Journal of Network and Computer Applications 222 (2024) 103785

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Research paper

A dynamic state sharding blockchain architecture for scalable and secure crowdsourcing systems

Zihang Zhen[a], Xiaoding Wang[a], Hui Lin[a,*], Sahil Garg[b,*], Prabhat Kumar[c], M. Shamim Hossain[d]

[a] College of Computer and Cyber Security, Fujian Normal University, Engineering Research Center of Cyber Security and Education Informatization, Fujian Province University, Fuzhou, Fujian, 350117, China
[b] Electrical Engineering Department, École de Technologie Supérieure, Montreal, Canada
[c] Department of Software Engineering, LUT School of Engineering Science, LUT University, 53850 Lappeenranta, Finland
[d] Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 12372, Saudi Arabia

ARTICLE INFO

Keywords:
Blockchain
State sharding
Crowdsourcing
Deep reinforcement learning

ABSTRACT

Currently, the crowdsourcing system has serious problems such as single point of failure of the server, leakage of user privacy, unfair arbitration, etc. By storing the interactions between workers, requesters, and crowdsourcing platforms in the form of transactions on the blockchain, these problems can be effectively addressed. However, the improvement in total computing power on the blockchain is difficult to provide positive feedback to the efficiency of transaction confirmation, thereby limiting the performance of crowdsourcing systems. On the other hand, the increasing amount of data in blockchain further increases the difficulty of nodes participating in consensus, affecting the security of crowdsourcing systems. To address the above problems, in this paper we design a blockchain architecture based on dynamic state sharding, called DSSBD. Firstly, we solve the problems caused by cross sharding transactions and reconfiguration in blockchain state sharding through graph segmentation and relay transactions. Then, we model the optimal block generation problem as a Markov decision process. By utilizing deep reinforcement learning, we can dynamically adjust the number of shards, block spacing, and block size. This approach helps improve both the throughput of the blockchain and the proportion of non-malicious nodes. Security analysis has proven that the proposed DSSBD can effectively resist attacks such as transaction atomic attacks, double spending attacks, sybil attacks, replay attacks, etc. The experimental results show that the crowdsourcing system with the proposed DSSBD has better performance in throughput, latency, balancing, cross-shard transaction proportion, and node reconfiguration proportion, etc., while ensuring security.

1. Introduction

Based on cryptography and P2P networks, blockchain technology has the characteristics of natural tamperability and decentralization. Blockchain technology is widely used to solve problems such as single point of failure, privacy disclosure, and unfair arbitration in traditional crowdsourcing systems (Zhu et al., 2019). For example, in a scenario where crowdsourcing and blockchain are combined, messages such as information registration, task release, and task submission in the crowdsourcing system are transformed to transactions and stored on the blockchain (Liang et al., 2022; Wang et al., 2022). However, due to the limitations of consensus mechanisms, there is an upper limit to the scalability of blockchain, which limits the performance of blockchain-based crowdsourcing systems. For example, the throughput of Bitcoin is about 10 transactions per second (Tschorsch and Scheuermann, 2016), while the throughput of Ethereum after adjusting the block spacing can only reach about 30 transactions per second (Buterin et al., 2014), which makes the throughput of crowdsourcing systems based on these two blockchain also close to the theoretical upper limit. In contrast, the throughput of visa servers can reach thousands of transactions per second (Klarman et al., 2018). From the perspective of security, decentralization, and scalability, the main problem with traditional blockchain is that scalability is sacrificed to ensure their security and decentralization, which makes it difficult to positively feedback the improvement of blockchain computing power into performance. However, after the security and decentralization of blockchain reach a certain level, sacrificing efficiency is not an option. Therefore, plenty of blockchain scalability technologies have gradually been proposed.

[*] Corresponding authors.
E-mail addresses: zzihang@foxmail.com (Z. Zhen), wangdin1982@fjnu.edu.cn (X. Wang), linhui@fjnu.edu.cn (H. Lin), sahil.garg@ieee.org (S. Garg), prabhat.kumar@lut.fi (P. Kumar), mshossain@ksu.edu.sa (M.S. Hossain).

https://doi.org/10.1016/j.jnca.2023.103785
Received 17 May 2023; Received in revised form 19 September 2023; Accepted 2 November 2023
Available online 10 November 2023

Blockchain scalability technologies can be roughly divided into Layer0, Layer1, and Layer2 technologies according to different application scopes. Layer0 technology is mainly aimed at improving the way of data transmission, by reducing the transmission delay to improve blockchain performance. Layer1 technology, also known as on-chain scaling technology, achieves blockchain performance improvement through on-chain technologies such as consensus protocols, P2P networks, and data structures, including sharding technology, Bitcoin-NG, and DAG distributed ledgers. Layer2 technology is also called off-chain scaling technology, which does not change the basic protocol of the blockchain, and the security of the system is still guaranteed by the blockchain itself. It only improves scalability through off-chain methods such as state channel technology and side chain technology (Zhou et al., 2020). Whether it is Layer0 or Layer2 technology, blockchain performance is not optimized from its structure. Therefore, Layer1 technology has gradually received attention in recent years. In Layer1 technology, there are mainly three types of techniques: reducing overhead, vertical scaling, and horizontal scaling. Vertical scaling mainly involves increasing the resources of nodes to reduce block interval and increase block size. Horizontal scaling is mainly achieved by replicating and partitioning data in the blockchain to improve parallelism. There are obvious theoretical limits to reducing overhead and vertical scaling. Therefore, improving the scalability of the blockchain through horizontal scaling has become a research focus, and sharding technology is the key to it (Nasir et al., 2022). On the other hand, while improving the scalability of blockchain, how to balance its security and decentralization is another issue we face. For the blockchain, full nodes maintain and store the data of the entire blockchain, allowing them to verify the validity of transactions and blocks. Therefore, the more full nodes there are in the blockchain, the more reliable the consensus of the blockchain can be guaranteed, and the safer the system can operate. However, taking Bitcoin and Ethereum as examples, the current hard disk space requirement for running Bitcoin full nodes exceeds 500GB[12] This seriously hinders nodes from becoming full nodes, increases the cost of nodes participating in transaction consensus, and restricts the decentralization and security of blockchain. Besides, after security is satisfied, excessive redundant data can cause a large amount of storage resources to be wasted in blockchain-based crowdsourcing systems.

For a crowdsourcing system, the entire business process is roughly illustrated in Fig. 1. The requester first publishes tasks through the crowdsourcing platform, and the crowd workers sense the tasks and submit answers through the platform. The submitted answers are then forwarded to the requester for evaluation, and finally the requester pays the workers. A blockchain-based crowdsourcing system makes the entire business process open and transparent. Steps such as task publishing, evaluation, and payment can be automatically completed through smart contracts, without relying on the reputation of the requester, workers, or any third party. However, due to the scalability limitations of blockchain, there is a certain upper limit on the efficiency of business processing in blockchain-based crowdsourcing systems. At the same time, a large amount of redundant and repetitive information is stored in the underlying blockchain, which actually limits the security of the crowdsourcing systems. As mentioned earlier, using state sharding blockchain technology in the construction of crowdsourcing systems can solve these problems. Through this technology, we can ensure both the fairness and transparency of the business process of the crowdsourcing system, improve system efficiency, and reduce unnecessary information redundancy.

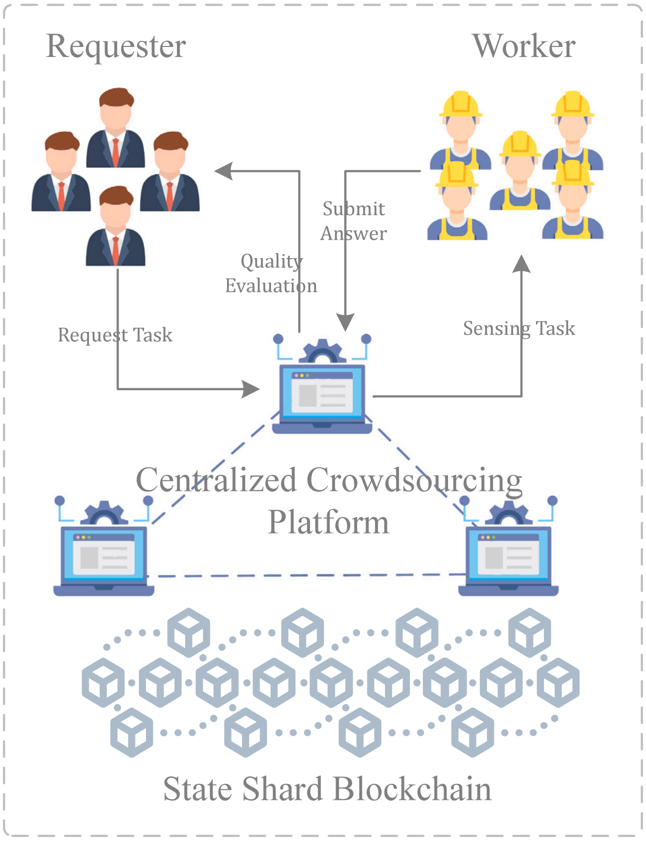

Fig. 1. Example of improved scalability of blockchain sharding.

However, in state sharding, dividing the storage space into different shards will inevitably results in many transactions that need to span different shards, which are called cross-shard transactions. Due to the consensus involved in different shards, the verification process of cross-shard transactions is very difficult. Thereby, ensuring the atomicity of these cross-shard transactions is extremely important, otherwise the consensus of the entire blockchain will become chaotic. On the other hand, the tolerance for malicious nodes or downtime nodes is also reduced to 1/n of the previous level, and shards with fewer participating nodes are less resistant to double-spending attacks. To ensure the long-term security of the blockchain, it is necessary to periodically reconfigure the nodes. However, excessive node reconfiguration during the state sharding reconfiguration phase can significantly impact the stability of the blockchain. Therefore, finding a way to leverage the benefits of state sharding to enhance blockchain performance while addressing the challenges it presents is an ongoing issue.

In summary, in order to address the performance limitations of current blockchain-based crowdsourcing systems, we will face the following four challenges.

- Blockchain faces difficulties in improving scalability.
- The storage space requirements of blockchain nodes are too high, which restricts the centralization and security of blockchain.
- Improving the scalability of blockchain will reduce its security and stability.
- How to dynamically adjust the scalability, security, and decentralization of blockchain in order to adapt to the different performance requirements of crowdsourcing systems in various scenarios?

