Hindawi
Security and Communication Networks
Volume 2020, Article ID 8881881, 20 pages
https://doi.org/10.1155/2020/8881881

Research Article

A Secure and Effective Construction Scheme for Blockchain Networks

Chaoxia Qin,[1] Bing Guo,[1] Yan Shen,[2] Tao Li,[1] Yun Zhang,[3] and Zhen Zhang[1]

[1]*College of Computer Science, Sichuan University, Chengdu, Sichuan 610065, China*
[2]*School of Control Engineering, Chengdu University of Information Technology, Chengdu, Sichuan 610225, China*
[3]*School of Statistics, Southwestern University of Finance and Economics, Chengdu, Sichuan 611130, China*

Correspondence should be addressed to Bing Guo; guobing@scu.edu.cn

Received 16 September 2020; Revised 21 November 2020; Accepted 28 November 2020; Published 10 December 2020

Academic Editor: Omar Cheikhrouhou

Blockchain technology has emerged as a novel distributed ledger technology, facilitating data sharing and system management securely and efficiently without interventions from a central authority. However, blockchain technology alone is not suitable for enterprise-class applications, mainly due to the limitations in capacity expansion and verification speed of blockchain systems. This paper proposes a secure and effective construction scheme for blockchain networks to improve performance and address the effective management concerns of blockchain data based on transaction categories. We designed a network link protocol to construct a directed acyclic graph (DAG) blockchain network and used a sharding protocol to divide the DAG blockchain into multiple category shards to process transactions in parallel. We then extensively evaluated our proposed design on local clusters. The experimental results show that our link and shard protocols achieved high throughput and the category-based sharded DAG blockchain demonstrated high scalability.

1. Introduction

The traditional data storage and acquisition method has been questioned due to data privacy leakage and the "data island" dilemma in the big data industry. In the era of all-around digitalization, managing data information to improve the efficiency of data and decrease potential misuse poses a formidable challenge for all kinds of enterprises and even national governments.

Among existing technologies of data management, blockchain provides a promising solution for problems in data management. Unlike cloud storage [1], blockchain operates a distributed, trustless, nonintermediary, and tamper-resistant business mode, enhancing the security of blockchain systems and guaranteeing data credibility. The early blockchain model was limited to issuing and trading digital currencies, with bitcoin as a typical product [2]. The second generation of blockchain opens the multiapplication era of blockchain by introducing smart contracts and programmable bottom layers [3]. Traditional blockchain has introduced a profound impact on human society [4–6]. In reality, blockchain is leveraged to strengthen the access control of personal data, ensuring that users own and utilize their data without having to worry about privacy disclosures [7–9]. A blockchain-enabled electronic health system provides efficient and flexible electronic medical record management, breaking the "barrier" of multisource heterogeneous data management platforms [10, 11]. The Internet of Things (IoT) system based on blockchain is able to remotely control and configure IoT devices that collect data information and track resource assets in real time [12, 13]. The blockchain-based notarization service can ensure transparency, immutability, and nonrepudiation of data in a potentially Byzantine failure environment [14, 15].

Traditional blockchain with a single chain is characterized by high security and low misoperation rate. This mode largely attributes success to the application of consensus mechanism, which mainly includes Proof of Work (PoW) and Proof of Stake (PoS). However, this mode scales and performs poorly due to consensus participation, data synchronization, multiple backups, and redundant network loading. Currently, the bitcoin network can only handle seven transactions per second (i.e., 7 TPS, transactions per second), Ethereum has an average speed of about 25 TPS, and Hyperledger Fabric can handle about 1000 TPS [16]. Compared with Alipay's 256k TPS [17], the performance and availability of blockchain lag considerably and would need to be improved.

Directed acyclic graph (DAG) blockchain is a multichain network composed of transactions [18], which is essentially similar to traditional single-chain blockchain since DAG transactions can be viewed as "blocks" (referred to as nodes or transaction nodes in this paper). Aside from the network structure, there are two other significant differences between DAG and single-chain blockchain: (1) the record form of transaction (also known as data state or state in blockchain) and (2) the consensus mechanism. On the one hand, the traditional single-chain blockchain first packages many transactions into one block and maintains the order between blocks through hash in a block header to achieve a consensus on the transaction sequence. However, the DAG blockchain abandons the concept of "block" and allows each transaction to initially participate in the maintenance of the transaction sequence [18]. In theory, the DAG blockchain is much more efficient since it skips the stage of packing blocks. On the other hand, the single-chain blockchain uses the consensus mechanism based on synchronous communication for global sorting, which means the consensus delay is longer, typically 60 minutes in bitcoin [2] and 3 minutes in Ethereum [3]. In contrast, the DAG blockchain adopts the DAG consensus mechanism [19], in which consensus delay reaches the millisecond level under a high transaction arrival rate because it addresses asynchronous communication and concurrent processing concerns [20]. Therefore, DAG blockchain is taking the core role in data management technology due to its high performance [21].

The fundamental idea of blockchain sharding [22] is to divide the nodes in the blockchain network into relatively independent shards. A single shard deals with small-scale transactions (or even only stores part of the network states), and multiple shards deal with transactions in parallel to increase the transaction throughput of the whole network. However, the absence of application value when applying shard techniques to blockchain only depends on the number of nodes. Disordered sharding rules can increase transaction costs caused by frequent cross-shard communication since users would have to traverse all shards to locate the latest data state. Therefore, we propose dividing the blockchain network nodes into different shards based on transaction categories. Each category shard deals with one kind of transaction processing and data storage. For transactions with multiple categories, users only need to load involved shards. The kind of high cohesion and low coupling network structure can improve the validation speed and transaction throughput.

The purpose of this paper is to improve the blockchain performance and address the classification management concerns of blockchain data to promote security as much as possible. A novel DAG-type blockchain network model is proposed, which is different from existing works in terms of the following (see Table 1). First, the link rules of transaction nodes are modified. In previous models, Byteball [23] and IOTA [18] adopt the simplest and most basic form to randomly link the latest nodes into the Tangle network. Nano [24] uses all nodes belonging to the same account to construct a relatively independent single chain. In the proposed model, nodes of the same category are linked together as much as possible. Second, the sharding rules of the network in the proposed model are significantly improved. In particular, our shard protocol is category-based, which organizes data according to the category to reduce the frequency of cross-shard communication. Third, we changed the network structure. Given that IOTA is a Tangle structure and Nano is a lattice structure, our category-based DAG network is a tree tangle structure, as shown in our simulation experiments.

(1) To our knowledge, we are the first to propose a DAG blockchain model supporting classification management of blockchain data.

(2) We are also the first to propose the link and shard protocols of a DAG blockchain based on transaction categories, which satisfies data-intensive blockchain workloads.

(3) We adjusted the DAG consensus mechanism to conform to a tree tangle blockchain, which ensures the overall consistency of the blockchain data.

(4) We conducted simulation experiments to evaluate the performance of our design. The experimental results show that the category-based sharded DAG blockchain performed well in network loading and transaction throughput.

The major innovations and contributions of this study are as follows:

The rest of this paper is structured as follows. Section 2 reviews several major sharded blockchains and DAG-type blockchains. Section 3 introduces the proposed blockchain model. Section 4 describes our protocols in detail, including a link protocol and a sharding protocol. Section 5 provides security analysis. Section 6 presents a performance evaluation of our methods. Section 7 summarizes and concludes our work.

2. Related Work

2.1. Sharded Blockchain. Sharding was first proposed by Hua and Lee [25] and is now applied to blockchain to expand the limitations of transaction throughput. Generally speaking, there are three types of sharding: network sharding, transaction sharding, and state sharding (see Table 2). Network sharding (also known as physical sharding) divides the whole blockchain network into multiple groups, with each group (called shard) composed of multiple servers. All shards process different transactions simultaneously to parallelize bookkeeping. In transaction sharding, transactions are assigned into different shards according to transaction hash or account addresses or by means of

Table 1: Comparison between our work and existing work.

System	Network construction	Network structure	Sharding	State model	Node time	Smart contracts	Bookkeeping nodes
Byteball [23]	Based on time	Tangle	No	Account	0.5 minutes	Yes	Whole network
IOTA [18]	Based on time	Tangle	No	UTXO	Instant	No	Whole network
Nano [24]	Based on account	Block lattice	No	Account	Instant	No	Account chain
Our work	Based on category	Tree tangle	Based on category	Account	Instant	Yes	Category shard

Table 2: Comparison of sharding based on their representative blockchain systems.

System	Sharding type	Consensus mechanism	Cross-shard communication	State model	Capacity	TPS	Project phase
Zilliqa	Network; transaction	PoW; PBFT	Main-chain driven	Account	Low	2,828 TPS with 3600 nodes and 6 shards [26]	Test network online
QuarkChain	Network; transaction; state	PoW	Main-chain driven; Kademlia routing	Account	High	318,052 TPS with 512 shards [27]	Test network online
Elrond	Network; transaction; state	SPoS (Secure Proof of Stake)	Binary tree regulation	Account	High	3750+ TPS in a single shard [28]	Test network online
Monoxide	Network; transaction; state	PoW	Shard driven; Chu-ko-nu mining	Account	High	11,694 TPS with 2,048 shards [29]	Theoretical research
Near	Network; transaction; state	Doomslug; nightshade finality gadget	Shard driven; Merkle proof	Account	High	1000 TPS with 8 shards, 24 block/block producers and 800 verifiers [30]	Test network online
Our work	Network; transaction; state	PoW; root shard; main chain	Shard driven; Merkle proof	Account	High	Depends on the number of concurrent threads without upper limit in theory	Theoretical research

verifiable random function (VRF). For state sharding, part of ledger information is stored and maintained in each shard. This is the most complex way of sharding since it may involve cross-shard communication, cross-shard trading, and other issues.

