates a hash value P with the random value R_{val} and ID, in which $P = \mathrm{Hash}\,(R_{\mathrm{val}} \| ID_i)$. If the hash value meets the given conditions (i.e., $<$ Target), it becomes a candidate node. The candidate node generates a Proof associated with its ID and broadcasts it to the blockchain. R_{val} and Target will be modified at each consensus iteration according to equations (2) and (3), respectively. Proof generation refers to part (1) in Section 4.1. The second stage is to compare the P in the Proof of all candidate nodes, and the candidate node with the smallest P becomes the miner of the current consensus iteration.

In order to facilitate the verification of candidate nodes, each candidate node includes its public key in the broadcast proof and signs it with its private key. It takes a certain time interval (t_1 in Algorithm 1) before the candidate nodes get Proof s from other candidate nodes. The t_1 is based on the

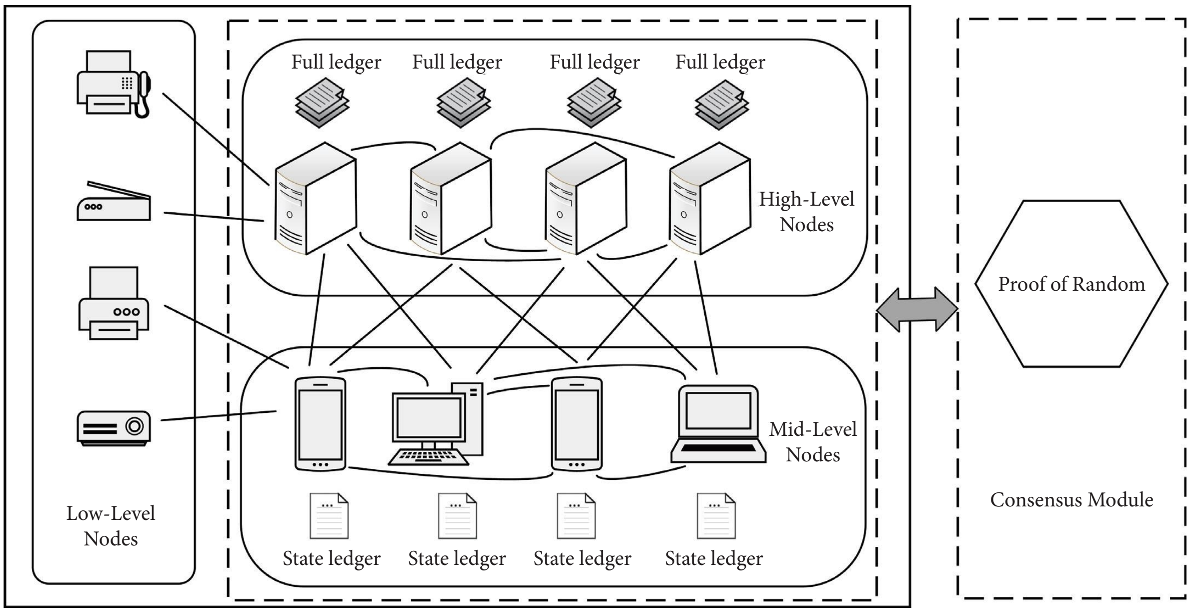

FIGURE 1: The system architecture.

TABLE 1: Some major notations used in this paper.

Notations	Description
Node_i	The i-th node
ID_i	The identity of node_i
$i_{\text{Level}} \in \{\mathbb{H}, \mathbb{M}, \mathbb{L}\}$	The role level identifier of node_i, where $\mathbb{H}$, $\mathbb{M}$, and $\mathbb{L}$ represent high-level, mid-level, and low-level nodes, respectively
$(\text{PK}_i, \text{SK}_i)$	The public and private key pair of node_i
BH	The block height
H_{pre}	The hash of previous block (the last validated block)
N_{cand}	The number of candidate nodes in the current consensus iteration
R_{val}	Random value $R_{\text{val}} = \text{Hash}(\text{BH}\|H_{\text{pre}}\|N_{\text{cand}})$
W	The maximum number of candidate nodes that the network load can tolerate
$\text{Sign}(\bullet)$	A signature algorithm
B_i	The i-th block
$B_i.h$	The i-th block header
$B_i.s$	The state body of i-th block
$B_i.t$	The transaction body of i-th block
$\text{Acct}_j.\text{ver}$	The version number of account j

present state of the network. The nodes that do not follow the rules will be prevented from taking part in consensus. After t_1, if the number of candidate nodes (N_{cand}) is in the range $[W/2 - W]$, candidate nodes normally execute the second stage consensus to determine the final miner. Otherwise, the message of consensus failure is broadcast. If the miner has been elected ($W/2 \leq N_{\text{cand}} \leq W$), all candidate nodes will get a new block generated by the miner after a time interval (t_2 in Algorithm 1).

Once receiving a new block, the candidate node updates R_{val} and Target and broadcasts the triplet (R_{val}, Target, and Block). If the block verification fails, the miner node will be reselected. Similarly, the candidate node broadcasts "consensus failure, new Target" if $N_{\text{cand}} \notin [W/2 - W]$. If a node is not a candidate, it must wait for a new (R_{val}, Target, and Block) or the message of consensus failure. In order to reduce the probability of consensus failure, only candidate nodes have the right to determine the state of the current consensus. A decision is considered valid only when it is from at least half of the candidate nodes. Algorithm 1 depicts the suggested consensus process.