[1] Data source: Jamie Redman, 2022, The size of Bitcoin's distributed ledger nears a half terabyte, https://news.bitcoin.com/the-size-of-bitcoins-distributed-ledger-half-terabyte/ (accessed 22 July 2023), and the requirement for running Ethereum full nodes exceeds 2TB.

[2] Data source: Go-ethereum, 2022, Hardware requirements, https://geth.ethereum.org/docs/getting-started/hardware-requirements/ (accessed 22 July 2023).

In order to solve the above problems, in this paper, we first use state sharding technology to improve the scalability and decentralization of the blockchain of the crowdsourcing system. And through the sharding module, reconfiguration module and security module, the atomicity of blockchain cross-shard transactions and the security of the blockchain are guaranteed. Finally, through deep reinforcement learning, the blockchain achieves an optimal balance among decentralization, security, and scalability, and improves the performance of the crowdsourcing system.

- We propose a blockchain architecture based on dynamic state sharding using deep reinforcement learning, called DSSBD, to solve the problem of performance degradation of the crowdsourcing system when the blockchain is in a non-optimal balance between decentralization, security, and scalability. To the best of our knowledge, DSSBD is the first architecture that uses deep reinforcement learning in state sharding blockchain to optimize sharding and block generation. By ensuring the security of blockchain and the effectiveness of transactions, it greatly improves the performance of blockchain based crowdsourcing systems.
- We consider load balancing for a blockchain-based crowdsourcing system during the reconfiguration phase. We utilize a modified state tree to quickly differentiate transaction types, reducing the time complexity of querying transaction types to $O(1)$.
- Through security analysis, we prove that DSSBD can effectively resist attacks such as transaction atomicity attack, double spending attack, sybil attack and replay attack. The experimental results demonstrate that for the blockchain-based crowdsourcing system, the proposed architecture is superior to the most advanced baselines in terms of throughput, transaction delay, cross-shard transaction proportion, node reconfiguration proportion, and average shard transaction load.

The rest of this paper is organized as follows. We present the state-of-the-art blockchain sharding technologies in Section 2. The system model is given in Section 3. We elaborate the design of the proposed DSSBD in Section 4. The security analysis is presented in Section 5. The performance evaluation is conducted and the experimental results are discussed in Section 6. We conclude this paper in Section 7.

2. Related works

In this section, we first review blockchain sharding technologies, and then introduce relevant researches on deep reinforcement learning technology of blockchain sharding.

2.1. Sharding blockchain

Sharding technology is widely used in databases and cloud services. With the gradual increase of stored data, in order to optimize the storage method of each node, the storage efficiency of data is improved by using sharding technology. Today, sharding technology is also widely used in blockchain to solve scalability problems.

The sharding technology in the blockchain is mainly divided into three types, network sharding, transaction sharding and state sharding. Among them, network sharding is to divide the entire network into different subnets through a certain organizational method at the level of network connection, and each subnet processes transactions in the blockchain in parallel. The transaction sharding is aimed at the transactions to be verified in the blockchain, and divides these transactions into different processing areas through specific rules. For state sharding, it divides the entire blockchain ledger into many sub-ledgers, and these sub-ledgers are stored in each shard. Each shard can not only verify transactions at the same time, but also reduce storage pressure. In state sharding, a complete ledger can only be formed through the entire network. Under the current technical conditions, both network sharding and transaction sharding have relatively ideal methods. There are still many technical problems in the realization of state sharding, and there are some technical problems in state sharding (Hashim et al., 2022).

2.2. State sharding

According to the access rules of nodes, the blockchain can be divided into two types, permissioned blockchain and permissionless blockchain. Therefore, the corresponding state sharding blockchain is also divided into two cases to consider.

For a permissioned blockchain, since its nodes are operated by identified participants with a certain degree of trust, the security of the nodes is high, and the accounting rights in the shards are often in a small number of certain nodes. Therefore, state sharding of permissioned blockchains usually does not require reconfiguration after sharding. For example, Sharper (Amiri et al., 2021) is a state sharding blockchain based on a permissioned chain. Since it does not need to consider the reconfiguration of nodes, it adopts a directed acyclic graph (DAG) (Gao et al., 2020) structure for the ledger, so that the ledger of each shard chain can store cross-shard transactions.

For non-licensed blockchains, due to the low degree of node trust (or even no trust), the overall security of the blockchain needs to be considered during the sharding process, and many mechanisms to ensure security are added. For example, although Elastico (Luu et al., 2016) is a transaction shard, its way of establishing an identity and forming a committee through POW, and then forming an on-shard consensus through BFT has affected the blockchain architecture of many transaction shards.

RapidChain (Zamani et al., 2018) is a state-sharded blockchain architecture that adopts a committee method similar to Elastico for sharding. In order to reduce the cost of reconfiguration, a limited Cuckoo Rule is proposed to select active nodes for committee reconstruction. OmniLedger (Kokoris-Kogias et al., 2018) mainly focuses on the atomicity of cross-shard transactions after state sharding. It performs state sharding by constructing the main chain and other sub-chains, and proposes a hierarchical processing that trusts in different shards first, and then merges and verifies them. The mechanism (Trust-but-Verify) ensures the atomicity of cross-shard transactions. Monoxide (Wang and Wang, 2019) is a relatively balanced state sharding technology. It proposes the concept of final atomicity to efficiently process cross-shard transactions, and uses the merge mining method called Chu-ko-nu to improve the security after sharding. BrokerChain (Huang et al., 2022) addresses the hotspot issues encountered in Monoxide by utilizing a graph partitioning method. This method converts cross-shard transactions into intra-shard transactions, effectively resolving the problem.

2.3. Deep reinforcement learning based sharding

With the improvement of computer computing power, blockchain technology and machine learning are increasingly being applied to practical scenarios such as the Industrial Internet of Things, and there are more and more solutions that combine deep reinforcement learning and sharding technology to solve blockchain scalability.

Liu et al. (2019) combined industrial IoT devices with blockchain, proposed the DRLB model, used the DQN algorithm to solve the problem of limited scalability of blockchain, and proposed a performance optimization framework for blockchain-based IoT systems. Yun et al. (2020) proposed a permissioned chain-based sharding blockchain architecture DQNSB, mainly analyzing the performance of the model under the PBFT consensus, and improving the performance of DRLB by optimizing the DQN model. Yang et al. (2022) used the K-means algorithm for transaction shard, and used the D-DQN algorithm to optimize the performance of the system.

Although the research on the combination of blockchain sharding and deep reinforcement learning has gradually become popular in recent years, due to the complexity of state sharding, the current research is based on account sharding. Therefore, combining state sharding with deep reinforcement learning is the focus of this paper.

3. System model

Considering that blockchain based crowdsourcing systems rely on the underlying blockchain, we aim to optimize the scalability and decentralization of blockchain while ensuring security by proposing a blockchain architecture based on state sharding using deep reinforcement learning, thereby improving the performance and security of blockchain based crowdsourcing systems. The system model is shown in Fig. 2.

3.1. Types of nodes in DSSBD

To obtain the optimal block generation strategy, we first define epoch as a fixed clock period. In every epoch, both consensus and node reconfiguration operations are performed. For blockchain, there are two types of nodes, namely miner nodes and non miner nodes. In order to ensure the atomicity and security of transactions between different shards at the consensus phase, and the stability and consistency of the blockchain at the reconfiguration phase, we need to encourage some nodes to maintain the blockchain as non-miner nodes, which can be further divided into relay nodes and shard nodes.

- *Relay-Node(R-Node)*: Relay-Nodes are generated through smart contracts by pledging equity. Each R-Node has a relay account. In order to realize the relay of cross-shard transactions, this account needs to be divided into multiple sub-accounts and added to different shards respectively. Other nodes achieve cross-shard transactions by conducting transactions with the sub-accounts of relay nodes.
- *Shard-Node(S-Node)*: Shard-Nodes are generated through application and election. They do not join any specific state shards, but form a new shard called S-Shard similar to a committee. S-Nodes are mainly responsible for planning and reconfiguring the nodes of each shard in each epoch based on transaction load and security.

Compared to regular mining nodes, R-Nodes retain the state information of multiple shards and reach consensus on each shard. Essentially, it is equivalent to a full node that stores multiple shard state information. If a R-Node stores the state information of all shards, it is equivalent to a regular full node before sharding. On the other hand, S-nodes not only retain the entire blockchain's state information in the S-Shard but also need to reach consensus on the sharding division for the next epoch. However, in order to become such non-mining nodes, nodes need to stake equity or participate in elections to prevent malicious behavior.

In particular, if a node wants to become a R-Node for more shards, it not only needs to pay more resources to store multi-shard state information, but also needs to pay more equity for mortgage. This approach effectively prevents the processing of cross-shard transactions from being too centralized on individual relay accounts. However, in order to encourage nodes to become R-Nodes to process cross-shard transactions, maintain system operation, and increase the degree of decentralization, part of the transaction fee will be transferred to the relay account.

3.2. Main phases of DSSBD

In each epoch, DSSBD needs to process the interaction information between the workers and the crowdsourcing system in two phases: blockchain consensus and node reconfiguration. Then, decisions are made through deep reinforcement learning. Note that within each epoch, all legal transactions in the consensus phase will become the basis for S-Shard to re-shard in the reconfiguration phase. In the next epoch's blockchain consensus phase, the proportion of cross-shard transactions and the transaction load of each shard will be affected by the results of the previous round of reconfiguration.

- *Blockchain Consensus Phase*: The consistency of the blockchain is achieved through two types of consensus nodes, namely the miner nodes and R-Nodes. They verify transactions and form consensus to ensure the consistency of the entire network state. Regardless of whether the consensus protocol is POW or PBFT, for transactions where the sending and receiving addresses belong to the same shard, consensus among the consensus nodes within the shard is sufficient. To handle cross-shard transactions, R-Nodes are utilized to split the raw transaction into pre-cross-shard transactions and post-cross-shard transactions. These two sub-transactions are then forwarded to the respective shards. The consensus nodes of each shard verify these sub-transactions, forming sub-consensus. Eventually, these sub-consensus are merged into a complete consensus. This process is necessary since the state information of the entire blockchain is divided into different shards. This process is described in detail in Section 4.2.2.
- *Node Reconfiguration Phase*: It is necessary to reconfigure shards for all nodes except the S-Nodes at this phase. To improve the overall efficiency of the blockchain, we have to reduce the blockchain's ability against malicious nodes. Therefore, in order to prevent malicious nodes from gathering in the same shard to control the entire blockchain, we need to reconfigure the shards of the nodes. To this end, given the transactions within an epoch, we let the S-Nodes in the S-Shard reach a consensus based on the re-partition of the node shard information, which is elaborated in Section 4. To reduce the cost of reconfiguration, only partial reconfiguration is performed on active nodes at the end of each epoch. Through node reconfiguration, we reduce the proportion of cross-shard transactions and optimize the transaction load of each shard.