Zilliqa [31] is a sharded blockchain system which is redesigned from scratch. Network sharding110 and transaction sharding are both applied to Zilliqa to shard the network into smaller node groups. Each node group is called a shard and handles different transactions. In Zilliqa, each node must save all the states of the blockchain to process transactions successfully. As the blockchain's transaction volume increases, the transaction costs and the performance requirements of the node equipment also increase.

QuarkChain [32] expands blockchain capacity by introducing state sharding based on network and transaction sharding since each node only needs to record part of the data and states to handle specific transactions. QuarkChain consists of a two-tier blockchain. The first tier is the slicing layer used for transaction bookkeeping, while the second tier is a root chain responsible for confirming transactions in the slicing. Without affecting the root chain, the number of slices can be increased dynamically to improve the overall throughput of the system, which is expected to achieve 100,000 + TPS. Root chains account for a large proportion (more than 50%) of the total network computing power, which means that malicious miners would need at least $50\% \times 51\% > 25\%$ computing power to perform attacks.

Elrond [33] also adopts the full-state sharding mechanism, focusing on performance and adaptability. It puts forward a novel "adaptive state sharding" technology, which performs calculation and reorganization about shards according to the number of computing tasks and online network nodes. Compared with the static sharding technology, the main challenge lies in effectively addressing the time latency issues caused by data communication and synchronization during sharding adjustment. Elrond proposes a significant solution using a binary tree to cut apart the account address space. This is conducive to accurately grasp the number of shards in each epoch and reduce the workload and delay of system sharding.

Entities, such as scheduling nodes or verifying nodes, can make the implementation of some functions easier (e.g., dynamic workload balancing) since they can master the whole situation. However, this may sacrifice decentralization and can lead to serious performance bottlenecks. Monoxide [29] advocates for a simple design principle as much as possible. On the premise of ensuring security and performance, Monoxide neither tries to introduce additional entities and mechanisms nor improves existing mechanisms. As a concurrent multichain system, each chain in Monoxide has its own account book status, blockchain, a group of unconfirmed transactions, and full nodes (including miners). Beyond that, this network has no other role nor any supernode.

Near [34] is a scalable public chain based on full-state sharding. Unlike other sharded blockchains composed of one beacon chain or multiple shard chains, Near is a single chain sharded at the block level. Each block in Near stores logically all states of the whole network and is cut into multiple pieces. Similar pieces from all the blocks comprise a complete shard. For example, the Near network with 20 blocks/block producers and 1000 verifiers is divided into ten shards. This means that each block is divided into ten pieces, and each piece requires 100 verifiers to confirm that there are no problems among the 20 similar pieces in one shard.

2.2. Consensus Mechanism. The consensus mechanism of DAG-type blockchains mainly includes Tangle, Witness, Coordinator, and Delegated Proof of Stake (DPoS) (see Table 3).

Byteball [23] and IOTA [18] are the well-known DAG-type blockchains in the industry that adopt Tangle consensus. Each transaction of Byteball confirms one or more previous transactions, while IOTA must confirm two existing transactions. Byteball uses transaction fees to prevent Sybil [36] and Denial of Service (DoS) [37] attacks, while IOTA draws support from PoW consensus [2].

Theoretically, if someone can initiate transactions amounting to 51% (POW consensus) or 33% (Practical Byzantine-Fault-Tolerant, namely, PBFT consensus [38]) of the entire network, it would be possible to make invalid transactions effective. To solve this problem, Byteball adopts 12 witnesses to generate the “main chain” in DAG by serializing transactions in order to achieve an overall consensus. For IOTA, a central node called “the coordinator” is used to “double-check” all transactions in the network until the Tangle network reaches a certain scale, making it difficult for individuals to reach 51% or 33% of all transactions.

DPoS [39] is the consensus mechanism used in Nano [40]. The general idea is that users elect a deputy to handle the bifurcation. The deputy first broadcasts the bifurcation to all users and then complies with the voting results from high-power users for a fixed time range to determine which block to retain. DPoS can ensure the reasonable operation of blockchain with low energy consumption to substantially reduce the number of nodes participating in the validation and bookkeeping.

3. Proposed Sharded DAG Blockchain

In this paper, we present the model design of a sharded DAG blockchain and the implementation of its network construction. Through directed acyclic graph (DAG) technologies, we transformed the blockchain single-chain structure into a bifurcated graph structure. Based on sharding technology, we transformed the single-chain distributed storage into multishard parallel distributed storage. Our method improves the scalability and throughput of blockchain systems and facilitates the orderly management of massive data. Our scheme also enhances the controllability and security of the whole network through a shard-driven communication mode and Merkel verification mechanism. The category-based DAG blockchain network construction theory brings opportunities for new design ideas, components, and operation modes in the blockchain system. We first introduce the fundamentals and prerequisites to lay the foundation and contextualize the improvements in our proposed model.

3.1. Blockchain Entity. DAG-type blockchain entities mainly refer to user, data, and transaction. Users are external entities that use blockchain infrastructure to transfer assets or run smart contracts. They are represented as *user* = *(address, public-key, private-key)*, where *address* identifies blockchain users while *public-key* and *private-key* are the security technology to protect user accounts [41]. Concretely, *public-key* and *private-key* are a key pair obtained by an algorithm [42]: *public-key* is the public part of the key pair that can be disclosed to all nodes without any potential risk, and *private-key* is the nonpublic part which will cause property loss and information disclosure once exposed. In addition, the *public-key*, *private-key,* and *address* are essentially a string of 0 or 1 and have the generation relationship [43], as shown in Figure 1.

Blockchain transaction is a kind of data structure used to unify and standardize the representation of blockchain data, given that it packs extracted original data into a transaction record. As shown in Figure 2, the data structure is represented as *transaction* = *(index, timestamp, from_account, to_account, data[], pre_index{}, code, category, nonce)*, where *index* is the index (or digital fingerprint) of a transaction, *index* = *hash(timestamp, from_account, to_account, data[], pre_index{}, code, category, nonce)*; *timestamp* indicates the time of the transaction, *timestamp* = *time()*; *from_account* represents the *address* of the data seller; and *to_account* represents the *address* of the data buyer. *pre_index* is the set of parent node indexes of a transaction; *code* is the smart contract program; *category* is the classification of a transaction, *transaction_category* ∈ *{dataItem_category}*; and *nonce* is the counter to ensure that each transaction is executed in sequence.

Data[] is a dynamic list containing multiple data items, expressed as *data* = *[{ID, owner, value, describe, URL, category, nonce} {} ⋯]*, denoting the original data in a transaction or the asset information of users. *Data_ID* = *hash (owner, value, describe, URL, category, nonce)* represents the index (or digital fingerprint) of each data item; *owner* refers to the data owner (after the close of trading); *value* is the value quota of digital currency or resource-type data; *describe* is a brief description of the data item; *URL* refers to the uniform resource location of the data item; *category* is the classification of the data item; and *nonce* is a counter used to make sure the data consistency. The data sources are diverse, including physical assets, electronic documents, network data, digital currency, and even work IP, patent number, and URL address.

3.2. Model Design. The category-based DAG blockchain model is shown in Figure 3. The whole system mainly includes five components: shards, smart contracts, a category

Table 3: Comparison of consensus mechanisms based on their representative DAG-type blockchains.

System	Consensus mechanism	Transaction fee	State model	TPS	Node time	Genesis node
Byteball [23]	Main chain; witnesses	Yes	Account	10 [23]	0.5 minutes	Oct. 21, 2015
IOTA [18]	Tangle; coordinator	No	UTXO	182.8 [35]	Instant	Sept. 5, 2016
Nano [24]	DAG; DPoS	No	Account	7000 [35]	Instant	Feb. 29, 2016
Our work	PoW; root shard; main chain	No	Account	Without upper limit in theory	Instant	—

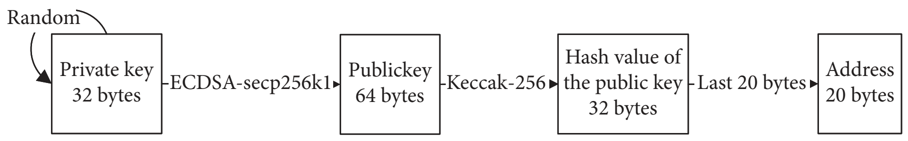


Figure 1: The generation relationship between the *public-key*, *private-key*, and *address*. Step 1: randomly generate a *private-key*; step 2: generate a *public-key* through the *private-key*; step 3: obtain the *address* by intercepting the last 20 bytes of the *public-key*.

```
transaction = {index: (256bits) transaction hash
               timestamp: (64 bits)
               from_account: (160 bits) account address
               to_account: (160 bits) account address
               data: [{ID: (256bits) data hash; owner: (160 bits) account address;
                      value: (unlimited); describe: (unlimited);
                      URL: (unlimited); category: (unlimited); nonce: (64 bits)}
                      {} ......]
               prev_index: (512 bits) index list
               code: (unlimited) contract code
               category: (unlimited)
               nonce: (64 bits)
            }
```

Figure 2: Data format of transaction. A transaction is regarded as a node which is linked to previous nodes.