According to the synchronous consensuses, such as PoW, the system state is updated upon the completion of every consensus round. Nevertheless, there is no clear upper restriction for message delivery in asynchronous consensuses, such as DPoS. To address deadlocks, partially synchronous consensuses, such as PBFT, can use timeouts, preset hierarchies, and additional rules [40]. In our PoR, the nodes cannot forward to the next consensus round until a new block or consensus failure message is broadcast.

```
Require: params: BH; $R_{\text{Val}}$; Target; $t_1$, $t_2$: time interval
Ensure: consensus
(1) procedure
(2)   $P \leftarrow \text{Hash}(R_{\text{Val}} \| ID_i)$
(3)   if ($P <$ Target) then
(4)     generate Proof ($P, \text{ID}_i, \text{PK}_i, \text{Sign}(R_{\text{Val}})$)
(5)     broadcast Proof
(6)     waiting for $t_1$, and:
(7)     receive Proof (incrementing $N_{\text{cand}}$)
(8)     if ($N_{\text{cand}} < W/2 \| N_{\text{cand}} > W$) then
(9)       update (Target)
(10)      broadcast "consensus failure, new Target"
(11)      go to (line 2)
(12)    else
(13)      select the luck Proof: the $P$ of Proof is minimum
(14)      if (my Proof is the luck) then
(15)        generate a new block
(16)        update block height: BH$\leftarrow$BH + 1
(17)        update $R_{\text{Val}} \leftarrow \text{Hash}(\text{BH} \| H_{\text{pre}} \| N_{\text{cand}})$
(18)        update Target$\leftarrow$currentTarget/$\log_{3W/4} N_{\text{cand}}$
(19)        broadcast new ($R_{\text{Val}}$, Target, Block)
(20)        go to (line 2)
(21)      else
(22)        waiting for $t_2$, and:
(23)        if (the new Block has arrived) then
(24)          verify it
(25)          if (the Block passes validation) then
(26)            update Blockchain
(27)            update ($R_{\text{Val}}$, Target)
(28)            broadcast new ($R_{\text{Val}}$, Target, Block)
(29)            go to (line 2)
(30)          end if
(31)        else
(32)            update (Target)
(33)            broadcast "consensus failure, new Target"
(34)            go to (line 2)
(35)        end if
(36)      end if
(37)    end if
(38) else
(39)   wait for the new ($R_{\text{Val}}$, Target, Block) or (consensus failure, new Target), go to (line 2)
(40) end if
(41) end procedure
```

Algorithm 1: Proof of random.

4.1. PoR Main Operation Principles.

The main steps defined in our PoR are explained as follows:

4.1.1. *Generating* Proof

$$P = \text{Hash}\left(R_{\text{val}} \middle\| \text{ID}_i\right). \quad (1)$$

The node calculates the hash value P by equation (1). If $P <$ Target, the node becomes a candidate node and generates a Proof $(P, \text{ID}_i, \text{PK}_i, \text{Sign}(R_{\text{Val}}))$. The generation of ID is very important here, which needs to ensure uniqueness, unpredictability, and verifiability. The uniqueness guarantees the uniqueness of P and ensures that only one miner is generated in the second stage of the consensus. Thus, the fork problem can be avoided. Unpredictability is the guarantee of system security. Verifiability is to ensure the legitimacy of ID generation and prevent nodes from maliciously forging ID to skip candidate tests. We give the calculation principle of ID, which is $\text{ID}_i = \text{Hash}(\text{Sign}(R_{\text{Val}}))$. $\text{Sign}(R_{\text{Val}})$ is the signature of Node_i to the random value R_{Val}, which is unique, unforgeable, and verifiable. In each consensus interation, the values of R_{val} and Target will be updated according to equations (2) and (3). Therefore, in each consensus iteration, the ID is changing and unpredictable. In addition, in order to prevent malicious nodes from forging multiple ID through multiple public and private keys to increase their probability of passing the candidate test, nodes need to pledge a certain number of tokens when generating Proof.

4.1.2. R_{Val} Update. In equation (2), BH, H_{pre}, and N_{cand} are the block height, the hash of the previous block, and the number of candidate nodes, respectively. Therefore, the random value R_{Val} is computed by the concatenation of these three parameters. The block height can ensure that each update process is uniquely identified when the other two parameters stay unchanged. The H_{pre} is used because it cannot be manipulated by the miner of the current consensus. The N_{cand} is used in the update procedure to prevent malicious nodes from manipulating their blocks to pass the next candidate test.

$$R_{\mathrm{Val}} = \mathrm{Hash}\left(\mathrm{BH}\|H_{pre}\|N_{\mathrm{cand}}\right). \tag{2}$$

4.1.3. Target *Update*

$$\mathrm{Target} = \frac{\mathrm{currentTarget}}{\log_{3W/4} N_{\mathrm{cand}}}. \tag{3}$$

After each round of consensus, we use equation (3) to update N_{cand}. If $N_{\mathrm{cand}} > 3W/4$, the Target value will be reduced after the update. The difficulty of the next round of the consensus candidate test will increase. As a consequence, fewer nodes can pass the candidate test. If $N_{\mathrm{cand}} < 3W/4$, the Target value will be increased after the update. The difficulty of the next round of the consensus candidate test will decrease. More nodes can pass the candidate test. Thus, N_{cand} remains near $3W/4$, i.e., in the range $[W/2 - W]$. If N_{cand} is equal to 0 or 1, then $N_{\mathrm{cand}} \leftarrow 2$. Here, W is dependent on the Proof size and the network throughput, i.e., the propagation of W Proof in the P2P network will not cause network congestion.

4.1.4. Miner Select. After t_1, nodes will receive the Proof of all candidate nodes that pass the candidate test. These Proof form a candidate proof list. After that, one of these candidate nodes needs to be selected as a miner node. The specific measure is that the candidate node sorts the hash value P in the proof list and selects the Proof with the smallest P. If the Proof passes the verification, it is considered that the node that generated the Proof becomes the miner of the current consensus iteration. Because the P generated by equation (1) is unique, a unique miner will be determined in this process. The miner will construct a new block and broadcast it to other candidate nodes for verification.