3.3. Security model

This paper considers transaction atomicity attack, double spend attack, sybil attack and replay attack against DSSBD. These attacks from the perspective of transaction atomicity and consensus security will cause malicious nodes to obtain improper benefits by repeating transactions, modifying the state of the blockchain, and even controlling the network. This behavior undermines the integrity and reliability of the state of the entire blockchain network. According to the analysis of Section 5, the architecture proposed in this paper can effectively resist these attacks.

- *Transaction Atomicity*: It means that any type of transaction should be treated as an indivisible unit, either fully executed successfully or completely rolled back in case of failure, without any partial execution.
- *Consensus Security*: It refers to the process in which participating nodes in each shard reach consensus, which can prevent malicious behavior and maintain the integrity and consistency of the entire blockchain network.

In addition, due to the existence of the reconfiguration phase, stability and consistency are also important security indicators. Within one epoch, the more participating nodes in the reconfiguration, the

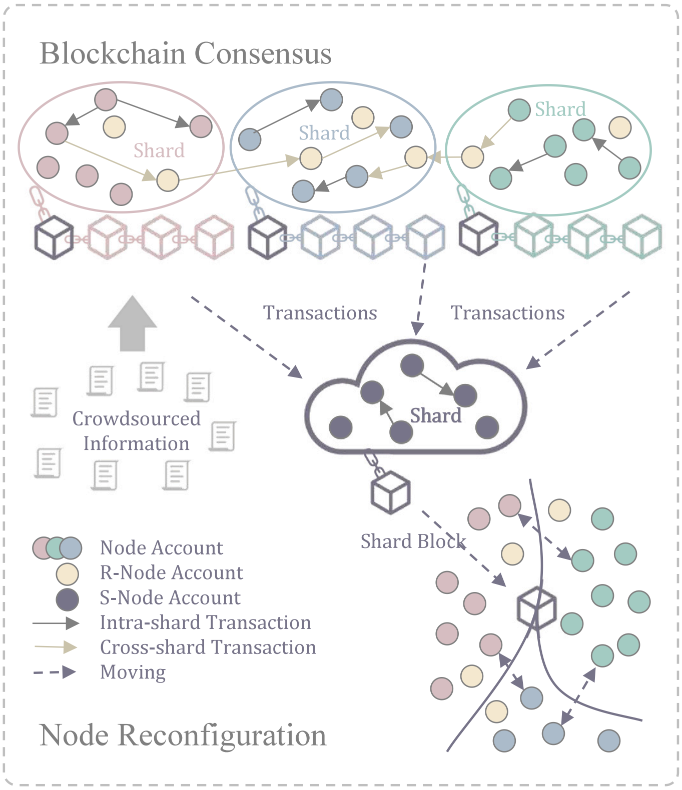

Fig. 2. Model diagram of a blockchain-based crowdsourcing system.

greater the possibility of multiple states simultaneously due to network transmission delays and node processing capabilities. This can lead to forks, resulting in inconsistency and confusion in the entire blockchain network. With continuous reconfiguration, this confusion and inconsistency will continue to intensify. In this situation, malicious nodes, with clear intentions, can control the entire blockchain network with a small amount of computing power.

In order to make the article more clear, In order to make the article more clear, brief explanations of some important terms mentioned in this paper are provided in the Table 1.

4. Strategy design of DSSBD

In this section, we illustrate the design of the proposed DSSBD in details. First, we introduce the static environment and transaction flow of the state sharding blockchain. Then, we will explain how to dynamically optimize the state sharding using deep reinforcement learning.

4.1. Static environment of state sharding blockchain

In order to build an efficient, secure and stable state sharding blockchain environment for the blockchain-based crowdsourcing system, we need to consider the sharding module of newly added nodes, graph partitioning module, node reconfiguration module, and sharding security module.

4.1.1. Initial sharding

To avoid a Sybil attack (Zhang and Lee, 2019), any newly joining node will need to solve a hash puzzle to determine which shard it is joining. In particular, for account-based blockchain, the shard to which the account belongs can be quickly determined through the hash value of the first or last digits of the account address.

4.1.2. Graph partitioning

Considering the correlation between transactions, in order to reduce the number of cross-shard transactions, we use the METIS (Karypis and Kumar, 1998) algorithm to partition the graph. METIS is a well-known heuristic graph partitioning tool that can divide the transaction flow snapshot into non-overlapping parts while considering workload balancing. In each epoch, we extract the accounts involved in the transaction as the vertices of the graph, and the transactions between the accounts as undirected edges. Since the cost of cross-shard transactions is mainly related to the number of transactions, the weight of an edge is expressed as the number of transactions involved in the two accounts in this epoch. If the account does not have transactions in this epoch, the graph partition will not be considered to reduce the cost of blockchain reconfiguration.

Thereby, we define a transaction flow snapshot at epoch t by $Y(t) = \{y_{i,j}^{(t)}\}$, where $Y(t)$ is a $M \times M$ matrix, and M is the number of accounts involved in all transactions in this epoch. Suppose we are at the epoch t, if the number of transactions between accounts i and j in the block is x, then the transaction flow snapshot is presented by $y_{i,j}^{(t)} = x$; if there is no transaction between i and j, then $y_{i,j}^{(t)} = 0$. For accounts i and j at epoch t, if there are no transactions packaged into blocks, then these two accounts should not be considered for $Y(t)$.

4.1.3. Reconfiguration

At the beginning of each epoch, the miner nodes of each shard reach a consensus on the transactions in the transaction pool according to the results of the previous epoch reconfiguration, package legal transactions into blocks and link them into chains. At the same time, the S-Node in the S-shard reads and records legitimate transactions. After all transactions in this round are read, S-Node divides the miner nodes according to the graph division strategy, and the division result will reach a consensus in S-Shard to form S-Block. S-Blocks will be recorded in the S-Shard chain and broadcast to other nodes. The miner nodes of other shards reconfigure according to the content of the S-Block, adjust

Table 1
Meanings and abbreviations.

Abbreviations	Meanings
Vertical Scaling	It belongs to the Layer1 scalability method, which mainly improves the blockchain scalability by improving parallelism.
Horizontal Scaling	It belongs to the Layer1 scalability method, which mainly improves the blockchain scalability by increasing the resources of a single node.
R-Node	Also called Relay-Node, it is mainly used for the relay of cross-shard transactions.
S-Node	Also called Shard-Node, it mainly used for node reconfiguration and decision-making.
S-Shard	It is a committee formed by all S-Node.
S-Block	This is the block that saves the results formed by S-Shard after consensus on the current blockchain state.
Pre-Cross-Shard Transaction	This is the broadcast transaction constructed after the cross-shard transaction is divided by R-Node.
Post-Cross-Shard Transaction	This is a transaction constructed and forwarded to another shard after the cross-shard transaction is divided by R-Node.
Modified State Tree	This is a tree structure that uses the Merkle Patricia Tree to store state information for each shard.
Epoch	The clock cycle for DSSBD to perform a blockchain consensus and node reconfiguration, which is also the cycle for a decision in deep reinforcement learning.
Episode	This is a complete running process in deep reinforcement learning, starting from the initial state, through a series of actions, reaching the termination state or reaching the termination condition.

the shards, and record the S-Block as the first consensus block in the current epoch in their own chains.

To ensure the security and stability of the blockchain during the reconfiguration process and minimize the cost involved, it is crucial to take into account the number of nodes that require reconfiguration in each round of epoch $\Delta(t)$.

4.1.4. Sharding security

Since the sharding strategy tends to divide related nodes into the same shard, in order to resist the problem of security degradation caused by sharding, we need to improve security through a method of merged mining (Wang and Wang, 2019). If all the nodes of the two shards store all the state information of these two shards, it is essentially equivalent to merging the two shards into one.

From the perspective of security, for a POW consensus blockchain, the number of malicious nodes $N_p < \frac{1}{2}N$ that the entire blockchain can tolerate, where N is the total number of nodes. Considering the worst case, that is, there is no merged mining in the blockchain at all, then the number of malicious nodes that each shard can tolerate is reduced to $N_p < \frac{1}{2K}N$, where K represents the number of shards. So from a security point of view, the maximum number of shards in the worst case is $K^{max} < \frac{N}{2N_p}$.

4.2. Transaction flow of state sharding blockchain

For the blockchain-based crowdsourcing system, the transfer of data is realized through transactions. Therefore, in the state sharding blockchain, how to deal with intra-shard transactions and cross-shard transactions is a very important issue. We need to consider modifying the state tree, cross-shard transaction process, and transaction confirmation delay.

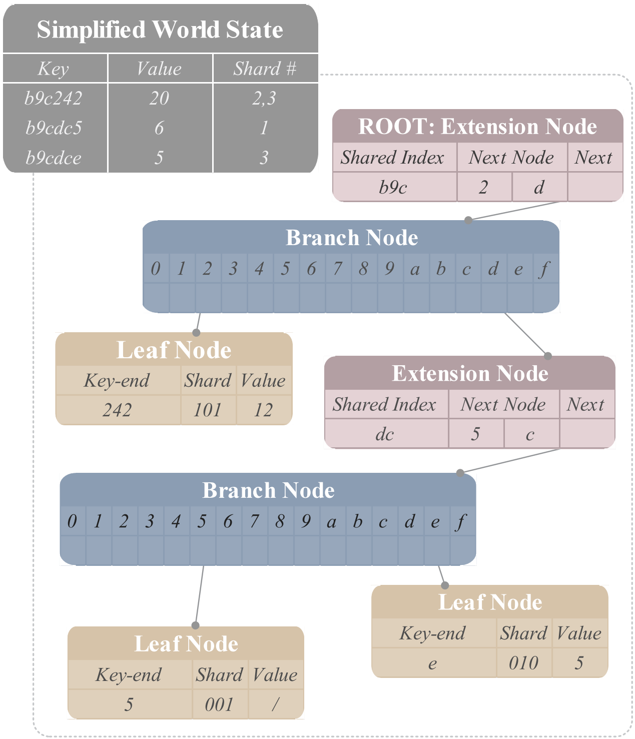

Fig. 3. Example of modified state tree of shard #3.