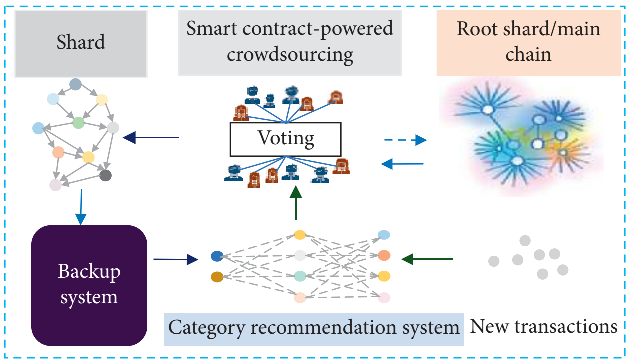

Shard
Smart contract-powered crowdsourcing
Root shard/main chain
Voting
Backup system
Category recommendation system
New transactions

Figure 3: DAG blockchain model based on transaction categories.

recommendation system, a set of consensus rules, and a backup system. Transaction processing can be divided into four stages:

(1) The category recommendation system assigns a specific category and shard to new transactions.

(2) System verifies and confirms transactions through crowdsourcing driven by smart contracts.

(3) System appends new transaction nodes to the corresponding shard.

(4) System updates the root shard and main chain periodically.

Shard is a part of the blockchain system employed to store ledgers, connect participants, and broadcast transactions. This paper adopts a full-state sharding technique to shard the DAG blockchain network into a root shard and multiple category shards. The shard, where the genesis node is located, is the root shard, and category shards are derived from the division of the root shard, whose number has no theoretical upper limit:

$$\begin{aligned} \text{root\_shard} &\longleftarrow G^0\left\{V^0, E^0\right\}, \\ \{\text{shard\_1}, \text{shard\_2}, \ldots\} &\longleftarrow \text{root\_shard}. \end{aligned} \tag{1}$$

Category recommendation system is an information filtering system that predicts a transaction category according to the categories of its data items. The system appends new transaction nodes to the corresponding shard.

$$\begin{aligned} t\_\text{category} &\longleftarrow \{t\_\text{dataItem\_category}\}, \\ \text{shard\_category} &\longleftarrow \text{shard\_category} \cup \text{node\_new}. \end{aligned} \tag{2}$$

However, if the number of nodes in a shard reaches the limit, the recommendation system appends the subshard sequence to the category value of the new transaction. The system then temporarily assigns it into a category that would require shard splitting:

$$\begin{aligned} t\_\text{category} &\longleftarrow \text{append}\left(t\_\text{category}, \text{sub\_shard\_sequence}\right), \\ \text{shard\_category} &\longleftarrow \text{shard\_category} \cup \text{node\_new}. \end{aligned} \tag{3}$$

If the recommendation system fails to find this category, it gives the new node a new category (deriving from the *dataItem_category*). The system then assigns it to the root shard to be split:

$$\begin{aligned} t\_\text{category} &\longleftarrow \{t\_\text{dataItem\_category}\}, \\ \text{root\_shard} &\longleftarrow \text{root\_shard} \cup \text{node\_new}. \end{aligned} \tag{4}$$

Note that the category recommendation here only considers the basic situation that a transaction only involves one data item category. The complex situation that a transaction involves multiple data categories will be discussed later.

Smart contract-driven crowdsourcing is a set of smart contracts used to publish information, such as task requirements, contract provisions, and project delivery, to blockchain in the form of programs to realize the automatic processing of transaction logic. Crowdsourcing collaboration eliminates third parties, making the source task open and transparent. The automated execution of smart contracts addresses the supervision issues of contract execution status, including the trigger control for tasks and access to private data. The crowdsourcing driven by smart contract is mainly divided into four steps:

(1) Build the crowdsourcing powered by smart contract:
   (a) Define the user set for smart contract, such that $U = \{u_1, u_2, \ldots, u_n\}$, and n represents the number of participating users including witnesses.
   (b) Release task requirements, such that $T = \{t_1, t_2, \ldots, t_m\}$, and m represents the number of task requirements.
   (c) Publish contract rules, such that $S = \{s_1, s_2, \ldots, s_p\}$, and p represents the number of contract rules.

(2) Store the smart contract: the smart contract is broadcasted to each user through the P2P network and stored in the blockchain's corresponding shards.

(3) Execute the smart contract: the smart contract verifies the transaction whether it meets necessary conditions and automatically begins to execute after reaching a consensus among participating users.

(4) Deliver project: the smart contract periodically checks the completion of tasks and the execution status of contracts and links the transaction into appropriate shards after confirming that the transaction is closed correctly.

According to our design, the blockchain transaction is directly confirmed by participating users (including witnesses) without the need for consensus of the whole network or confirmation of subsequent transactions. However, it ensures that the third party is excluded, since the transaction is initiated and supervised by the participating users and the delivery is performed automatically after the contract tasks are completed. The security is equivalent to IOTA since the link relationship of the Tangle structure among the transaction nodes is still preserved.

Root shard/main chain is a set of consensus strategies for the time of the whole network. The optimal path from any top node to the genesis node is called the candidate main chain. Different candidate main chains will cross at a certain node (the worst case is at the genesis node), referred to as the stable node. For all candidate main chains, the path from the stable node to the genesis node is the same, known as the stable main chain [23]. Sharded DAG blockchain contains many parallel tangle chains with a stable main chain in each tangle, referred to as "shard main chain." The shard main chain is an unbranched path on which all nodes can be sorted. This sequence number is termed "SMCI" (shard main chain index), which reaches a consensus on the transaction order in each shard. The root shard retains the initial transaction node of each category and ranks these transaction nodes to reach the consensus of the global network.

Backup system refers to a large-scale storage device that replicates the data of the electronic computer storage and stores it separately to deal with unexpected situations, such as data loss or damage. Compared with traditional blockchain breakdowns caused by the collapse of more than 50% of nodes, the failure of one shard may affect more shards, such as refusing to respond and deleting shard information and data, which can be dangerous. To improve stability, we suggest building an additional adaptive backup system. In particular, the blockchain system selects some extra supernodes to build a P2P storage system (e.g., IPFS (interplanetary file system)) [44]. This storage system is dedicated to backing up blockchain transactions and data. The backup system is also self-adaptive such that (1) it only backs up the data of the smaller shard system; (2) when the shard system reaches a safe scale, the backup system will automatically release the shard storage space to back up new shard data; and (3) it maintains the backup situation adaptively to ensure basic backup requirements to prevent data loss. When the shard system is offline or reports a timeout response, the backup system responds to the request. When the backup system fails, the blockchain system reconstructs and rewrites data into the new backup system.

3.3. Data Sharing.

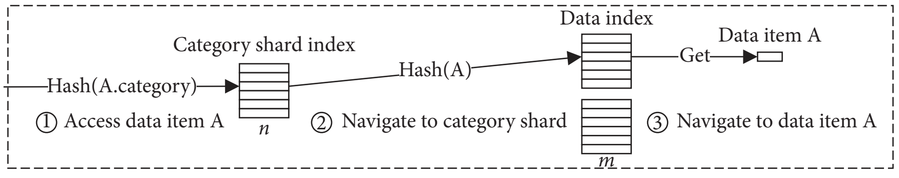
According to the design of "root shard" and "category shard" in Section 3.2, our category-based DAG blockchain system mainly uses the "three-step" scheme to share data among shards. The specific data sharing mechanism is as follows:

(1) Construct a hash index table for the root shard. According to our sharding mechanism, each shard of the blockchain system backs up the genesis node of the new shard to the root shard. This would allow for the quick determination of the specific category shard based on the hash index table of the root shard.

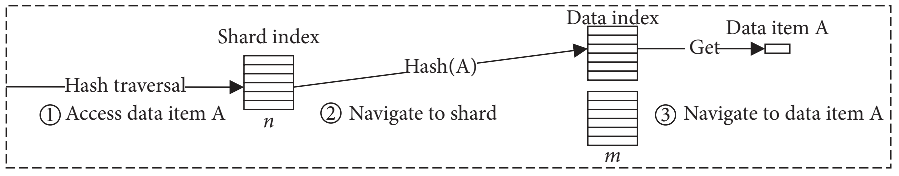
(2) Construct a hash index table of the category shard. After successfully locating the corresponding category shard, we use the hash index table of the category shard to further locate the specific data.

(3) Implement data sharing: the system would first need to verify and authorize the permissions and then establish a Transmission Control Protocol (TCP) connection to start data transmission.