4.1.5. Verification. This process includes Proof verification, block verification, and transaction verification. The verification work is the responsibility of the candidate node. Because our HLOChain is based on the account model, transaction verification mainly depends on the account state of the node. For Proof verification, first verify the legitimacy of the signature according to the node public key, then verify whether the ID generation is legal according to the signature, and finally verify the legitimacy of the hash value P according to the ID and R_{val}. The verification of the block includes the following point:

(1) To verify the Proof of the miner node, the candidate node is compared with the Proof selected in the second stage of consensus with the Proof contained in the block. The signature is verified with the public key in the Proof.

(2) If the miner is a high-level node, it needs to include a digest of the full ledger in the block to prove that it has stored the full ledger. The verification of the digest depends on the high-level nodes. Therefore, the mid-level nodes among the candidate nodes need to randomly send requests for verification digest to 10 high-level nodes. When more than half of the verification success messages are received, the digest is considered to be legal.

(3) All transactions are verified and executed according to the account state. If any transaction is illegal, the block will not pass the verification.

After the candidate node is successfully verified, the ledger is updated and the new block is broadcast to the noncandidate nodes. If the verification fails, a consensus failure message will be broadcast.

4.2. PoR Additional Rules. A fork means that there are multiple different versions of the blockchain simultaneously. It seriously affects the scalability of the blockchain. To reduce the probability of forks, we make the following additional rules:

(1) The candidate node is required to wait t_1 before selecting the miner. The t_1 depends on the network conditions. It is guaranteed that the Proof of all candidate nodes can be spread throughout the network during this period.

(2) The noncandidate node of each consensus iteration must wait until a new block or a consensus failure message is received

(3) Each candidate node is required to forward the received Proof data to all candidate nodes

(4) Each noncandidate node is required to forward the received Proof data to 10 randomly nodes

(5) If a new candidate node for the previous block consensus iteration (i.e., block $k - 1$) appears in the block k consensus iteration, and the block k's mining work must be aborted. Nodes must reselect the miner based on the proof list of this candidate node.

(6) If a new candidate node for the block $k - 2$ consensus iteration appears in the block k consensus iteration, this new candidate node should be disregarded

(7) If there are two candidate proof lists U and U', where $U \subset U'$, only the U' is taken into consideration by the network. Otherwise, if $U \not\subset U'$ and $U' \not\subset U$, both U and

$U^{'}$ are disregarded. Simultaneously, consensus failure message is broadcasted.

5. Storage Optimization Strategy

We optimize data storage based on the Ethereum account model. Each node has a unique account, and the verification of its transactions mainly depends on the account state of the node. The structure of the state tree is a Merkle Patricia Tree (MPT). The leaf nodes store the account states. Once a new block is generated, the states of the accounts involved will change. The corresponding nodes in the state tree will change as well. Note that the change is not made directly in the original nodes. Instead, some new branches are created to store the changed accounts. As shown in Figure 2, we will take the genesis block and block 1 as an example to show the state tree update process. Assume that the block 1 only changes the account 7. Thus, a new branch marked by the blue boxes is created to store the new state of the account 7. Other unchanged nodes directly point to the counterparts in the previous block. In this way, the state tree is updated, and a new MPT is created. So, there is only one latest state tree in the blockchain. To participate in mining and verify transactions, nodes need to keep a full ledger.

In order to solve the storage pressure of resource-constrained devices, so that they can participate in the consensus by only storing a small amount of data, we have optimized the data layer. First, we optimize the block structure, as shown in Figure 3. A complete block B_i includes block header $B_i.h$, state body $B_i.s$, and transaction body $B_i.t$, i.e., $B_i = \{B_i.h, B_i.s, B_i.t\}$. The block header contains some common parameters such as block height BH, timestamp TS, hash of the previous block H_{pre}, hash of the root of the state tree SR, hash of the root of the transaction tree TR, Proof of miner, role level identifier i_{level}, and full ledger digest Dig, i.e., $B_i.h = \{\text{BH}, \text{TS}, H_{\text{pre}}, \text{SR}, \text{TR}, \text{Proof}, i_{\text{Level}}, \text{Dig}\}$. The state body contains a list of accounts whose current state has changed, i.e., $B_i.s = \{\text{Acct\_list}\}$. The transaction body is the list of transactions in the current block, i.e., $B_i.t = \{Tx\_\text{list}\}$. We separate the account state from the block, and nodes only need to store the block header and state body to participate in the consensus. Therefore, the transaction body can be offloaded when the storage space is insufficient without affecting the normal operation of the blockchain.

In addition, we found that the state of accounts in the blockchain is constantly changing, and the historical state of those accounts has no effect on subsequent transaction verification. Therefore, they can be deleted without affecting the normal operation of the blockchain to save space. Those blocks that do not contain the latest account state are defined as useless blocks. Add a version number flag $\text{Acct}_j.\text{ver}$ to the account, which corresponds to the block height of the block storing the current account, i.e., $B_i.\text{BH} = \text{Acct}_j.\text{ver}$. Therefore, we only need to find the smallest version number of all accounts from the latest state tree, and the historical blocks before the block height corresponding to this version number are all "useless blocks." So, they can be deleted. Here, we consider all accounts to be active because IoT devices are collecting, processing, and exchanging data all the time. These actions will be recorded in the blockchain in the form of transactions.

Finally, we designed a historical block offloading algorithm for mid-level nodes. As shown in Algorithm 2, our purpose is to delete those useless blocks and transaction bodies. First, obtain the latest account list from the state tree of the latest block and find the minimum version number of the account from the account list. Then, delete those historical blocks whose block height is less than the minimum version number. Finally, delete the transaction body of the remaining blocks. The rest is the state body that only contains the block header and the latest account. We call this part of the data the state ledger.

In this study, edge nodes with strong computational power and large storage space are deployed at the high-level nodes layer, which need to store the full ledger to maintain the integrity of blockchain ledger. The mid-level nodes whose storage space is limited just need to store the state ledger to participate in consensus. The state ledger only occupies a small amount of storage space, and it does not strictly increase with the increase of the block height. The new block contains the new state of the account. The historical blocks that store the expired account state will eventually be unloaded. Therefore, as time goes by, the size of the state ledger tends to a stable range, which depends on the total number of accounts in the current blockchain and the average number of accounts stored in each block.