4.2.1. Modified state tree

In order to quickly distinguish cross-shard transactions and intra-shard transactions, we need to modify the state tree and map it to the local memory's account-shard dictionary database. Since the maximum number of shards is K^{max}, the leaf nodes of the state tree need to add a K^{max}-bit field to indicate the shard to which the account address belongs. For an R-Node, which retains state information for multiple shards, there are multiple bits in this field that need to be set to one.

As shown in Fig. 3, it is a case of a Modified State Tree for shard #3. For accounts in shard #3, the Modified State Tree will retain all their state information. For accounts not in shard #3, the Modified State Tree will only retain their shard information and use a regular pointer to point to these accounts instead of a hash pointer. For relay accounts, the Modified State Tree retains their shard information, as well as the account states belonging to this shard. In this way, the time complexity of checking whether a transaction is a cross-shard transaction is simplified to $O(1)$.

If the state tree is not modified and the local account-shard dictionary database is not built, then to determine whether a transaction is a cross-shard transaction, it is necessary to query the entire path in the shard's state tree to determine if the account exists. The time complexity becomes $O(n)$, where n is the length of the query path in the shard's state tree. Additionally, if the transaction is a cross-shard transaction, it is also necessary to delegate the relay node to query the state tree of other shards to obtain the shard number it belongs to.

4.2.2. Cross-shard transaction process

For a cross-shard transaction, since the state of the account is divided into different shards, the consensus process of the transaction needs to be divided into two steps. A typical cross-shard transaction case in DSSBD is shown in Fig. 4.

- **Create an raw transaction**: First create an raw transaction tx_{raw} in the sending account A, the receiving account is B, the number of tokens sent is v, and the relay account $C1$. and an appropriate transaction lock time H_{lock} is selected, which is essentially the number of subsequent blocks. The raw transaction is then sent to the relay account $C1$, which is expressed as follows,

$$tx_{raw} := \langle\langle B, v, C, H_{lock}, \eta_{sender}, \eta_{relay}\rangle, \sigma_A\rangle \tag{1}$$

where η_{sender} and η_{relay} represent the transaction nonce of the sending account A and the sub-account $C1$ of relay node C in shard #1, and σ_A represents the signature of the account A.

- **Creates a pre-cross-shard transaction**: After the relay node C receives the raw transaction, it creates a pre-cross-shard transaction tx_{pre} and broadcasts it to the blockchain network. After the nodes of each shard receive tx_{pre}, they verify the legitimacy of tx_{pre} through signature verification, and calculate the address of sender A. After querying and the modified state tree, the nodes of each shard can obtain the shard sequence of the source address and destination address, and forward the transaction to shard 1 and shard 2. After verifying that the transaction is legal, the tx_{pre} will be added to the transaction pool of shard 1 where account A is located, waiting to be packaged. The representation of the pre-cross-shard transaction is as follows,

$$tx_{pre} := \langle\langle \Omega_{pre}, tx_{raw}, H_{current}\rangle, \sigma_C\rangle \tag{2}$$

where Ω_{pre} represents the transaction type is a pre-cross-shard transaction, $H_{current}$ represents the current block height, and σ_C represents the signature of the relay account $C1$.

- **Confirm the pre-cross-shard transaction**: After packaging tx_{pre} into blocks, the sender A will transfer v tokens to the relay account $C1$, and the account $C1$ will retain these tokens until the transaction lock H_{lock} expires. At the same time, the nonce of sender A will be increased by 1 to avoid replay attacks (Hu et al., 2019).
- **Creates a post-cross-shard transaction**: After the tx_{pre} is confirmed, the relay node C will create a post-cross-shard transaction tx_{post}, which will be broadcast to shard #2 where the destination account B is located. When the transaction is verified as valid, it will be added to the transaction pool of shard #2. In particular, it is necessary to ensure that the balance of the intermediary node in shard #2 is greater than the token v. The representation of the post-cross-shard transaction is as follows,

$$tx_{post} := \langle\langle \Omega_{post}, tx_{raw}\rangle, \sigma_C\rangle \tag{3}$$

where Ω_{post} represents the transaction type is a post-cross-shard transaction, σ_C represents the signature of the relay account C.

- **Confirm the post-cross-shard transaction**: After the tx_{post}n is packaged into the block, if the block height does not exceed the transaction lock height, the relay account $C2$ will transfer v tokens to the receiver B. At the same time, in order to avoid replay attacks, the nonce of relay account $C2$ will also increase by one.
- **Transaction fail process**: If the transaction fails, since the tx_{pre} information has been broadcast to the network, shard #2 will package the tx_{pre} into the block, and the nonce of the relay account $C2$ will also increase by 1, and at the same time, in shard #2 node will send a failure proof tx_{fail} to shard #1. The representation of the failure proof is as follows,

$$tx_{fail} := \langle tx_{raw}, dest, H_{dest}, \{P_{dest}\}\rangle \tag{4}$$

where $dest$ represents the sequence of the target shard, H_{dest} represents the block height of the target shard, and $\{P_{dest}\}$ represents the path from the root node of the merkle tree to the previous shard transaction. Once the failed proof is verified by the nodes of shard #1, it will be packaged into the block of shard #1, and the locked balance v in $C1$ will be returned to A.

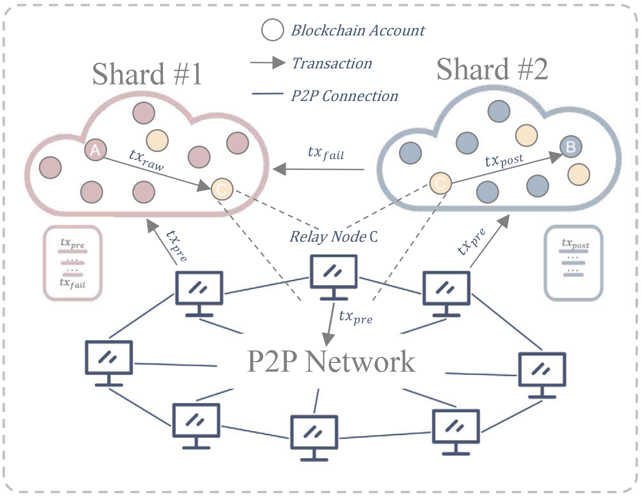

Fig. 4. Example of cross transaction flow.

4.2.3. Transaction confirmation latency

For intra-shard transactions, its security is jointly guaranteed by all nodes in the shard through consensus. The transaction confirmation delay needs to consider the block generation delay T_k^{hash} of the block node through the hash collision, the propagation delay $T_k^{transmit}$ of the block broadcast and the verification delay $T_k^{validate}$ of the block, i.e.,

$$T_{Latency} = T_k^{hash} + T_k^{transmit} + T_k^{validate} \tag{5}$$

Therefore, for intra-shard transactions, only $T_{Latency}$ needs not to be greater than the limit delay $T_{Limit} = T_I$, i.e.,

$$T_{Latency} \leq T_{Limit} \tag{6}$$

For a cross-shard transaction, the consensus of the transaction needs to be divided into two steps and completed in different shards. For pre-cross-shard transactions and pre-cross-shard transactions divided by R-Nodes, it is also necessary to consider the block generation delay $T_k^{transmit}$, propagation delay $T_k^{transmit}$ and verification delay $T_k^{validate}$ in their respective shards, so that they satisfy the above formula.

From the overall perspective of cross-shard transactions, relay latency also needs to be considered. In DSSBD, the relay of transactions is completed locally at the relay node, so the relay delay T_k^{relay} can be ignored. The formula for calculating the confirmation delay of cross-blockchain transactions is as follows,

$$T_{Latency} = 2(T_k^{hash} + T_k^{transmit} + T_k^{validate}) \tag{7}$$

Moreover, the DSSBD guarantees the atomicity of transactions through transaction hash locks H_{lock}. For cross-shard transactions, the limit delay $T_{Limit} = T_I \times H_{lock}$ at this time, where $H_{lock} \in \{1, 2, \ldots, H_{lock}^{max}\}$.

4.3. Dynamic optimization of state sharding mechanism

To optimize the performance of the blockchain, it is necessary to make joint decisions and adjustments on the selection of the number of shards, block interval and block size. We formulate the joint optimization problem as a Markov decision problem consisting of state, action and value function (Jin et al., 2023). In actual scenarios, the number of blockchain nodes involved in decision-making increases linearly with throughput. To avoid the curse of dimensionality, we need to combine deep neural networks with reinforcement learning, called deep reinforcement learning. In particular, in order to balance the choice between exploration and optimization in deep reinforcement learning, we choose the off-policy method to train the model, and considering the discreteness of the action space, Double DQN is the most suitable algorithm.

4.3.1. State space

We use the notation $S(t)$ to represent the observed state of the environment at a discrete time epoch t, where t ranges from 1 to T.

$$S(t) = \{\mu(t), \sigma(t), Y(t)\} \tag{8}$$

where $\mu(t) = \{\mu_{i,j}\}$ represents the transmission rate between nodes i, j, $\sigma(t) = \{\sigma_i\}$ represents the computing power of nodes i, and $Y(t)$ represents the snapshot of the current transaction graph, its definition is detailed in Section 4.1.2.

In the DSSBD model, for the state space, at each epoch t, we model the transaction flow snapshot $Y(t)$ and the transmission rate $\mu(t)$ between nodes as a symmetric matrix, and the computing power $\sigma(t)$ as a diagonal matrix. In order to speed up the training process, we remove redundant data in the state space, and finally simplify the state space to$M * M$ size, where M is the maximum number of nodes in the epoch.

4.3.2. Action space

As mentioned above, in Layer1 blockchain scalability technology, overhead reduction, horizontal scaling and vertical scaling are the three main means. Among them, reducing overhead is not the focus of this paper, and we do not make decisions on it. However, the number of sharding K during horizontal scaling, the block interval time T_I and max block size S_B during vertical scaling will affect the scalability of the blockchain. Therefore, the action space when epoch is t is expressed as follows,

$$A(t) = \{K(t), S_B(t), T_I(t)\} \tag{9}$$

where $K(t) \in \{8, 16, \dots, K^{max}\}$, $S_B(t) \in \{10, 20, \dots, S_B^{max}\}$, the unit is MB, $T_I(t) \in \{4, 8, \dots, T_I^{max}\}$, and its unit is seconds.