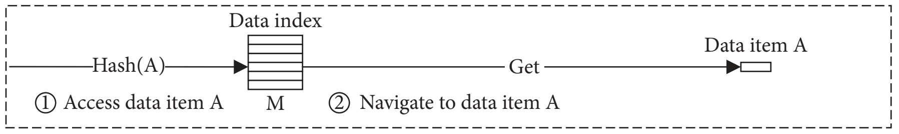
In Figure 4, the data retrieval mechanism for category-type sharded blockchain systems is compared with the unordered-type sharded blockchain systems and blockchain systems without shards. In the figure, n is the number of shards, m is the number of transactions in a shard, and all hash index tables are stored in the memory. The time complexity of a hash search is $O(1)$, and the space complexity is $O(n)$ [45]. Based on this, we can get



(a)



(b)



(c)

Figure 4: Data retrieval mechanism of blockchain systems. (a) Category-type sharded blockchain system. (b) Unordered-type sharded blockchain system. (c) Block chain system without shards.

(1) The retrieval time complexity of category-type sharded blockchain systems is the time complexity for category shard retrieval plus the time complexity for data retrieval, i.e., $O(1) + O(1) = O(1)$. The space complexity is the space complexity of category shard retrieval plus the space complexity of data retrieval, that is, $O(n) + O(m) = O(n + m)$.

(2) In an unordered-type sharded blockchain, the system cannot directly determine the shard of the target data and would need to traverse the shard index. In general, the average time complexity is $((n + 1)/2) \times O(1) \longrightarrow O(n)$. The space complexity is the space complexity of shard traversal plus the space complexity of data retrieval, that is, $O(n) + O(m) = O(n + m)$.

(3) The retrieval time complexity of the blockchain systems without shards is $O(1)$, and the space complexity is $O(M)$, where M is the number of all transactions in the whole blockchain system.

In general, category-based sharded blockchain systems show the best data retrieval performance, in which the time delay is fewer than that for unordered sharded blockchain systems, and the memory cost is distinctly smaller compared to blockchain systems without shards.

A hash list provides fast insert and query operations. However, the hash list needs to set the array size in advance and does not support dynamic expansion since it is implemented based on the array. In addition, the hash list has the problems of hash conflict and unable to sort. Merkle tree [46] is a generalized hash list, which is widely used in blockchain systems, such as bitcoin, Ethereum, and Hyperledger. It can not only ensure the integrity and consistency of data but also support the download and verification of branch data. Ethereum proposes an improved Merkle tree (Merkle Patricia tree) [47] to organize state data so as to achieve fast data validation and status data update. Qu et al. [48] further improved the Merkle Patricia tree by adding spatiotemporal information for transactions, which enriches the query functions of logistics management systems. Our blockchain design abandons the concept of block and aims to achieve data classification management of information systems. Moreover, Merkle tree is an unbalanced tree pattern. Therefore, it is not suitable for directly applying above index strategies to our system. Further research on data indexing and sharing will be carried out in the future.

3.4. Cross-Shard Trading. Even among independent blockchain systems, the demand for interoperability between blockchains is extremely high. Cross-shard trading must be considered in the category-based sharding mechanism.

Suppose a simple trading scenario: Alice exchanges her house A for Bob's stock C, which involves two data categories and shards (house and stock). Synchronous trading means that when a cross-shard transaction is executed, blocks are simultaneously generated in multiple shards containing related transactions [49]. This trading method was used since our proposed system can quickly retrieve and locate the corresponding data states. The system can also package and back up the transaction to multiple relevant shards after confirming the transaction promptly. The synchronization mode has better security, but it needs to maintain fast cross-shard data retrieval capabilities.

The asynchronous mode refers to the asynchronous execution of a cross-shard transaction affecting multiple shards. For example, the property right authentication and transfer processing of house A are completed in the house shard, while the stock authentication and transfer processing are completed in the stock shard. At present, Cosmos [50], Ethereum serenity [51], Near [34], and Kadena [52] suggest asynchronous cross-shard trading because for existing sharding mechanisms, the asynchronous mode is simpler, and the cooperation between regions is easier.

4. Protocol Design

4.1. Link Protocol

4.1.1. Link Mechanism. The link mechanism makes the nodes from the same category close to each other in physical space, which helps the segmentation in DAG blockchain networks. In a DAG-type blockchain, new transaction nodes will be generated whenever registering or trading data, and the transaction category is decided by data items in this transaction. The category types involved and the number of nodes can be generated by

$$\begin{aligned} t\_\text{category} &\longleftarrow \{t\_\text{dataItem\_category}\},\\ \text{node\_new} &\longleftarrow \text{pack}(t). \end{aligned} \tag{5}$$

For example, Alice uses house A and car B under her name to exchange stock C under Bob's name. This transaction involves three categories (i.e., house, car, and stock). The content of this transaction is shown in Figure 5, and the blockchain states after this transaction are shown in Figure 6. As a result, three nodes with different categories would generate three category shards (house, car, and stock):

$$\begin{aligned} \text{shard\_house} &\longleftarrow \text{shard\_house} \cup \text{node\_new}(\text{house}),\\ \text{shard\_car} &\longleftarrow \text{shard\_car} \cup \text{node\_new}(\text{car}),\\ \text{shard\_stock} &\longleftarrow \text{shard\_stock} \cup \text{node\_new}(\text{stock}). \end{aligned} \tag{6}$$

Some category shards may not exist during the trading process because nodes of the same category have to reach a secure scale before being separated from the root shard to form a relatively independent category shard. The new transaction node is assigned to the root shard and linked to the back of nodes of the same category. In extreme cases, all new nodes associated with a transaction are assigned to the root shard. The generation mechanism for multiple shard nodes is conducive to constructing a high cohesion and low coupling network. This kind of redundant bookkeeping method also improves the speed of state verification since a category shard stores all data states of a specific category. As shown in Figure 6, the house shard always keeps track of all the states of housing data. This mechanism also supports synchronous trading, which avoids failure atomicity in asynchronous trading [53–55].

4.1.2. Link Algorithm. The link algorithm is the link logic between transaction nodes and is the last step in the transaction process of the proposed model. This algorithm has nothing to do with the transaction content, state verification, and transaction confirmation. The detailed link algorithm (as shown in Algorithm 1) is as follows:

(1) The recommendation system provides the category of a transaction node.

(2) The system is positioned to the corresponding shard:

   (i) If the number of nodes with the same category is greater than or equal to 2, randomly select two of the latest ten nodes as parent nodes and chain the new node into the shard.

   (ii) If there only exists one node of the same category, select this node as one of the parent nodes and randomly select another latest node as the second parent node and then chain the new node into the shard.

   (iii) If there are no nodes of the same category, randomly select two latest nodes as parent nodes and link the new node into the shard.

(3) Update the backup system and regularly update the root shard and the main shard chain.

4.2. Shard Protocol

4.2.1. Sharding Mechanism. In the early stage of a blockchain system, the network scale is small. Thus, we let all the nodes link to the root shard to reduce the risk of a 51% attack:

$$\text{root\_shard} \longleftarrow G^0\{V^0, E^0\}. \tag{7}$$

In the middle stage, once nodes of a specific category in the root shard reach a secure scale, we start up the sharding mechanism to separate these nodes. To record the global series of all category shards, we need to back up and store the genesis node of each category shard to the root shard:

$$\begin{aligned} \{\text{root\_shard}, \text{shard\_category}, \ldots\} &\longleftarrow G(V, E),\\ \text{root\_shard} &\longleftarrow \text{root\_shard} \cup \text{genesis\_category\_node}. \end{aligned} \tag{8}$$

In the later stage, when the scale of the category shard becomes too large, divide the category shard into multiple new category shards according to sequence tags for transaction nodes. Then, link copies of the genesis nodes of the subcategory shards into the root shard:

$$\begin{aligned} \{\text{shard\_category}, \text{shard\_category\_1}, \ldots\} &\longleftarrow \text{shard\_category},\\ \text{root\_shard} &\longleftarrow \text{root\_shard} \cup \{\text{sub\_genesis\_category\_node}\}. \end{aligned} \tag{9}$$

Based on this network construction theory, the root shard always keeps the sharding records of the whole network, and all shards are dynamic and divisible.

4.2.2. Sharding Algorithm. The key of sharding is to cluster category shards, which is realized through community discovery algorithms. Traditional community discovery algorithms mainly include graph segmentation theory [56, 57], random walk [58, 59], label method [60], and hierarchical clustering [61]. New community discovery algorithms mainly refer to community overlapping [62], density theory [63], distance theory [64], and statistical inference [65].

```
transaction = {index: transaction hash
               timestamp:
               from_account: Alice
               to_account: Bob
               data: [{ID: A; owner: Bob; value; describe; URL; category: house; nonce}
                      {ID: B; owner: Bob; value; describe; URL; category: car; nonce}
                      {ID: C; owner: Alice; value; describe; URL; category: stock; nonce}]
               pre_index: {pre_index_0, pre_index_1}
               code: smart contract
               category: house | car | stock
               nonce:
  }
```

Figure 5: Transaction content of node. The initiator is Alice and the responder is Bob.