6. Security Analysis

Unpredictability and unmanipulability are significant properties required for a secure consensus protocol. Unpredictability assures that malicious nodes cannot predict the consensus result until it is generated. Unmanipulability ensures that nodes cannot maliciously manipulate and tamper with the consensus result. In our PoR, malicious nodes can neither forecast the outcome of the miner's selection nor tamper with the outcome by manipulating the consensus process. This is because we use two unpredictable parameters H_{pre} and N_{cand} to calculate R_{Val}. R_{Val} cannot be manipulated because its calculation parameters are public to all nodes. Also, our *ID* generation is an unpredictable process. The R_{Val} and Target used by all nodes in the candidate test are the same, and the node that passes the candidate test must generate a Proof to prove its candidate identity. Therefore, a malicious node cannot forge its Proof because each Proof is broadcast in the blockchain network. The malicious nodes cannot tamper with the proof list to fool other candidate nodes when honest nodes are in the majority.

Furthermore, the malicious nodes cannot manipulate the outcomes of miner selection because our selection methods are reasonable and verifiable. If a malicious node violates Rule 1 in Section 4.2, i.e., it does not wait for t_1, but uses a smaller proof list ($W/2$) to recommend itself or its collaborators to become miners, this decision is disregarded because the blockchain network contains a larger proof list (Rule 6 of Section 4.2). Similarly, if a candidate node does

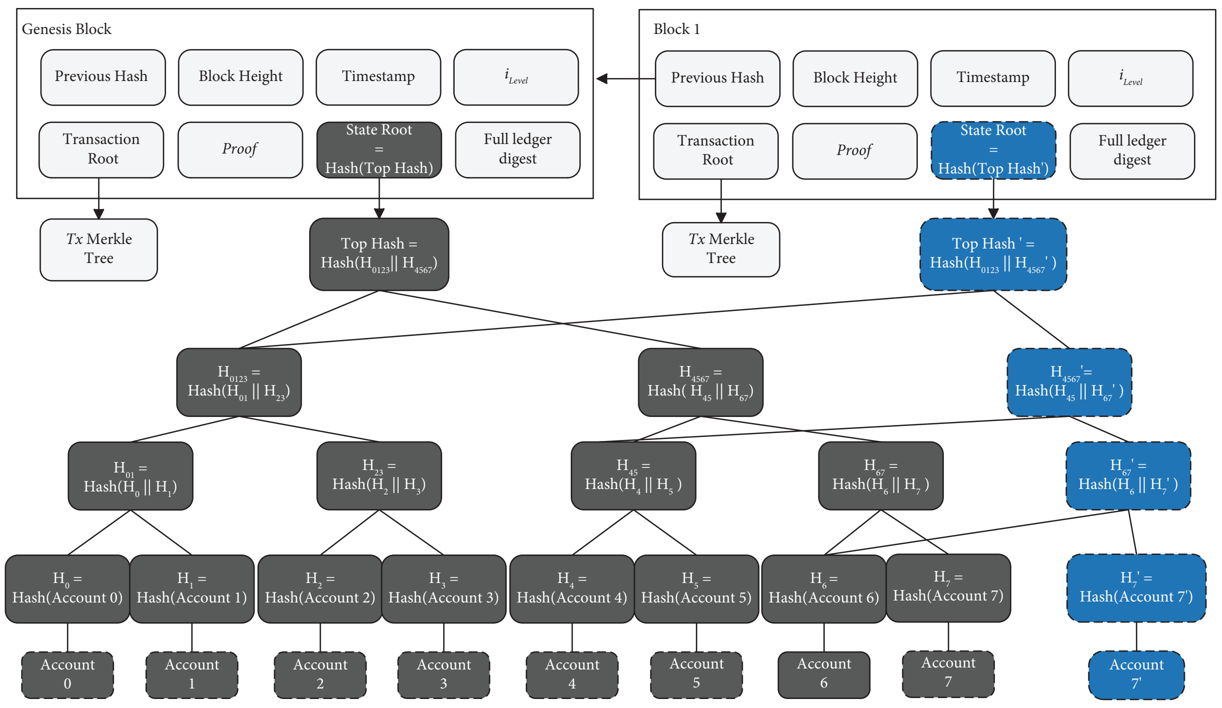

FIGURE 2: Example of the state tree update process (only account 7 state changes).

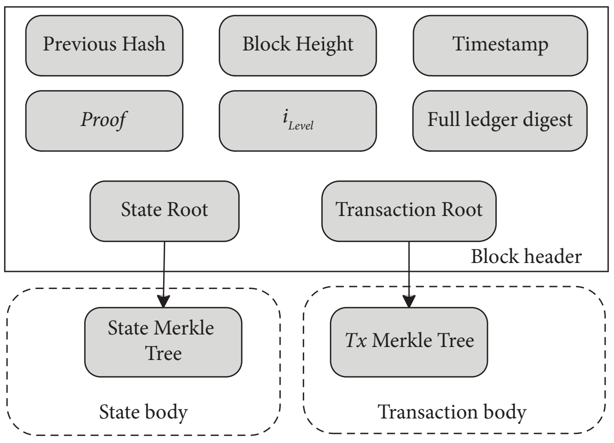

FIGURE 3: Block structure.

```
Input: $\mathbb{B} = \{B_1, B_2, ..., B_t\}$: A blockchain composed of $t$ blocks $B_1, B_2, ..., B_t$ where the blocks are sorted in chronological order, and $B_t$
    is the latest block.
Output: $\mathbb{B}$: A shorter blockchain
(1)  procedure
(2)     get list_of_newAcct from $\mathbb{B}$
(3)     get the min Acct.ver from list_of_newAcct
(4)          $V \leftarrow$ the min Acct.ver
(5)     for all $B_i \in \mathbb{B}$ do
(6)          if $B_i$.BH < $V$
(7)             delete $B_i$
(8)          end if
(9)     for all $B_i \in \mathbb{B}$ do
(10)         delete $B_i.t$
(11)    return $\mathbb{B}$
(12) end procedure
```

ALGORITHM 2: Historical block offloading.

not follow the consensus rules to wait for t_2 and maliciously broadcasts its new block or the consensus failure message, this decision is also disregarded because the decision must come from half of the candidate nodes to be valid.