4.3.3. Value function

In deep reinforcement learning, the agent can directly or indirectly learn the optimal policy by maximizing the long-term reward. In particular, in the Double DQN algorithm, its long-term reward is the estimated Q value in different states, and the agent learns the policy indirectly by maximizing the Q value. At the same time, when learning decision-making to improve blockchain performance, it is also necessary to meet the basic security and delay constraints. Therefore, the value function is expressed as the following,

$$\begin{array}{rl} & \max Q(S, A) \\ L1: & T_{Latency} \leq T_{Limit} \\ L2: & K \leq \frac{N}{2N_p} \end{array} \tag{10}$$

where $Q(S, A)$ represents the long-term reward of the system, and T_{Limit} represents the transaction limit delay, its value depends on the type of transaction.

In order to meet the security and delay constraints, the corresponding penalty is set in the immediate reward function, expressed as (11), where $\Xi(t)$ represents the throughput of the blockchain at epoch t, $\Psi(t)$ represents the proportion of cross-shard transactions at epoch t, $\Phi(t)$ indicates the mean square error of the number of transactions in each shard in this round, which is used to indicate the balance of transaction load in each shard. $\Delta(t)$ represents the number of nodes that need to be reconfigured at epoch t,

$$R(t) = \begin{cases} \omega_T \Xi(t) + \omega_C \Psi(t) + \omega_D \Phi(t) + \omega_R \Delta(t), & L1, L2 \\ 0, & otherwise \end{cases} \tag{11}$$

where $\omega_T, \omega_C, \omega_D, \omega_R$ represents the weight of each parameter, and different hyperparameter selections will indicate the agent's emphasis on different indicators. But no matter what, ω_T should always be positive as an incentive, and $\omega_C, \omega_D, \omega_R$ should be negative as a penalty.

We define the throughput of the blockchain as

$$\Xi(S_B, K, T_I) = \frac{K(S_B/\lambda)}{T_I} \tag{12}$$

where λ represents the average size of the transaction. Since a cross-shard transaction will be relayed to another shard, according to the above definition, a cross-shard transaction will be recalculated twice. Therefore, after removing redundancy, the actual throughput in a DSSBD is defined as (13), where g_k represents the number of cross-shard transactions in the kth shard.

$$\Xi(S_B, K, T_I) = \frac{K(S_B/\lambda)}{T_I}[1 - \frac{1}{2}\sum_{k=1}^{K} g_k] \tag{13}$$

And the value function Q of the DSSBD can be expressed as (14), where $\rho \in (0, 1]$ is the discount factor, which is used to reduce the influence of historical decisions.

$$Q(S(t), A(t)) = \left[\sum_{t'=t}^{\infty} \rho^{t-t'} R(t)\Big(S(t), A(t)\Big)\right] \tag{14}$$

4.3.4. Double DQN learning process

At each epoch t, the agent observes the state $S(t)$ of the static environment of DSSBD, which includes the transmission rate $\mu(t)$ between nodes, computing power $\sigma(t)$, and a snapshot of the transaction graph $Y(t)$. Using a greedy algorithm to explore or exploit strategies, the agent selects an action $A(t)$ to determine the block size $S_B(t)$, block interval $T_I(t)$, and the number of shards $K(t)$. After DSSBD executes blockchain consensus and reconfigures based on these decisions, the agent re-observes the static environment of DSSBD and transitions the state to $S(t+1)$. Simultaneously, the agent receives an immediate reward $R(t)$.

Therefore, in order to deal with high-dimensional system states $S(t)$, we need to use deep neural networks. First, we flatten the state space into dimension $1 \times M^2$ data as input, construct a Q deep neural network with three linear layers, using ReLU as activation function of each layer. Specifically, the input layer of the Q deep neural network is a linear layer of dimension $M^2 \times 128$, the hidden layer is a linear layer of dimension 128×128, and the output layer is a linear layer of dimension $128 \times I$, where I is the number of total actions. Finally, it outputs a Q-value table of dimension $1 * I$ for the current state. The decision is made by selecting the action with the highest Q-value from this sub-Q-value table.

In addition, in order to deal with high-dimensional states $S(t)$, we need to use deep neural networks. As shown in Fig. 5, in the Double DQN network, we adopt the method of fixed target network and experience replay (Van Hasselt et al., 2016), improving the stability of model training. Specifically, through the method of fixed target network, during the learning process, the weights of the fixed target Q network are used for policy learning, and the weights of the evaluated Q network are updated using the learned data. After the G step is executed, the weights of the evaluated Q network are copied. Give the target Q network and start a new round of the method of fixed target network. By updating two Q networks, the problem of cumulative learning rewards higher than the actual value in Q-learning is eliminated (Wang et al., 2020). And through the method of experience playback to take advantage of off-policy, after each round of epoch, the agent will track the trajectory in the form of $(S(t), A(t), R(t), S(t+1))$ store it in the playback space D, and then randomly select multiple experiences to train the deep neural network, eliminate the correlation between samples, reduce the variance of parameter updates, and accelerate convergence.

Specifically, DSSBD selects experience d from the experience replay pool each time it is trained, uses the $S(t)$ and $A(t)$ in d to calculate $Q(S(t), A(t))$ through the Policy Q Network in Double DQN, and estimate the action $A(t+1)$ at $S(t+1)$ through the Policy Q Network (Wang et al., 2021),

$$A(t+1) = argmax_A Q(S(t+1), A)$$

Then use $R(t)$ $S(t+1)$ and $A(t+1)$ to calculate the target $Q'(S(t), A(t))$ value at $S(t)$ through the Target Q' Network,

$$Q'(t) = R(t) + \rho Q'(S(t+1), A(t+1))$$

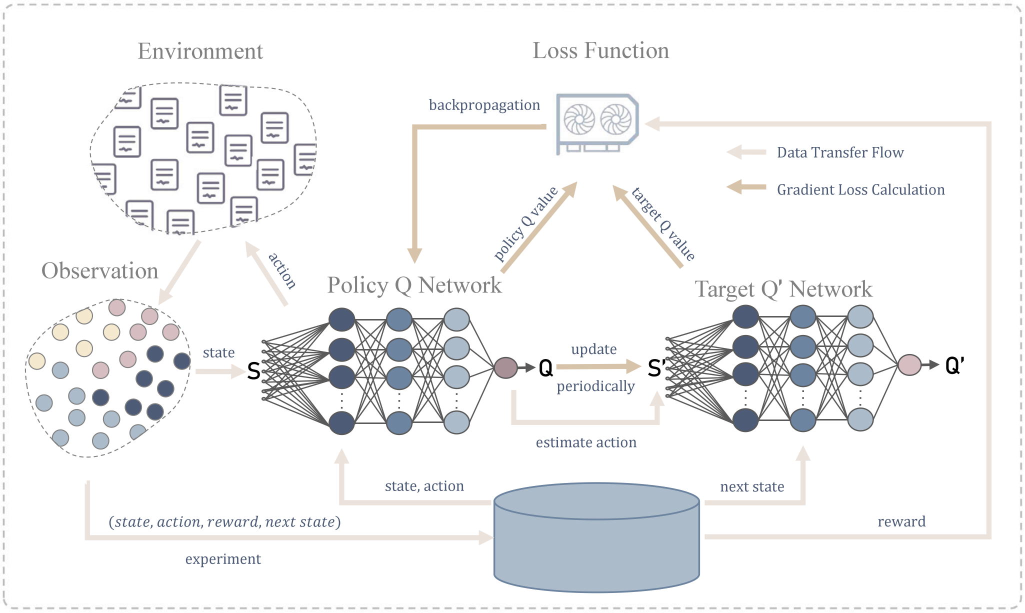

Fig. 5. Double DQN architecture of DSSBD scheme.

Therefore, in order to make the prediction of Q Network more accurate, we need to reduce the prediction difference between the two Q networks, then define the loss function $loss$ as follows,

$$loss = |Q(S(t), A(t)) - Q'(S(t), A(t))|$$

Through this training method, DSSBD can finally dynamically select and adjust the number of shards, block interval time and block size in different states.

5. Security analysis

In this section, we will discuss DSSBD's defense against transaction atomic attacks, double spend attacks, sybil attacks, replay attacks, etc., in blockchain from the perspective of security analysis.

5.1. Transaction atomic attack resistance

The atomicity of intra-shard transactions is naturally guaranteed by the blockchain. For cross-shard transactions, since the DSSBD introduces relay nodes, the atomicity of transactions is naturally affected by relay nodes. If a malicious node becomes a relay node, it can seize the tokens of the sending account. However, as mentioned in Section 4.1.4, due to the existence of the hash transaction time lock, the relay node can only temporarily retain the tokens of the sending node within the valid time. If the malicious relay node does not conduct post-cross-shard transactions or conducts illegal post-cross-shard transactions, the shard where the receiving node is located will send a proof of the failed transaction to the shard where the sending node is located, and eventually form a malicious relay node to return tokens The consensus for sending nodes ensures the atomicity of cross-shard transactions.

5.2. Double spend attack resistance

Double-spending attack refers to the sender using the same digital currency to conduct multiple transactions to obtain services repeatedly. For the DSSBD, it is based on the account address model, so the intra-shard double-spend attack initiated by the sender will not affect the DSSBD. However, due to the introduction of the relay node, the malicious node sender can create a intra-shard transaction tx_{intra} when the relay node creates a pre-cross transaction tx_{pre}. In order to achieve double spending, the sending account nonce of the two transactions must be the same. Considering the nonce counting mechanism, these two transactions cannot be verified as legal at the same time. Therefore, if the pre-cross transaction tx_{pre} is not included in the block, the post-cross transaction tx_{post} will not be created.

5.3. Sybil attack resistance

A Sybil attack refers to malicious nodes manipulating and controlling the blockchain by using multiple accounts or blockchain nodes. For example, in the POW consensus protocol, the attacker needs to occupy more than 51% of the computing power to affect the consensus of the blockchain. In the PBFT consensus protocol, an attacker can affect the consensus of the blockchain by forging multiple accounts. But for the DSSBD, although due to sharding, the security of the blockchain decreases proportionally due to the increase in the number of shards. However, through the initial sharding strategy mentioned in Section 4.1.1 and the combined mining method mentioned in Section 4.1.4, the blockchain can be prevented from being affected by a small number of malicious nodes and the consensus is controlled.