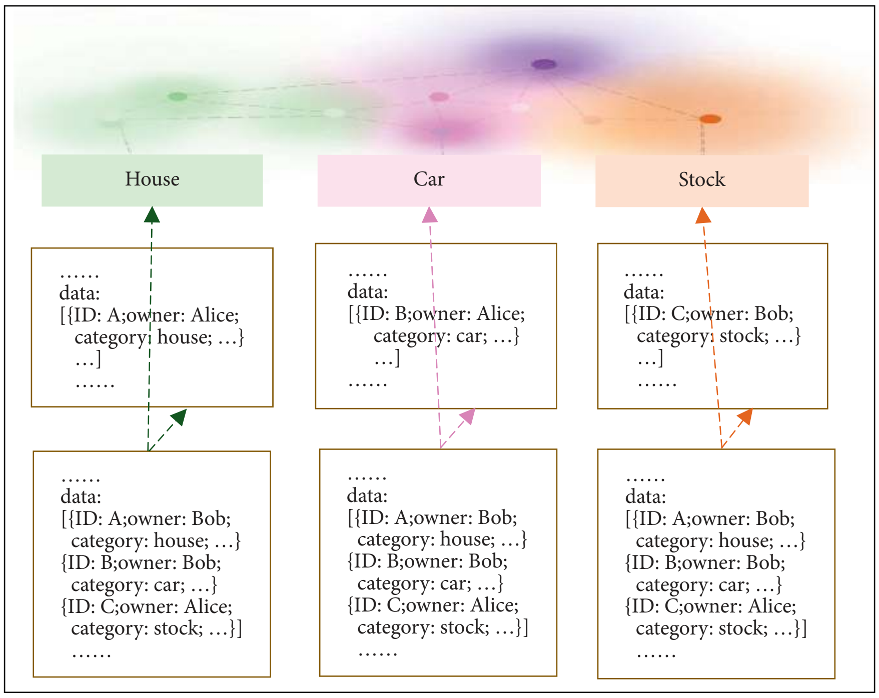


Figure 6: Network status of DAG blockchain. Shards (house, car, and stock) generate a transaction node.

According to our link mechanism, each node has a specific category, and the link relationship between nodes of the same category is compact while the relationship between nodes of different categories is loose. Clustering-based and label-based community discovery algorithms are both suitable for this network structure. However, the clustering algorithm is better than the label method in terms of performance and effect since it avoids substantial label index maintenance. Louvain algorithm [61] is one of the most popular clustering algorithms, which is an unsupervised algorithm that does not require the number of communities or their sizes before execution as input. The detailed process of the Louvain algorithm is as follows:

(1) Initialization: each node in the graph is regarded as an independent community, and the number of communities is the same as the number of nodes.

(2) First phase––node transfer:

   (i) For each node i, to allocate node i to the community of each neighboring node, calculate the modularity change (expressed as Q) before and after allocation and then record the neighboring node with $\max(Q)$.

   (ii) If $\max(Q)$ is greater than 0, assign the node i to the community where the neighboring node with $\max(Q)$ is located; otherwise, abandon this round of division.

   (iii) Repeat steps a and b until the nodes are no longer transferred.

(3) Second phase––graph rebuilding: treat all nodes in the same community as a new node. Take the sum of edge weights of the inner community as the ring weight of the new node and the edge weights between communities as the edge weights between the new nodes.

```
**Input**: transaction //input a transaction
**Output**: $v_{\text{new}}$ //generate a new node
$(V_i, E_i)$: the $i$th shard of the DAG network
$E(v_i)$: all edges connecting to vertex $i$
(1) $v_{\text{new}}$.category = recommendation system (transaction)
(2) $i = 1$
(3) for $V_{\text{index}}$ in $V$ do //shard positioning
(4)     if $V_{\text{index}}$.category == $v_{\text{new}}$.category
(5)         $i$ = index //category shard
(6)     else
(7)         $i = 0$ //root shard
(8) end for
(9) $j = \text{len}(V_i)$
    //Finding the latest ten nodes
(10) for $v_{--j}$ in $V_i$ do
(11)     if $v_j$.category == $v_{\text{new}}$.category:
(12)         $C_{\text{same}} \longleftarrow v_j$
(13)     else:
(14)         $C_{\text{other}} \longleftarrow v_j$
(15)     if $\text{len}(C_{\text{same}}) \geq 10$ & $\text{len}(C_{\text{other}}) \geq 10$
(16)         break
(17) end for
    //Finding parent nodes
(18) if $\text{len}(C_{\text{same}}) \geq 2$:
(19)     $v_{\text{pre\_0}}, v_{\text{pre\_1}} = \text{random}(C_{\text{same}})$
(20) elif $\text{len}(C_{\text{same}}) == 1$:
(21)     $v_{\text{pre\_0}} = \text{random}(C_{\text{same}})$
(22)     $v_{\text{pre\_1}} = \text{random}(C_{\text{other}})$
(23) else:
(24)     $v_{\text{pre\_0}}, v_{\text{pre\_1}} = \text{random}(C_{\text{other}})$
    //Node linking
(25) $V_i \longleftarrow v_{\text{new}}$
(26) $E_i \longleftarrow e(v_{\text{new}}, v_{\text{pre\_0}} \cup v_{\text{pre\_1}})$
(27) $v_{\text{new}}$.pre_index $\longleftarrow v_{\text{pre\_0}} \cup v_{\text{pre\_1}}$
(28) return $v_{\text{new}}$
```

Algorithm 1: Category-based link algorithm.

(4) Repeat the first/second phase until Q of the whole graph no longer changes.

After obtaining the clustering results of category shards through the Louvain algorithm, we start to shard the DAG network, which means filtering out the directions of all non-self-graphs. The process of the sharding algorithm (see Algorithm 2) is as follows:

(1) Get the community map of the DAG networks.
(2) Filter out links between shards.
(3) Copy genesis category nodes and chain them into the root shard.
(4) Return sharded networks.

5. Security Analysis

The innovation of blockchain technologies must take system security as premise. This section briefly analyzes the security of the category-based sharded DAG blockchain.

5.1. Consistency. DagCoin [66] and Byteball [23] apply the "main chain" mechanism to solve the data consistency problem. However, there are some differences between Byteball, DagCoin, and our category-based DAG blockchain in terms of network architecture (see Figure 7). Byteball and DagCoin networks use a tangle structure (see Figure 7(a)), where nodes are closely linked and a unique main chain is present (marked in dark black). However, based on our simulation network and theoretical analysis, the category-based DAG blockchain consists of a tree tangle, as shown in Figure 7(b). The relationship between different node categories is very loose, making it difficult to find the same main chain from different branches to the genesis node. Therefore, we use "root shard" to sort nodes globally in a wide and unordered directed acyclic network without high maintenance costs since "root shard" always keeps a small network scale. The single category shard network is also a tangle structure, which means the "main chain" mechanism is suitable for category shards. This kind of "main shard chain" mechanism forms a unique ordered single chain in each

```
Input: $G^0 = (V^0, E^0)$ //initial directed graph. $V^0$ is the initial set of vertices and $E^0$ is the initial set of edges.
Output: $G(V, E)$
(1) $G(V,E) = \text{Louvain}\ (G^0)$
(2) for $V_i$ in $V$ do //category shard
    //Cutting off connections between subgraphs
(3)     for $v_j$ in $V_i$ do
(4)         parent_index $\longleftarrow v_j$.pre_index $\cap V_i$
(5)         $E(v_j) \longleftarrow e(v_j, \text{parent\_index})$
(6)     end for
    //Copying genesis category nodes
(7)     $c = V_{i,0}$ //genesis node of shard $i$
(8)     $j = \text{len}(V_0)$ //root shard
(9)     for $v_{--j}$ in $V_0$ par-do
(10)        if $v_j$.category $==$ $c$.category:
(11)            $C_{\text{same}} \longleftarrow v_j$
(12)        else:
(13)            $C_{\text{other}} \longleftarrow v_j$
(14)        if $\text{len}(C_{\text{same}}) \geq 10$ & $\text{len}(C_{\text{other}}) \geq 10$
(15)            break
(16)    end for
    //Node linking and root shard updating
(17)    if $\text{len}(C_{\text{same}}) \geq 2$:
(18)        $c_{\text{pre\_0}}, c_{\text{pre\_1}} = \text{random}(C_{\text{same}})$
(19)    elif $\text{len}(C_{\text{same}}) == 1$:
(20)        $c_{\text{pre\_0}} = \text{random}(C_{\text{same}})$
(21)        $c_{\text{pre\_1}} = \text{random}(C_{\text{other}})$
(22)    else:
(23)        $c_{\text{pre\_0}}, c_{\text{pre\_1}} = \text{random}(C_{\text{other}})$
(24)    $V_0 \longleftarrow c$
(25)    $E_0 \longleftarrow e(c, c_{\text{pre\_0}}) \cup e(c, c_{\text{pre\_1}})$
(26)    $c$.pre_index $\longleftarrow c_{\text{pre\_0}} \cup c_{\text{pre\_1}}$
(27) end for
(28) return $G(V, E)$
```

Algorithm 2: Category-based sharding algorithm.