6.1. Fault Tolerance. For a blockchain network with n consensus nodes (a collection of high-level nodes and mid-level nodes), our HLOChain can tolerate f Byzantine nodes, where $n \geq 2f + 1$.

Since all nodes use the same R_{Val} and Target, they have the same probability of passing the candidate test. Assuming that Byzantine nodes account for 50% of the current system, in the worst case, there are W nodes that pass the candidate test, among which there are $W/2$ Byzantine candidate nodes. These $W/2$ Byzantine candidate nodes will form a candidate list with the lowest number of candidate nodes $W/2$ to successfully launch an attack. As a result, if Byzantine nodes do not surpass 50%, i.e., $n \geq 2f + 1$, then all nodes can ensure fault tolerance.

6.2. Attack Resistance. In this section, we evaluate PoR's resistance against some common attacks, including double-spending attack, 51% computing power attack, and Sybil attack.

(1) Double-spending attacks: In this attack, the attacker attempts to spend the same coin in different transactions. The attack relies on the fork of the blockchain and the transaction delay confirmation mechanism. The collusion of more than 50% of nodes can cause forks. However, the fork is prevented through the additional rules in which we restrict this phenomenon of forks through additional rules in Section 4.2. Therefore, within our fault tolerance range, malicious nodes cannot launch double-spending attacks through forks.

(2) 51% computing power attacks: In a 51% computing power attack, the attacker who owns more than 51% computing power of the entire blockchain network is able to become a miner at a high advantage. Then, he can generate a longer blockchain to roll back historical transactions. Nevertheless, in our proposed PoR, the miners are generated randomly, instead of depending on the computing power of nodes. Thus, even if malicious nodes occupy 51% of the computing power, they cannot launch attacks through forks.

(3) Sybil attacks: In the Sybil attack, the attacker forges a large number of false identities to participate in the blockchain to interfere with the normal operation of the blockchain system. Assume an attacker generates a large number of ID, which exceed 50% of the total number of nodes in the blockchain. The probability that the attacker becomes a miner is greatly increased. It will seriously affect the fairness of miner selection. Therefore, we increase the cost of becoming a candidate node by pledging tokens in the process of generating Proof, so as to resist Sybil attacks.

7. Performance Evaluation

To evaluate the performance of the proposed solution, a concept-proof prototype of HLOChain is implemented in Java. The performance metrics include consensus time, transaction throughput, transaction confirmation latency, and ledger storage. Consensus time refers to the time that it takes to reach consensus on a given number of transactions. Transaction throughput refers to the number of transactions per second which are packed into the blockchain. Transaction confirmation latency refers to the time from when a transaction is proposed to when it is deposited on the ledger. The ledger storage indicates the size of the ledger data required to be stored by different type nodes at a given number of blocks.

We developed a private blockchain environment based on a virtual machine cluster built over several laptops and local servers. 4 laptops and 2 servers make up the hardware environment. We simulate different types of nodes by assigning different CPU cores and hard disk space to virtual machines. Among them, the configuration of the server is Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20 GHz, and the configuration of the laptop is Intel(R) Core (TM) CPU i5-12500H @ 2.50 GHz.

7.1. Consensus Time. In the experiment, we deployed 50 nodes. The block size is 1 MB, and each block contains 3000 transactions on average. The changes in the consensus time of PoW, PoS, PoEWAL, PoCh, and our PoR under different number of transactions are simulated, and Figure 4 shows the data results. It is clear that the consensus time of PoW is much higher than that of other algorithms because nodes need to continuously solve a cryptographic problem to mine new blocks in PoW. As the computing power of nodes increases, the difficulty is also increasing. For example, Bitcoin generates a new block in about 10 minutes on average. PoS uses stake to reduce the difficulty of mining. PoEWAL expects nodes to reach a consensus by partially solving the cryptographic puzzle within a fixed mining time. Their consensus time is much shorter than Pow, but they still need to continuously calculate the cryptographic puzzle. In the consensus process of PoCh and our PoR, there is no need to continuously calculate the cryptographic puzzle. A more energy-saving random method is used to select miner nodes. However, PoCh consumes a lot of time in the process of node interaction. Figure 4 clearly shows that when the number of transactions mined increases, PoR requires minimal consensus time compared to other lightweight algorithms.

7.2. Transaction Throughput. The transaction throughput is affected by two factors: the consensus time and the block size. Different block sizes are used to test the throughput of PoW, PoS, PoEWAL, PoCh, and our PoR. Figure 5 shows the experimental results. We found that with the increase of block size, the throughput increased significantly. This is due to the block size affecting the number of transactions contained in each block. The larger the block size, the higher

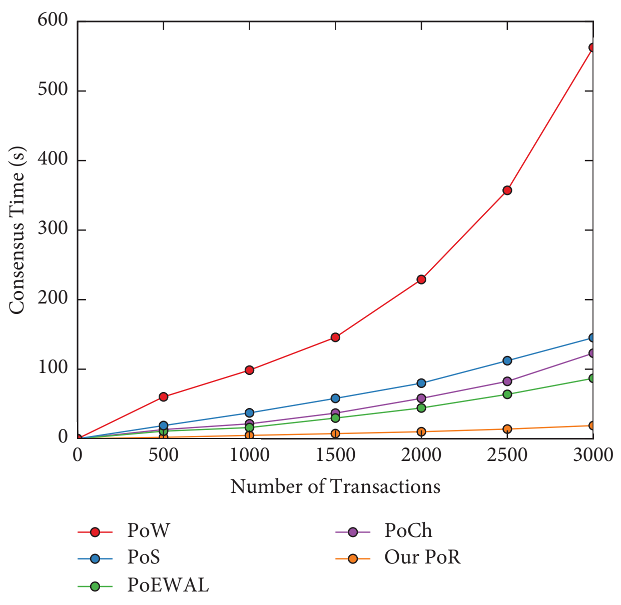

Figure 4: Consensus time under different number of transactions.