5.4. Replay attack resistance

A replay attack means that after receiving a transaction tx_1, the attacker modifies the timestamp data, obtains a new transaction tx_2 and broadcasts it again, so as to repeatedly obtain the transfer. For the DSSBD, whether it is a relay node or a receiving node, after verifying intra-shard and cross-shard transactions, the counting mechanism will increase the legal transaction nonce of the account by one. In this way, malicious nodes can avoid multiple transfers through replay attacks.

5.5. Other attack resistance

Generally speaking, a blockchain can be divided into data layer, network layer, consensus layer, contract layer and application layer from bottom to top according to user perception. Compared with the traditional blockchain, DSSBD mainly optimizes the structure of the consensus layer. Therefore, for attacks on the data layer, such as collision attacks, its security is still guaranteed by cryptographic tools. For the attack method aimed at the network layer, such as man-in-the-middle attack, its security is still based on the P2P network. As for

Table 2
Security comparison of mainstream state sharding blockchain architectures.

	DSSBD	Monoxide	Brockerchain
Transaction Atomic Attack	✓	✓	✓
Double Spend Attack	✓	✓	✓
Sybil Attack	✓	✓	✓
Replay Attack	✓	✓	✓

Table 3
Simulation parameters.

Symbol	Parameter	Value
$K^{\max}$	Maximum number of shards,	32
$S_B^{\max}$	Maximum block size,	40 MB
$T_I^{\max}$	Maximum block interval,	12 s
M	Maximum number of nodes in each epoch,	2500
N	The number of transactions arriving per second,	2000
λ	Average Transaction size,	200B
ϵ	Exploration probability of Double DQN,	0.99
R	Replay memory buffer size,	1000
ρ	Discount factor,	0.9
lr	Learning rate,	0.01
B	Batch size,	32
E_p^{max}	Maximum epoch per episode,	100
E	Maximum episode,	100

attacks against the contract layer and the application layer, its security is not the focus of this paper.

In summary, as shown in the Table 2,compared to the current mainstream state sharding blockchain architectures Monoxide and Brockerchain, although the implementation methods are different, DSSBD also has the ability to resist the main attack methods at the consensus layer.

6. Performance evaluation

In this section, we verify the performance of the proposed DSSBD. Specifically, in order to simulate the blockchain based on the account model, we built a simulator of DSSBD on the basis of the blockchain model, called Blocksim (Faria and Correia, 2019). Blocksim is a simulator that allows users to customize and expand blocks, transactions, ledgers and network modules. In addition, on the dataset, we use the Block and Transaction dataset of XBlock-ETH (Zheng et al., 2020), which records the transaction data of Ethereum from the genesis block. We build the model based on Pytorch1.12 and Python 3.7, use NetworkX and METIS to divide the graph, and deploy it to Intel(R) Xeon(R) Bronze 3204, NVIDIA GeForce RTX 3090, Ubuntu 20.04 servers. The parameters of the DSSBD simulator are shown in the Table 3, and different initial parameters can be set for different application scenarios.

Among them, the episode refers to a complete running process in deep reinforcement learning, starting from the initial state and reaching the terminal state or meeting the termination condition through a series of actions and decisions. In each episode, the agent learns by interacting with the environment and continuously adjusts its strategy based on the feedback from the environment (reward or punishment), where each interaction with the environment corresponds to one epoch in DSSBD.

6.1. Experimental setup

In order to simplify the model, we assume that all nodes are located in Ireland, Tokyo and Ohio, and set the computing power and transmission rate between nodes in order to fit the actual test situation (Faria and Correia, 2019). Considering that the rate of message transmission in real life has a strong correlation with the region between nodes, we select the basic value of the transmission rate from the Table 4 according to the region where the node is located, and simulate network fluctuations through normal distribution. In addition, we assume that the state of the node's computing power is low, medium, high, and very high, corresponding to the values of the set $\{10, 20, 30, 40\}$ GHz, respectively [29].

Table 4
Transmission rate between regions.

	Tokyo	Ireland	Ohio
Tokyo	80 Mbps	32 Mbps	16 Mbps
Ireland	32 Mbps	90 Mbps	48 Mbps
Ohio	16 Mbps	48 Mbps	100 Mbps

6.2. Experimental indicators

In order to verify the superiority of DSSBD performance, we will select the following indicators for illustration.

- *Throughput*: Throughput is the most direct indicator of the scalability of a blockchain architecture. It refers to the number of transactions or amount of data that a blockchain can process within a unit of time. It is typically measured in transactions per second (TPS).
- *Confirmation Latency*: Confirmation latency is another important indicator of the scalability of blockchain architecture. It refers to the time interval from when a transaction is submitted to the blockchain to when the transaction is confirmed and written into the blockchain. It can measure the transaction speed of the blockchain in seconds.
- *Percentage of Cross-shard Transactions*: In sharded blockchains, the cost of processing cross-shard transactions is higher than that of intra-shard transactions. Therefore, the proportion of cross-shard transactions directly affects the throughput and confirmation latency. In general, reducing this proportion improves the efficiency of the blockchain.
- *Percentage of Nodes Reconfigurated*: In sharded blockchains, each reconfiguration implies the rebuilding of the state for the nodes. Therefore, reducing the proportion of nodes undergoing reconfiguration not only reduces the cost of rebuilding for the nodes but also ensures the consistency and stability of the blockchain.
- *Load Balancing*: In sharded blockchains, load balancing means a high degree of parallelism and decentralization. In this paper, we use the mean square deviation of transaction quantities in each shard to represent transaction load. The smaller the value, the more balanced the load.

6.3. Experimental results

In the proposed DSSBD, transactions between nodes are modeled as data flow snapshots. Blockchain state sharding is implemented using VRF, METIS, and relay transactions. The number of shards, block size, and block interval are combined in Double DQN training. Simulation results demonstrate the performance of our architecture.

6.3.1. Performance experimental evaluation of long-term reward convergence of state sharding mechanism in DSSBD

The Fig. 6 shows the convergence process of the state sharding mechanism in DSSBD, reflecting the relationship between long-term reward and training time. We can observe that the long-term reward keeps increasing as the training epoch increases. In the early training, the long-term reward value of the mechanism increased very quickly, accompanied by shocks. This is because the strategy of the mechanism has a lot of exploration of the environment in the early stage, so the change of the long-term reward value of the model will be more

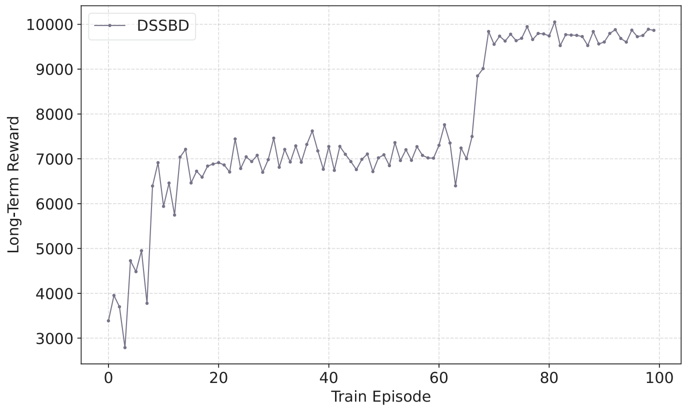

Fig. 6. Long-term reward convergence process of state sharding mechanism in DSSBD.

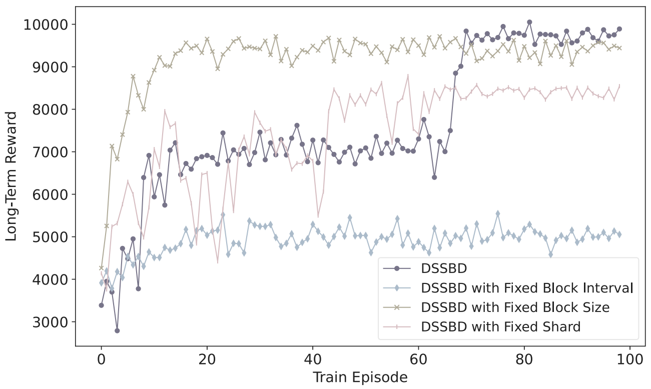

Fig. 7. Long-term reward process of state sharding mechanism in DSSBD with fix different actions.

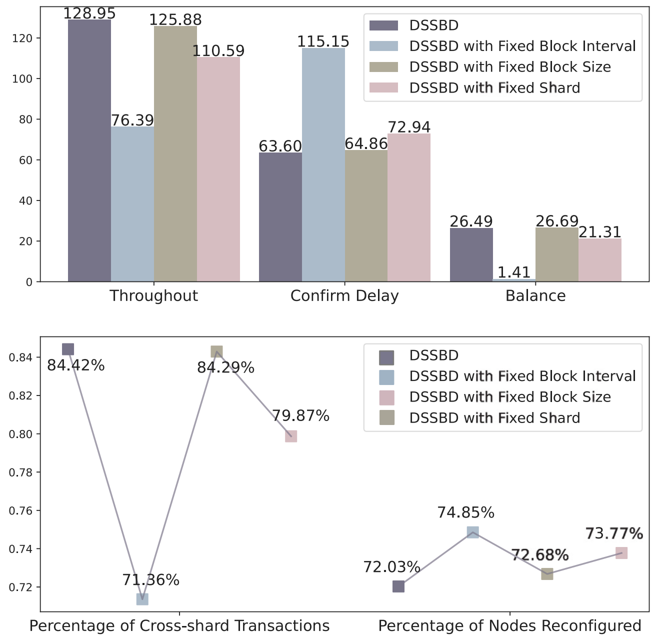

Fig. 8. Performance comparison of DSSBD with different actions fixed.

sensitive. After about 20 times, the long-term reward value of the mechanism changes suddenly and slowly. This is because in the greedy algorithm, as the number of training increases, the mechanism reduces the probability of exploring the environment and is more inclined to use the existing Strategy. After about 70 times of training, the mechanism scheme reached the convergence state, and the long-term reward was stable at about 9730. At this time, Double DQN has learned the optimal strategy to maximize the action state value function. Therefore, it can prove the effectiveness of the state sharding mechanism in DSSBD in the set scenario.

6.3.2. *Experimental evaluation of state sharding mechanism in DSSBD with fixed actions*

In order to illustrate the importance of various actions to the strategy, we fixed the number of state shards in the state sharding mechanism of DSSBD to 16, the block generation interval to 8 s, and the block size to 20 MB, and carried out policy learning in the same environment. As shown in Fig. 7, no matter which action is fixed, compared with the state sharding mechanism of DSSBD, the mechanism of fixed action can reach the convergence state faster, but the long-term reward value after convergence is smaller than that of the state sharding mechanism of DSSBD.