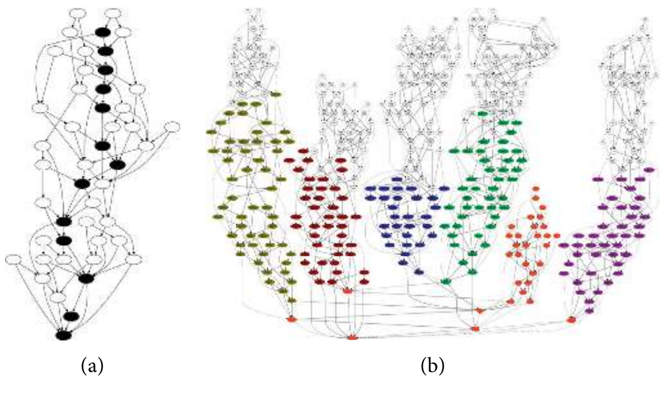


Figure 7: Network structure comparison of different DAG blockchains. Different colors represent different categories. (a) Tangle structure. (b) Tree tangle structure.

branch of the category-based DAG blockchain in order to reach consensus of data consistency within the category shard.

5.2. DDoS Attack. Distributed Denial of Service (DDoS) attack [67] means that the attacker uses multiple puppet hosts to send reasonable but useless requests simultaneously to consume much of the network resources so that legitimate users cannot promptly receive service responses. There are two principal difficulties to solve DDoS attacks: (1) these attacks may be difficult to identify because the requests initiated by the attacker are reasonable; (2) these attacks are difficult to locate since the attacker uses multiple IPs to launch the attacks. Our crowdsourcing mechanism, driven by smart contracts, addresses this problem due largely to the supervision of witnesses in smart contracts. In our design, the blockchain system builds a smart contract user set (including witnesses) for each transaction, in which the witnesses monitor the request of each user in real-time, find out the abnormal data traffic (such as DDoS attacks), and address them without affecting other normal businesses.

5.3. Double Spend. Double-spend attack (also known as 51% attack) [68] refers to user attempts to spend the same digital cryptocurrency on the blockchain twice. Blockchain adopts a decentralized governance mode, which does not require a third party, such as a bank or government, to guarantee transactions, making it possible for users to tamper with transaction records with computing power. For example, after a user spent some digital cryptocurrency on a stock, he launches a 51% attack on the blockchain, wherein he tries to establish a fake chain to replace the honest chain. If the user succeeds, he will own both the stock and the digital currency.

The essential cause of the double-spend attack is the delay in transaction confirmation. Our automatic delivery mechanism, powered by smart contracts, can address this issue since the payment of a transaction is instantaneously settled according to the contract settings. The crowdsourcing system establishes a task set and contract set for each transaction and regularly checks the contract's execution state and the task completeness status. Once the user completes all the required tasks correctly, the contract program will automatically perform the transaction delivery. At the same time, the transaction is confirmed without waiting for verification of the subsequent nodes, which means that the transaction will never be deleted or tampered with after it is successfully generated.

6. Experiment

6.1. Experiment Introduction. We conducted experiments on a computer with a MacOS 10 system, an inner core 6 i7 CPU, and 16 GB DDR4 RAM. Our link algorithm was used to construct blockchain networks based on the DAG library [69] in Python 3, ensuring no closed-loop generation. We then used our sharding algorithm to divide the blockchain network with a graph plugin called Graphviz [70] to visualize network graphs (see Figure 7). Finally, we tested the performance of the category-based sharded DAG blockchain on a local area network (LAN) with 100 Mbps bandwidth. The size threshold of each shard ledger was set to 300 (i.e., each shard ledger must have at least 300 transaction nodes).

6.2. Experiment Result

6.2.1. Algorithm Performance. We implemented algorithm stress testing based on four indicators: throughput, latency, concurrency, and scalability. The following is a brief description of the concepts involved.

Throughput refers to the number of transaction requests processed by a system (software) in unit time, reflecting the system's pressure bearing capacity (software). TPS is a common quantitative index of throughput.

Time delay refers to the time that a system (software) processes a transaction request and generally takes the average value from multiple tests.

Concurrency number is the number of transactions (or working threads) simultaneously processed by a system. In the python development environment, a client only supports one thread (concurrency) by default. This experiment uses the number of clients to represent the number of threads (concurrency).

Scalability in this experiment refers to the ability of the algorithm to adapt to the changes in the number of users, which is reflected in the change of algorithm performance with the increase of network devices.

Peak memory usage represents the maximum amount of memory a process has used from initialization.

According to Figure 8(a), as the number of transactions increases, the throughput of the link and sharding algorithms generally decreased, from 1000 TPS to 8.03 TPS and from 1000 TPS to 0.36 TPS. In Figure 8(b), the link algorithm's delay increased from 0.001 s to 0.125 s, while the delay in the sharding algorithm grew from 0.001 s to 2.81 s. This is because, according to our setting, each client had only one thread (concurrency), which means the processor can only process one transaction request at a time. High numbers of transactions can preempt memory space (see Figure 8(c)) and limit CPU I/O speed [71]. Aside from allocated memory for transactions, more transactions also mean requiring a larger network structure and segmentation workload for the sharding algorithm. Thus, the sharding algorithm yielded a worse throughput performance in terms of the number of transactions. The results also show that the throughput of the link algorithm reaches its peak at transaction volume 10, which can vary according to the machine's main memory's access cycle and I/O speed (Box 1).

Figure 9(a) shows that in terms of the number of transaction categories, the throughput for the link and sharding algorithms increased from 8.03 TPS to 149.10 TPS and from 0.38 TPS to 0.65 TPS, while in Figure 9(b), the latency decreased from 0.125 s to 0.0067 s and from 2.61 s to 1.53 s. As the number of categories increased, the system network generated more bifurcations, which resulted in stronger concurrency. Figure 9(c) shows that the peak memory consumption of the link and sharding algorithms was not affected by the number of categories (Box 2).

In Figure 10(a), when the number of clients in the blockchain network increased, the throughput of the link and sharding algorithms increased clearly, while the delay in both algorithms decreased significantly, as shown in Figure 10(b). This is because multiple clients share the computing tasks in the blockchain system and the clients' maximum load pressure (see Figure 10(c)). The experiment shows that with the increase of users (client devices) in the blockchain network, the system has a markedly performance expansion, which suggests that the overall system

Experimental settings:
Quantitative: one shard; one client (single thread); 20 transaction categories
Variable: number of transactions
Experiments: Figures 8(a), 8(b), and 8(c)

Box 1: The first group of experiments: throughput and latency.

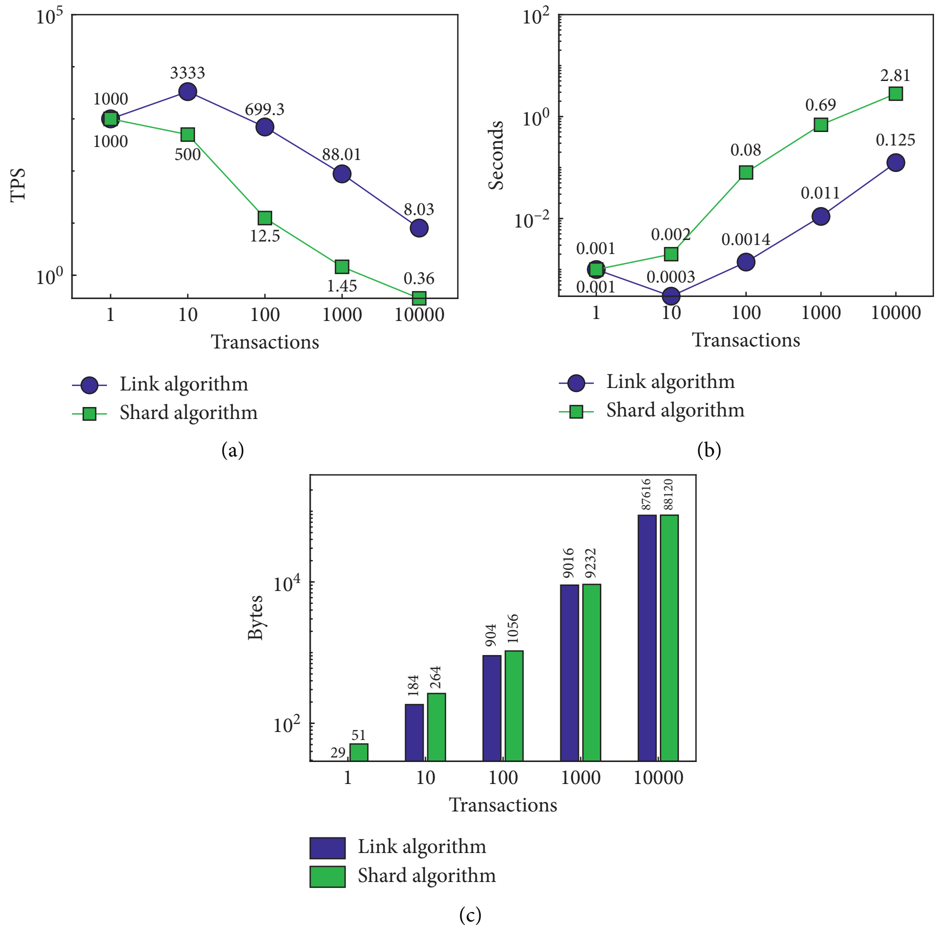


Figure 8: Performance of the link and sharding algorithms in terms of transactions. (a) Throughput. (b) Latency. (c) Peak memory cost.

Experimental settings:
Quantitative: one shard; one client (6 threads); 10000 transactions
Variable: number of categories
Experiments: Figures 9(a), 9(b), and 9(c)

Box 2: The second group of experiments: concurrent capability.

performance is not limited by the performance limit of a single client device but is improved by the increase in the number of clients (Box 3).