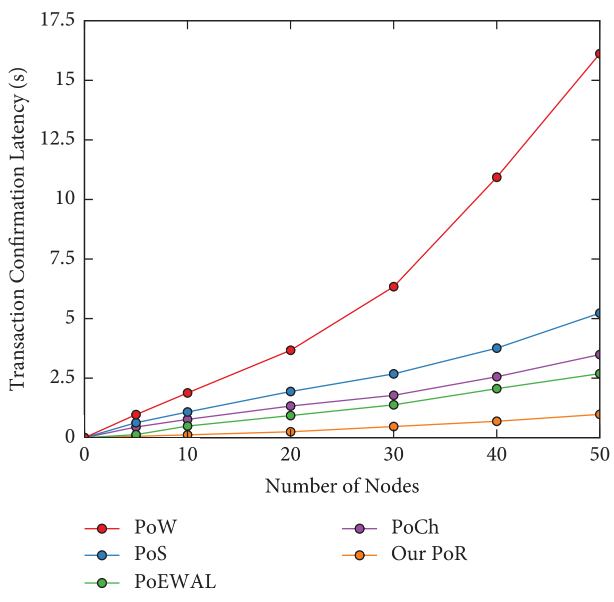

Figure 6: Transaction confirmation latency under different number of nodes.

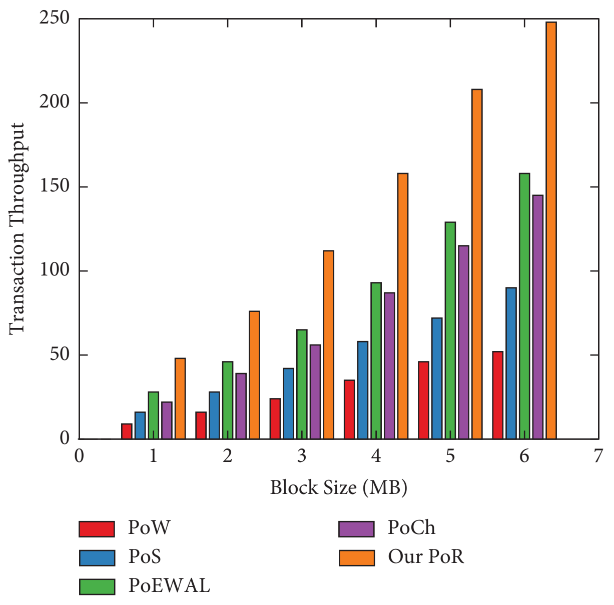

Figure 5: Transaction throughput under different block sizes.

the number of transactions per block. In addition, different consensus algorithms have different consensus time. It leads to different throughputs. The lower the consensus time, the higher the throughput. Compared to PoW, PoS, PoEWAL, and PoCh, our PoR has the lowest consensus time. As a consequence, the throughput of PoR is the highest. This shows that our HLOChain is more applicable to the IoT scenarios with large-scale transactions.

7.3. Transaction Confirmation Latency. We adjust the number of nodes to test the change of transaction confirmation latency. Figure 6 shows the experimental results. It can be found that with the increase of the number of nodes, the transaction confirmation latency of PoW changes significantly. It is mainly affected by the consensus time and the fork problem. The latency of PoS, PoEWAL, and PoCh is much lower than PoW and slightly higher than our PoR. However, they are still affected by the forking problem. In the case of 50 nodes, PoR only needs an average of 1.06 s for transactions to be confirmed. This is mainly due to the fact that we use additional rules to effectively alleviate the fork problem, and transaction confirmation does not need to wait for 6 blocks.

7.4. Ledger Storage. In the blockchain system, the full nodes need to store full ledger data to participate in the consensus process, while light nodes only need to store block headers and do not participate in consensus. We divide blockchain ledgers into a full ledger and a state ledger. High-level nodes need to store full ledgers, mid-level nodes only need to store state ledgers to participate in consensus, and low-level nodes only need to store block headers.

We compare the size of the full ledger and state ledger under different block heights, different block sizes (that is, the number of transactions contained in each block), and different numbers of accounts. Figure 7 shows the storage difference between the full ledger and the state ledger under different block heights. It is clear that the full ledger increases significantly with the increase of the number of blocks, while the state ledger is less affected by the number of blocks. Moreover, the state ledger is significantly smaller than the full ledger. Figure 8 shows the storage difference between the full ledger and the state ledger under different block sizes. It is clear that the impact of the block size is basically the same as that of the number of blocks. Figure 9 shows the storage difference between the full ledger and the state ledger under different account

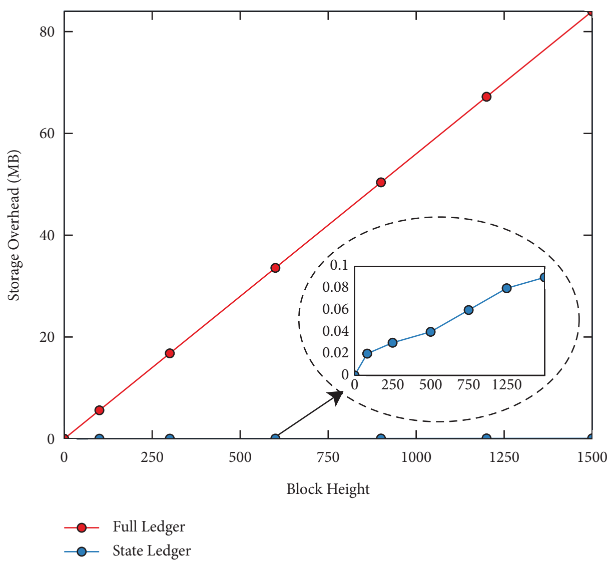

Figure 7: Ledger storage size under different block heights.