This is because too few shards, too small a block size, and too high a block interval will lead to a decrease in blockchain throughput. Increasing the number of shards, increasing the block size and reducing the block interval can effectively improve the throughput of the blockchain. However, too many shards will increase the proportion of reconfiguration nodes and the proportion of cross-shard transactions. A too short block interval will increase the probability of forks in the blockchain and increase the instability of the blockchain. At the same time, an excessively large block size will also increase the cost of block transmission in the network and increase the propagation delay of transactions. Therefore, after one action is fixed, each fixed scheme can also maximize the long-term reward value under the influence of the other two actions and reach the convergence state. However, compared with the state sharding mechanism of DSSBD, due to the reduction of the dimension of the action space, the long-term reward value of each fixed scheme after convergence will be limited to the upper limit, and the optimal strategy has not been learned. At the same time, it is also the reduction in the dimension of the action space that leads to a corresponding reduction in the state space of each fixed scheme, thereby reducing the learning cost and reaching the convergence state faster.

Observed from the results shown in Fig. 8, the DSSBD with fixed shards has better results, while the performance of the scheme with fixed block size is closest to the DSSBD, which is due to the excessively large block size. It will lead to an increase in block propagation delay, which will lead to smaller blocks reaching consensus and invalidating large blocks. A too small block size will lead to insufficient blockchain throughput. In our experiments, the fixed block size is 20 MB, which happens to be the suboptimal choice. However, the DSSBD is still superior to it in parameters such as throughput, confirmation delay,

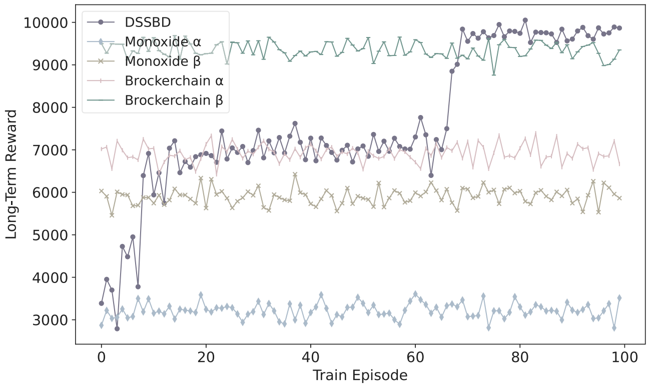

Fig. 9. Long-term reward process of state-of-the-art state sharding mechanisms.

load, and the proportion of reconfigured nodes. The throughput of the scheme with fixed block interval is most affected, and the optimization range is significant, which shows that reducing the block interval is the simplest and most direct means to improve blockchain scalability.

6.3.3. *Experimental evaluation of the performance of the state-of-the-art state sharding mechanism*

Monoxide is currently the most advanced blockchain state sharding mechanism, which simultaneously improves the scalability, decentralization and security of the blockchain through state sharding. Based on Monoxide, Brockerchain adds the reconfiguration steps of shards, which solves the problem of hot accounts and has better performance. However, the Monoxide and Brockerchain algorithms always have only a single decision for the environment. In order to illustrate the superiority of the state sharding mechanism of DSSBD, we experimented with two groups of action decisions with the best performance for Monoxide and Brockerchain. The group α selection action is the number of shards 16, the block generation interval of 8 s, and the block size of 20 MB. The group β selection action is the number of shards 32, the block generation interval of 4 s, and the block size of 40 MB.

As shown in Fig. 9, at the beginning, due to a lot of exploration in the early stage of decision-making of the state sharding mechanism of DSSBD, and even bad decisions may be made, the long-term reward value of the state sharding mechanism of DSSBD did not increase significantly compared with Monoxide and Brockerchain. But after convergence, the long-term reward of the state sharding mechanism of DSSBD floats around 9730, while the long-term reward values of Monoxide and Brockerchain in group α are about 3220 and 6925, and the long-term reward values of Monoxide and Brockerchain in group β are about 5904 and 9350, through policy learning, we made the state sharding mechanism of DSSBD's long-term reward achieve better results.

We compare the average performance of DSSBD after convergence with the two-group average performance of architectures with Monoxide and Brockerchain. As shown in Fig. 10, in terms of throughput, DSSBD increased by 110.1% and 50.5% compared with the two groups of architectures with Monoxide, and increased by 25.8% and 2.2% compared with the two groups of architectures with Brockerchain. In terms of the proportion of cross-shard transactions, DSSBD decreased by 8.87% and 11.1% compared with the two groups of architectures with Monoxide, respectively, and increased by 3.33% and decreased by 2.72% compared with the two groups of architectures with Brockerchain. In terms of the proportion of the number of reconfigured nodes, since architectures with Monoxide does not have a reconfiguration phase, no comparison is made. The proportions of the two groups of Brockerchains decreased by 2.4% and remained basically the same. In terms of transaction load balancing, the average mean square error of each shard transaction of DSSBD is significantly improved compared with the group α, and the performance of architectures with Monoxide and Brockerchain is basically the same compared with the group β.

Table 5
Transaction confirmation delays of different schemes.

Scheme	Confirm Delay
Monoxide α	124.05 s
Monoxide β	77.46 s
Brockerchain α	92.91 s
Brockerchain β	64.54 s
DSSBD	63.72 s

Compared to the two groups of Monoxide, DSSBD has significant advantages in terms of throughput and the percentage of cross-shard transactions. DSSBD achieves this by continuously optimizing the proportion of cross-shard transactions through reconfiguration. In sharding blockchain architectures, reducing cross-shard transactions is crucial for improving throughput. In comparison to Brockerchain, DSSBD shows slight improvements in all indicators for group β. This is because, while Brockerchain also optimizes the proportion of cross-shard transactions during reconfiguration, DSSBD further enhances all indicators through deep reinforcement learning and dynamic selection strategies. Regarding group α of Brockerchain, DSSBD has weaker cross-shard transaction proportions but significant advantages in other performance aspects. This is because, although group α chooses fewer shards to reduce the possibility of cross-shard transactions, its throughput is limited by the number of shards, resulting in lower performance.

The Table 5 shows the transaction confirmation delay in each scheme. The average transaction confirmation delay of DSSBD is reduced by 60.33 s and 13.74 s compared to the previous comparison, and compared with the two groups of architectures with Brockerchain, it is reduced by 29.19 s and 0.82 s respectively.

Regardless of the architecture, the confirmation latency of group β is lower compared to group α. This is because group β has higher throughput and processing efficiency. However, DSSBD has the lowest confirmation latency. However, DSSBD has the lowest confirmation latency due to its lower proportion of cross-shard transactions. The time required to process cross-shard transactions in a blockchain is significantly higher than that of intra-shard transactions. DSSBD achieves this by maintaining a lower proportion of cross-shard transactions while still achieving the highest throughput.

In general, the DSSBD scheme is significantly better than the two groups of Monoxide in terms of throughput, the proportion of cross-shard transactions, the transaction load of each shard, and the transaction confirmation delay. Although the proportion of cross-shard transactions in DSSBD is slightly lower than that of the Brockerchain α architecture, it outperforms Brockerchain α in terms of throughput, proportion of reconfiguration nodes, shard transaction load, and transaction confirmation delay. Compared with the architecture with Brockerchain β, DSSBD is similar in terms of the number of reconfiguration nodes, shard transaction load, and transaction confirmation performance, but it is superior to it in terms of throughput and the number of cross-shard transactions. In short, the experiments prove that the performance of the DSSBD scheme compared with the current optimal baseline can improve the scalability of the blockchain while taking into account multiple performance indicators.

7. Conclusion

In this paper, in order to address the problem of the performance and security of blockchain based crowdsourcing systems being constrained by the underlying blockchain, we propose a dynamic state sharding blockchain architecture, called DSSBD, using deep reinforcement learning. To the best of our knowledge, we are the first one to use deep reinforcement learning to intelligently optimize the state sharding blockchain in terms of shard count, block interval, and maximum block

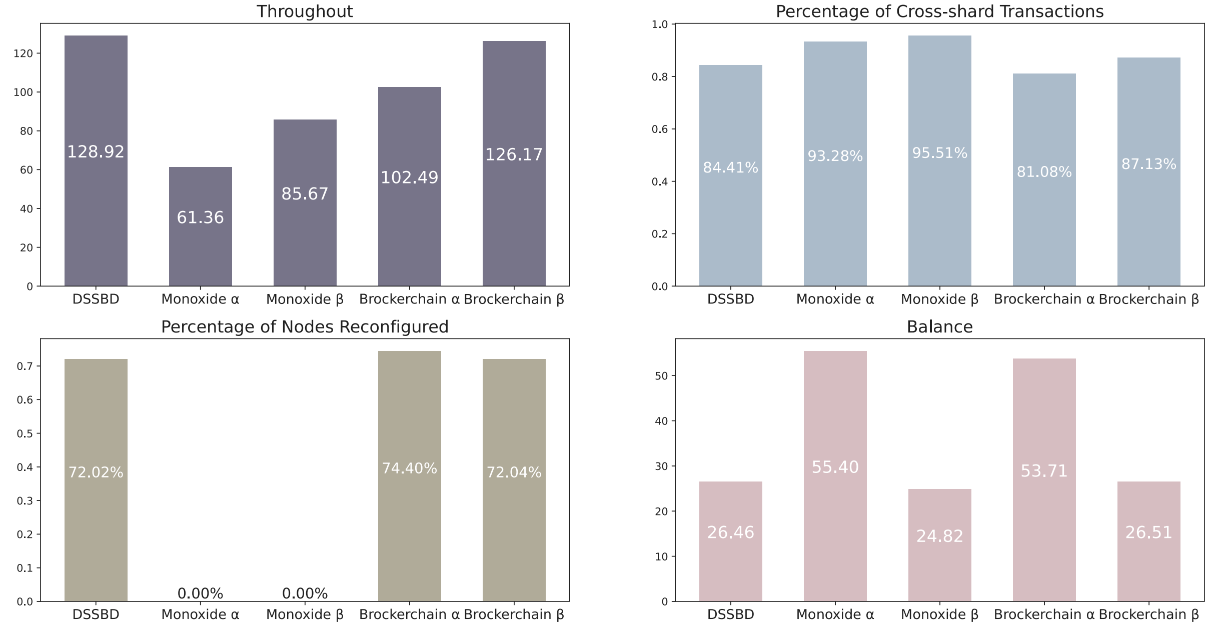

Fig. 10. Performance comparison of state-of-the-art architectures.

size, which greatly improves the scalability of blockchain. Security analysis proves that the DSSBD can effectively resist transaction atomicity attack, double spend attack, sybil attack, replay attack and so on. The experimental results show that the proposed DSSBD has better performance in throughput, latency, balancing, cross-shard transaction proportion, and node reconfiguration proportion, etc., while ensuring security.