6.2.2. Network Load of Ledger. Network performance usually refers to the network's service quality, which is the key evaluation metric of software system availability. Many factors affect network performance, such as network bandwidth, hardware system, code quality, data content, and network scale. In terms of network scale, we analyzed the network load performance of the blockchain ledger, including load latency and memory cost (Box 4).

The experimental results show that the load delay and memory cost of the ledger network increase linearly with the number of transactions in the network, but the loading cost of

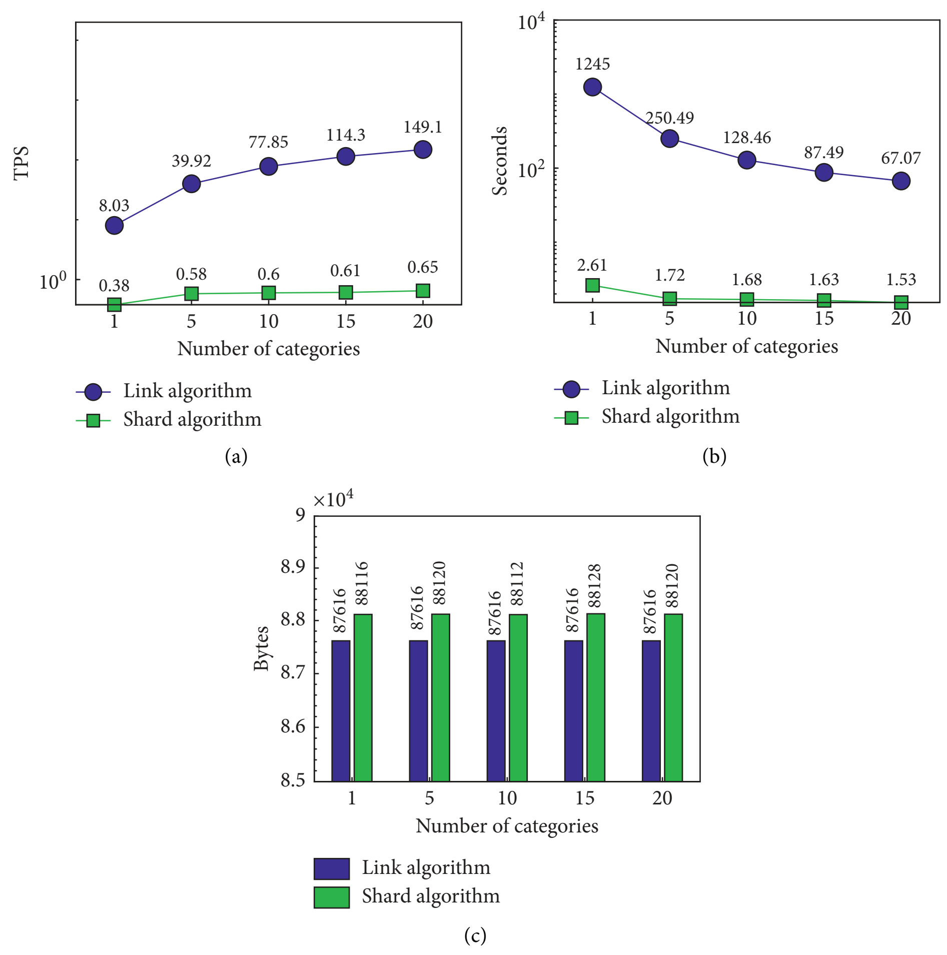


Figure 9: Performance of the link and sharding algorithms in terms of categories. (a) Throughput. (b) Latency. (c) Peak memory cost.

Experimental settings:
Quantitative: one shard; 10000 transactions; 20 transaction categories
Variable: number of clients (concurrency)
Experiments: Figures 10(a), 10(b), and 10(c)

Box 3: The third group of experiments: extending capability.

the whole network is significantly higher than that of the shard network (see Figures 11(a) and 11(b), respectively). These results suggest that the sharding scheme is conducive to improving the loading performance of the blockchain ledger.

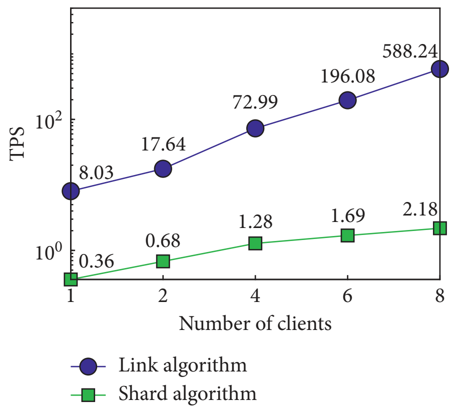
6.2.3. Performance of Data Sharing. We tested and analyzed the data-sharing performance of the blockchain systems in terms of data retrieval (Box 5) and data transmission (Box 6). The following is a brief description of the concepts involved:

Data retrieval is the operation to find the target data by scanning the ledger of blockchain systems, which is the basis of data sharing.

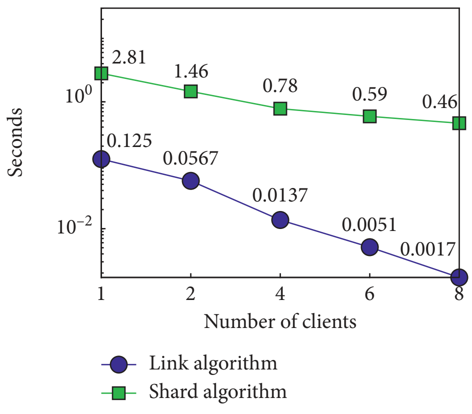
Data transmission refers to the operation of transmitting data from the source to the terminal through one or more data links according to certain procedures. Its main function is to realize the information transmission and exchange between ends, which is a critical step in data sharing.

Figures 12(a) and 12(c) show that time delays in data retrieval of the category-based sharded blockchain system do not vary with the number of transactions or shards, validating the theoretical derivation of time complexity $O(1)$ in Section 3.3. The time delay in data retrieval of an unordered sharded blockchain system increases linearly with the change of shards, consistent with the previous theoretical derivation of time complexity $O(n)$, where n represents the number of shards.

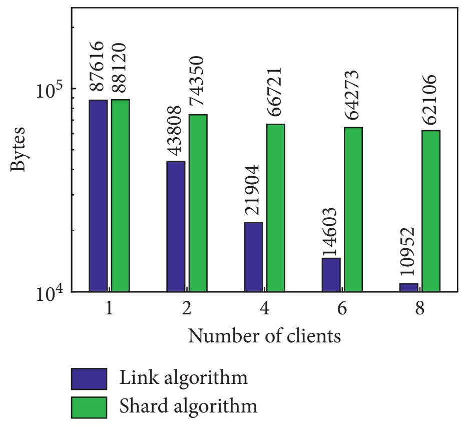
In Figure 12(b), the memory cost of data retrieval in the category-based sharded blockchain system is equal to that of



(a)



(b)

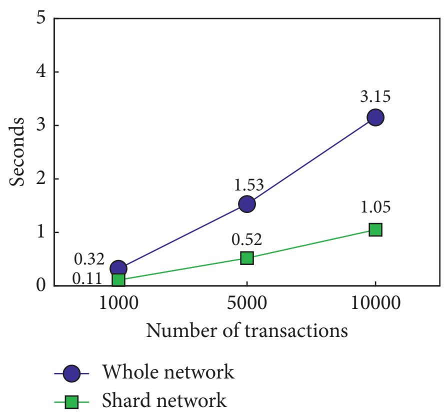


(c)

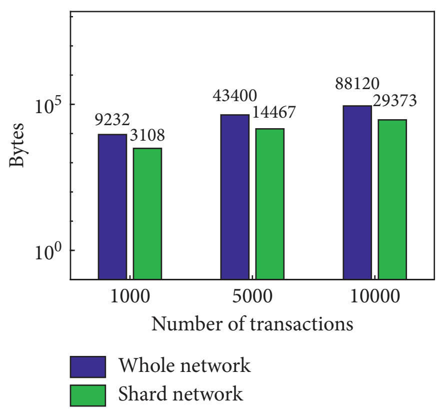
Figure 10: Performance of the link and sharding algorithms in terms of clients. (a) Throughput. (b) Latency. (c) Peak memory cost.

Experimental settings:
Quantitative: one client; whole system without shards; sharded system with three shards
Variable: number of transactions
Experiments: Figures 11(a) and 11(b)

Box 4: The first group of experiments: network load.



(a)



(b)

Figure 11: Loading performance of ledger of blockchain systems. (a) Latency. (b) Peak memory cost.

(1) **Experimental settings**:
Quantitative: one client; 20 shards
Variable: number of transactions
Experiments: Figures 12(a) and 12(b)
(2) **Experimental settings**:
Quantitative: one client; 5000 transactions
Variable: number of shards
Experiments: Figures 12(c) and 12(d)

Box 5: The first group of experiments: data retrieval.