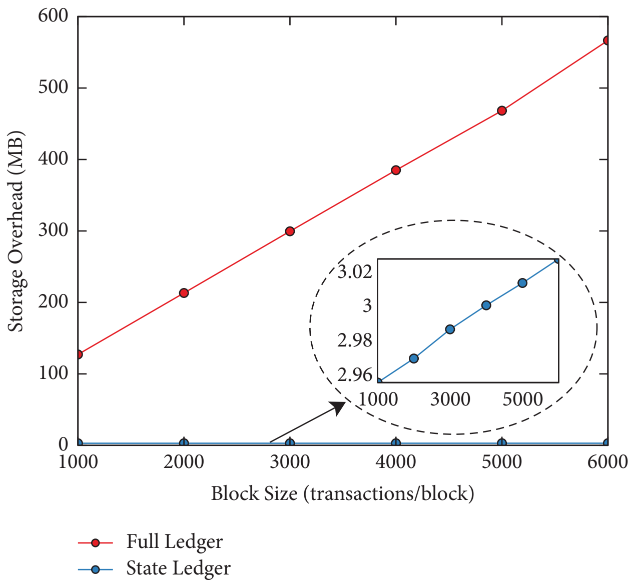

Figure 8: Ledger storage size under different block sizes.

numbers. In this respect, the growth rate of the full ledger decreases, while the growth rate of the state ledger increases. Therefore, it can be concluded that the state ledger is greatly affected by the number of accounts because the state ledger mainly stores the latest state of all accounts. It can be observed from the previous three figures that the size of the state ledger is significantly smaller than the full ledger, which is friendly for resource-constrained devices.

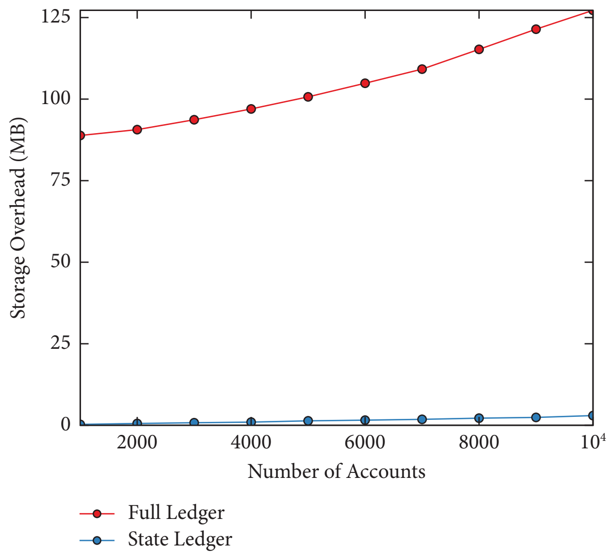

Figure 9: Ledger storage size under different number of accounts.

8. Conclusion

In this study, we proposed a hierarchical blockchain framework with lightweight consensus and optimized storage for IoT scenarios, which aims to solve the problems of high resource consumption and low efficiency of traditional blockchain. We fully considered the resource configuration of various types of devices in the IoT and combined edge computing to build a hierarchical system model. To enable mid-level nodes with limited resources to participate in the consensus, we designed a lightweight consensus protocol which provides lightweight and efficient mining operations and alleviates the computational power consumption of nodes. In addition, in order to reduce the storage pressure of nodes, we designed a storage optimization strategy, so that resource-constrained nodes just store a small amount of data to work like the full node. Finally, security analysis and experiments showed that our HLOChain is safe and feasible.

In the future, we will further study the data sharing problems under the proposed HLOChain for the IoT. Considering most IoT clients are resource-constrained and may not be trusted, realizing secure data sharing in IoT scenarios is facing the challenges of light weight, efficiency, and privacy preservation. To this end, we will try to utilize the smart contracts in implementing access control, to achieve lightweight, efficient, and secure data sharing for IoT.

Data Availability

All the experimental data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant nos. 62002139 and 62272203), Natural Science Foundation of Jiangsu Province (grant no. BK20200886), and Project funded by China Postdoctoral Science Foundation (grant no. 2019M651738).

References

[1] F. Xia, L. T. Yang, L. Wang, and A. Vinel, “Internet of things,” *International Journal of Communication Systems*, vol. 25, no. 9, pp. 1101-1102, 2012.

[2] A. Dorri, S. S. Kanhere, and R. Jurdak, “Blockchain for IoT security and privacy: the case study of a smart home,” in *Proceedings of the 2017 IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, pp. 618–623, Kona, HI, USA, March, 2017.

[3] M. Salimitari and M. Chatterjee, “A survey on consensus protocols in blockchain for IoT networks,” 2018, https://arxiv.org/abs/1809.05613.

[4] A. Faturahman, V. Agarwal, and C. Lukita, “Blockchain technology-the use of cryptocurrencies in digital revolution,” *IAIC Transactions on Sustainable Digital Innovation (ITSDI)*, vol. 3, no. 1, pp. 53–59, 2021.

[5] S. Alam, M. Shuaib, W. Z. Khan et al., “Blockchain-based initiatives: current state and challenges,” *Computer Networks*, vol. 198, Article ID 108395, 2021.

[6] A. Alkhateeb, C. Catal, G. Kar, and A. Mishra, “Hybrid blockchain platforms for the internet of things (IoT): a systematic literature review,” *Sensors*, vol. 22, no. 4, p. 1304, 2022.

[7] L. D. Xu, Y. Lu, and L. Li, “Embedding blockchain technology into IoT for security: a survey,” *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10452–10473, 2021.

[8] M. K. Lim, Y. Li, C. Wang, and M. L. Tseng, “A literature review of blockchain technology applications in supply chains: a comprehensive analysis of themes, methodologies and industries,” *Computers and Industrial Engineering*, vol. 154, Article ID 107133, 2021.