In our future work, in order to break through the efficiency bottleneck of the POW consensus protocol, we will consider integrating POS consensus mechanism and dynamically selecting high trust validators in S-Shard. We will also consider combining on chain and off chain scalability technologies to jointly improve the scalability of blockchain.

CRediT authorship contribution statement

Zihang Zhen: Writing – review & editing, Software. **Xiaoding Wang:** Validation, Editing. **Hui Lin:** Project administration, Validation, Supervision. **Sahil Garg:** Resources, Supervision. **Prabhat Kumar:** Validation, Editing. **M. Shamim Hossain:** Supervision, Editing.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the Researchers Supporting Project number (RSP2023r32), King Saud University, Riyadh, Saudi Arabia.

References

Amiri, M.J., Agrawal, D., El Abbadi, A., 2021. Sharper: Sharding permissioned blockchains over network clusters. In: Proceedings of the 2021 International Conference on Management of Data. pp. 76–88.

Buterin, V., et al., 2014. A next-generation smart contract and decentralized application platform. White paper, Vol. 3, (37), pp. 2–1.

Faria, C., Correia, M., 2019. BlockSim: blockchain simulator. In: 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, pp. 439–446.

Gao, Z.-F., Zheng, J.-L., Tang, S.-Y., Long, Y., Liu, Z.-Q., Liu, Z., Gu, D.-W., 2020. State-of-the-art survey of consensus mechanisms on dag-based distributed ledger. J. Softw. 31 (4), 1124–1142.

Hashim, F., Shuaib, K., Zaki, N., 2022. Sharding for scalable blockchain networks. SN Comput. Sci. 4 (1), 2.

Hu, B., Zhou, C., Tian, Y.-C., Qin, Y., Junping, X., 2019. A collaborative intrusion detection approach using blockchain for multimicrogrid systems. IEEE Trans. Syst. Man Cybern.: Syst. 49 (8), 1720–1730.

Huang, H., Peng, X., Zhan, J., Zhang, S., Lin, Y., Zheng, Z., Guo, S., 2022. BrokerChain: A cross-shard blockchain protocol for account/balance-based state sharding. In: IEEE INFOCOM.

Jin, R., Hu, J., Min, G., Mills, J., 2023. Lightweight blockchain-empowered secure and efficient federated edge learning. IEEE Trans. Comput..

Karypis, G., Kumar, V., 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. 20 (1), 359–392.

Klarman, U., Basu, S., Kuzmanovic, A., Sirer, E.G., 2018. Bloxroute: A scalable trustless blockchain distribution network whitepaper. IEEE Internet Things J..

Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., Ford, B., 2018. Omniledger: A secure, scale-out, decentralized ledger via sharding. In: 2018 IEEE Symposium on Security and Privacy (SP). IEEE, pp. 583–598.

Liang, X., Yan, Z., Kantola, R., 2022. GAIMMO: A grade-driven auction-based incentive mechanism with multiple objectives for crowdsourcing managed by blockchain. IEEE Internet Things J. 9 (18), 17488–17502.

Liu, M., Yu, F.R., Teng, Y., Leung, V.C., Song, M., 2019. Performance optimization for blockchain-enabled industrial Internet of Things (IIoT) systems: A deep reinforcement learning approach. IEEE Trans. Ind. Inform. 15 (6), 3559–3570.

Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P., 2016. A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 17–30.

Nasir, M.H., Arshad, J., Khan, M.M., Fatima, M., Salah, K., Jayaraman, R., 2022. Scalable blockchains — A systematic review. Future Gener. Comput. Syst. 126, 136–162.

Tschorsch, F., Scheuermann, B., 2016. Bitcoin and beyond: A technical survey on decentralized digital currencies. IEEE Commun. Surv. Tutor. 18 (3), 2084–2123.

Van Hasselt, H., Guez, A., Silver, D., 2016. Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 30, (1).

Wang, J., Hu, J., Min, G., Zhan, W., Zomaya, A.Y., Georgalas, N., 2021. Dependent task offloading for edge computing based on deep reinforcement learning. IEEE Trans. Comput. 71 (10), 2449–2461.

Wang, J., Hu, J., Min, G., Zomaya, A.Y., Georgalas, N., 2020. Fast adaptive task offloading in edge computing based on meta reinforcement learning. IEEE Trans. Parallel Distrib. Syst. 32 (1), 242–253.

Wang, J., Wang, H., 2019. Monoxide: Scale out blockchains with asynchronous consensus zones. In: 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19). pp. 95–112.

Wang, W., Wang, Y., Duan, P., Liu, T., Tong, X., Cai, Z., 2022. A triple real-time trajectory privacy protection mechanism based on edge computing and blockchain in mobile crowdsourcing. IEEE Trans. Mob. Comput..

Yang, Z., Yang, R., Yu, F.R., Li, M., Zhang, Y., Teng, Y., 2022. Sharded blockchain for collaborative computing in the internet of things: Combined of dynamic clustering and deep reinforcement learning approach. IEEE Internet Things J..

Yun, J., Goh, Y., Chung, J.-M., 2020. DQN-based optimization framework for secure sharded blockchain systems. IEEE Internet Things J. 8 (2), 708–722.

Zamani, M., Movahedi, M., Raykova, M., 2018. Rapidchain: Scaling blockchain via full sharding. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 931–948.

Zhang, S., Lee, J.-H., 2019. Double-spending with a sybil attack in the bitcoin decentralized network. IEEE Trans. Ind. Inform. 15 (10), 5715–5722.

Zheng, P., Zheng, Z., Wu, J., Dai, H.-N., 2020. Xblock-eth: Extracting and exploring blockchain data from ethereum. IEEE Open J. Comput. Soc. 1, 95–106.

Zhou, Q., Huang, H., Zheng, Z., Bian, J., 2020. Solutions to scalability of blockchain: A survey. IEEE Access 8, 16440–16455.

Zhu, S., Cai, Z., Hu, H., Li, Y., Li, W., 2019. zkCrowd: a hybrid blockchain-based crowdsourcing platform. IEEE Trans. Ind. Inform. 16 (6), 4196–4205.

Zihang Zhen received the bachelor's degree in mechanical engineering from Inner Mongolia University of Science and Technology, Baotou, China, in 2019. He is currently pursuing the master's degree with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China. His research interests include blockchain, deep learning and cyber security.

Xiaoding Wang received the Ph.D. degree from the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China, in 2016. He is an Associate Professor with the College of Computer and Cyber Security, Fujian Normal University, FuZhou, China. His main research interests include network optimization and fault tolerance.

Hui Lin received the Ph.D. degree in computing system architecture from the College of Computer Science, Xidian University, Xi'an, China, in 2013. He is a Professor with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China, where he is currently an M.E. Supervisor. He has published more than 50 papers in international journals and conferences. His research interests include mobile cloud computing systems, blockchain, and network security.

Sahil Garg received his Ph.D. degree from the Thapar Institute of Engineering and Technology, Patiala, India, in 2018. He is currently an Adjunct Associate Professor at École de Technologie Supérieure, Montréal, Canada. He has many research contributions in the area of machine learning, big data analytics, security and privacy, the Internet of Things, and cloud computing.

Prabhat Kumar received his Ph.D. degree in Information Technology, National Institute of Technology Raipur, Raipur, India, under the prestigious fellowship of Ministry of Human Resource and Development (MHRD) funded by the Government of India in 2022. Thereafter, he worked with Indian Institute of Technology Hyderabad, India as a Post-Doctoral Researcher under project" Development of Indian Telecommunication Security Assurance Requirements for IoT devices". He is currently working as Post-Doctoral Researcher with the Department of Software Engineering, LUT School of Engineering Science, LUT University, Lappeenranta, Finland. He has many research contributions in the area of Machine Learning, Deep Learning, Federated Learning, Big Data Analytics, Cybersecurity, Blockchain, Cloud Computing, Internet of Things and Software Defined Networking. He has authored or coauthored over 25+ publications in high-ranked journals and conferences.

M. Shamim Hossain (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Ottawa, Ottawa, ON, Canada, in 2009. He is currently a Professor with the Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. He is also an Adjunct Professor with the School of Electrical Engineering and Computer Science, University of Ottawa. He has authored or coauthored more than 350 publications. His research interests include cloud networking, smart environment (smart city, smart health), AI, deep learning, edge computing, the Internet of Things (IoT), multimedia for health care, and multimedia big data. He is a Distinguished Member of the ACM. He was a recipient of a number of awards, including the Best Conference Paper Award, the 2016 ACM Transactions on Multimedia Computing, Communications and Applications (TOMM) Nicolas D. Georganas Best Paper Award, and the 2019 King Saud University Scientific Excellence Award (Research Quality). He is the Chair of the IEEE Special Interest Group on Artificial Intelligence (AI) for Health with IEEE ComSoc eHealth Technical Committee. He is the Symposium Chair of Selected Areas in Communications (eHealth) with IEEE GLOBECOM 2024. He served as the Technical Program Co-Chair of ACM Multimedia 2023. He is also the Chair of the Saudi Arabia Section of the Instrumentation and Measurement Society Chapter. He is on the Editorial Board of the IEEE Transactions on Instrumentation and Measurement, IEEE Transactions on Multimedia, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), IEEE Multimedia, IEEE Network, IEEE Wireless Communications, IEEE Access, and Journal of Network and Computer Applications (Elsevier). He has served as a Lead Guest Editor for more than two dozen of Special Issues (SIs), including ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), ACM Transactions on Internet Technology, IEEE Communications Magazine, IEEE Network, IEEE Transactions on Information Technology in Biomedicine (currently JBHI), IEEE Transactions on Cloud Computing, and Future Generation Computer Systems (Elsevier). He is an IEEE Distinguished Lecturer (DL). He is the Highly Cited Researcher in the field of Computer Science (Web of Science).