[9] I. Yaqoob, K. Salah, R. Jayaraman, and Y. Al-Hammadi, "Blockchain for healthcare data management: opportunities, challenges, and future recommendations," *Neural Computing and Applications*, vol. 34, no. 14, pp. 11475–11490, 2022.

[10] A. Awad Abdellatif, L. Samara, A. Mohamed et al., "Medgechain: leveraging edge computing and blockchain for efficient medical data exchange," *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 15762–15775, 2021.

[11] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: a survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1508–1532, 2019.

[12] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11008–11021, 2018.

[13] W. Chen, Z. Zhang, Z. Hong et al., "Cooperative and distributed computation offloading for blockchain-empowered industrial internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8433–8446, 2019.

[14] K. Lei, M. Du, J. Huang, and T. Jin, "Groupchain: towards a scalable public blockchain in fog computing of IoT services computing," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 252–262, 2020.

[15] P. Lang, D. Tian, X. Duan, J. Zhou, Z. Sheng, and V. C. M. Leung, "Cooperative computation offloading in blockchain-based vehicular edge computing networks," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 783–798, 2022.

[16] R. Xu, L. Hang, W. Jin, and D. Kim, "Distributed secure edge computing architecture based on blockchain for real-time data integrity in IoT environments," *Actuators*, vol. 10, no. 8, p. 197, 2021.

[17] S. Guo, X. Hu, S. Guo, X. Qiu, and F. Qi, "Blockchain meets edge computing: a distributed and trusted authentication system," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1972–1983, 2020.

[18] C. Li, J. Zhang, X. Yang, and L. Youlong, "Lightweight blockchain consensus mechanism and storage optimization for resource-constrained IoT devices," *Information Processing & Management*, vol. 58, no. 4, Article ID 102602, 2021.

[19] D. Puthal, S. P. Mohanty, and P. Nanda, "Proof-of-Authentication for scalable blockchain in resource-constrained distributed systems," in *Proceedings of the 2019 IEEE International Conference on Consumer Electronics*, pp. 1–5, Yilan, Taiwan, May, 2019.

[20] N. Raghav, N. Andola, S. Venkatesan, and S. Verma, "PoEWAL: a lightweight consensus mechanism for blockchain in IoT," *Pervasive and Mobile Computing*, vol. 69, Article ID 101291, 2020.

[21] J. Huang, L. Kong, G. Chen, M. Y. Wu, X. Liu, and P. Zeng, "Towards secure industrial IoT: blockchain system with credit-based consensus mechanism," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3680–3689, 2019.

[22] M. Saad, Z. Qin, K. Ren, D. Nyang, and D. Mohaisen, "e-PoS: making Proof-of-Stake decentralized and fair," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 8, pp. 1961–1973, 2021.

[23] S. Khan, W.-K. Lee, and S. O. Hwang, "AEchain: a lightweight blockchain for IoT applications," *IEEE Consumer Electronics Magazine*, vol. 11, no. 2, pp. 64–76, 2022.

[24] R. Xu, Y. Chen, and E. Blasch, "Microchain: a hybrid consensus mechanism for lightweight distributed ledger for IoT," 2019, https://arxiv.org/abs/1909.10948.

[25] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "LSB: a lightweight scalable blockchain for IoT security and anonymity," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 180–197, 2019.

[26] M. Kara, A. Laouid, M. Hammoudeh, M. AlShaikh, and A. Bounceur, "Proof of chance: a lightweight consensus algorithm for the internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 8336–8345, 2022.

[27] Y. Zhao, B. Niu, and P. Li, "A novel enhanced lightweight node for blockchain," in *Blockchain and Trustworthy Systems*, pp. 137–149, Springer, Singapore, 2020.

[28] Light client protocol, "Light client protocol," 2019, https://github.com/ethereum/wiki/wiki/Light-client-protocol.

[29] Getting synced, "Getting synced," 2019, https://github.com/paritytech/parity/wiki/Getting-Synced.

[30] Electrum, "Electrum," 2019, https://electrum.org/#home.

[31] Y. Liu, K. Wang, Y. Lin, and W. Xu, "A lightweight blockchain system for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3571–3581, 2019.

[32] X. Wang, C. Wang, K. Zhou, and H. Cheng, "ESS: an efficient storage scheme for improving the scalability of Bitcoin network," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1191–1202, 2022.

[33] C. Ehmke, F. Wessling, and C. M. Friedrich, "Proof-of-Property - a lightweight and scalable blockchain protocol," in *Proceedings of the 2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pp. 48–51, Gothenburg, Sweden, May, 2018.

[34] T. Kim, J. Noh, and S. Cho, "SCC: storage compression consensus for blockchain in lightweight IoT network," in *Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1–4, Las Vegas, NV, USA, January, 2019.

[35] S. Biswas, K. Sharif, F. Li, S. Maharjan, S. P. Mohanty, and Y. Wang, "PoBT: a lightweight consensus algorithm for scalable IoT business blockchain," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2343–2355, 2020.

[36] I. Eyal, A. E. Gencer, and E. G. Sirer, "Bitcoin-NG: a scalable blockchain protocol," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, pp. 45–59, Santa Clara CA, USA, March, 2016.

[37] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 931–948, Toronto, Canada, October, 2018.

[38] E. Kokoris-Kogias, P. Jovanovic, and L. Gasser, "Omniledger: a secure, scale-out, decentralized ledger via sharding," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, pp. 583–598, San Francisco, CA, USA, May, 2018.

[39] L. Luu, V. Narayanan, and C. Zheng, "A secure sharding protocol for open blockchain," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 17–30, Vienna, Austria, November, 2016.

[40] J. Nijsse and A. Litchfield, "A taxonomy of blockchain consensus methods," *Cryptography*, vol. 4, no. 4, p. 32, 2020.