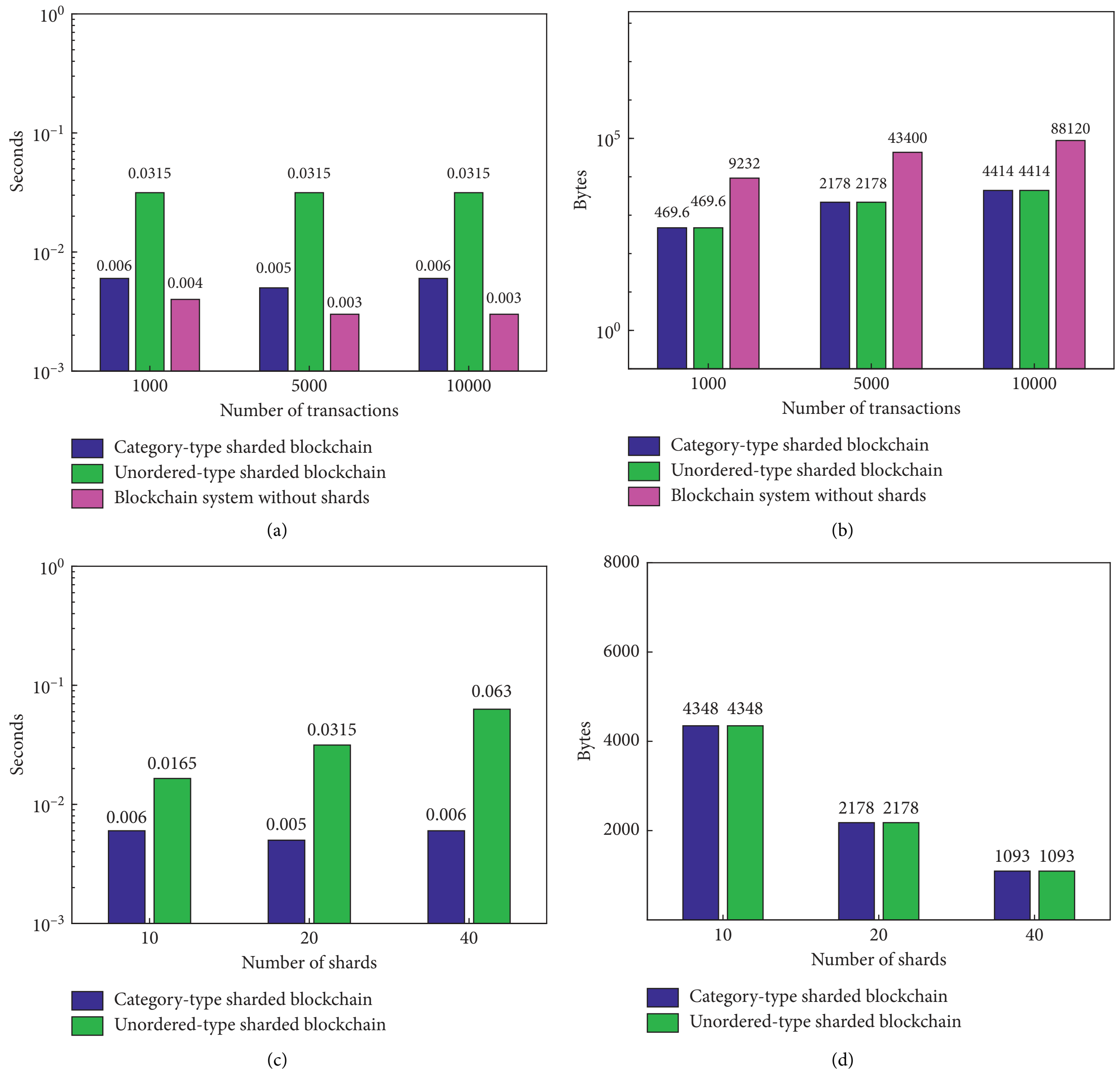


Figure 12: Time latency and memory cost of data retrieval. (a) Latency. (b) Peak memory cost. (c) Latency. (d) Peak memory cost.

(1) **Experimental settings**:
Quantitative: one shard; one client
Variable: data size
Experiments: Figures 13(a) and 13(b)
(2) **Experimental settings**:
Quantitative: one shard; 100 M data
Variable: number of clients
Experiments: Figures 13(c) and 13(d)

Box 6: The second group of experiments: data transmission.

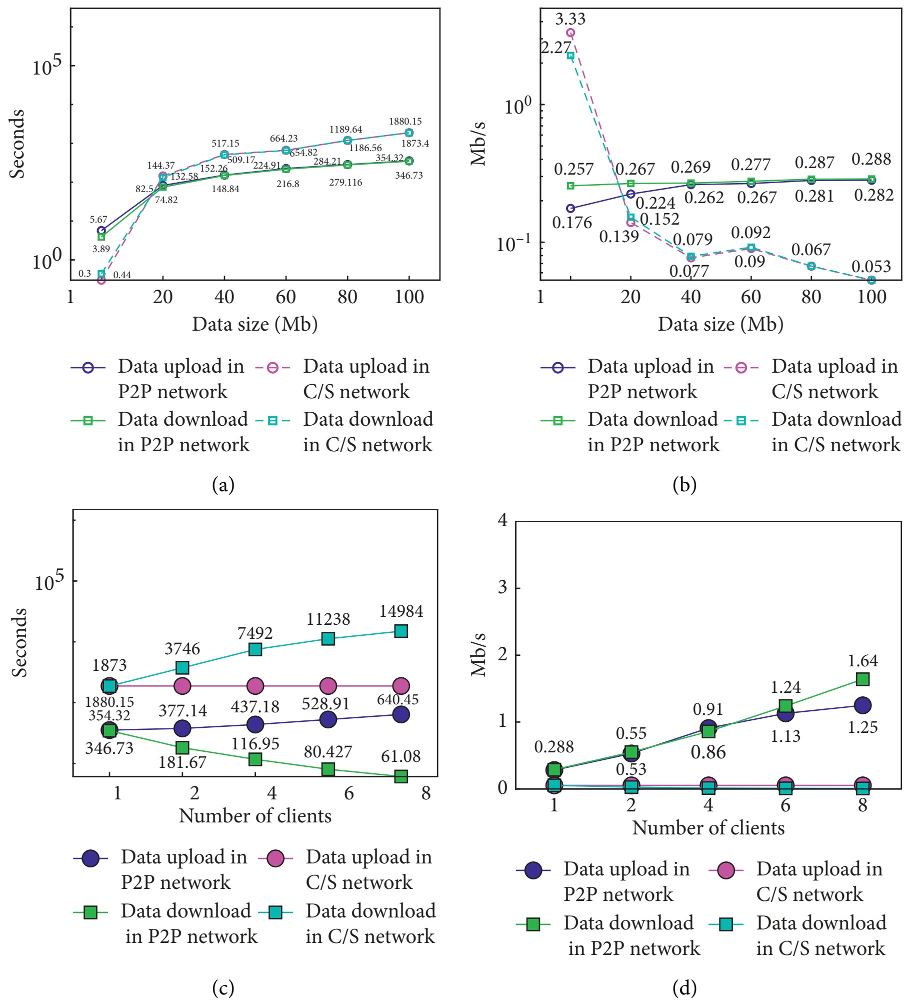


Figure 13: Network latency and load of data transmission. (a) Network latency. (b) Average network load. (c) Network latency. (d) Average network load.

the unordered sharded blockchain system. Both increase linearly with the number of transactions, far lower than the memory consumption of systems without shards. In Figure 12(d), when the total transaction volume is fixed, as the number of shards increases, the memory consumption from data retrieval in a client decreases since the average volume for each shard ledger is reduced.

Generally, the category-based sharded blockchain system has a steady data retrieval delay and minimum memory consumption and provides the best data retrieval performance.

Figures 13(a) and 13(b) show that in a P2P network, the network delay and the network load of data uploading and downloading increase with data size. However, compared to data download, data upload has a slightly higher network delay and slightly lower network load due to the setting of network bandwidth and computer hardware configuration.

In Figure 13(c), the more the clients in a P2P network, the worse the network delay for data upload caused by the multiple backups and synchronization of network data. The network delay in data download has a downward trend in the number of clients since in a P2P network, multiple hosts can improve the processing performance by dispersing the computing tasks of a blockchain system. The data upload and download latency in P2P networks is far lower than that of C/S architecture.

Figure 13(d) shows that the network load of data transmission increases due to data synchronization and communication between multiple hosts in a distributed architecture. In extreme cases, network congestion and network delay occur in P2P networks [72, 73]. Although multiple data backup increases memory consumption, the P2P network architecture in blockchain significantly improves data transmission speed.

7. Conclusions

Blockchain has developed considerably in recent years. At present, technological innovation has largely focused on improving the performance and security aspects of blockchain. DAG technology has been proven to be a feasible approach to significantly improve the performance of single-chain blockchain, in which consensus delay depends on transaction arrival rate as generally reaching the millisecond level and transaction throughput depends on the number of concurrent threads without upper limit in theory [20].

We developed a secure and efficient construction scheme for blockchain networks using a category-based sharding DAG blockchain. Compared with the scale-based sharded DAG blockchain, the category-based sharded DAG blockchain has higher cohesion and lower network coupling since similar data types and transaction information are organized and stored together. Compared with the account chain-based DAG blockchain [24], the category-based DAG blockchain has better interactivity since a transaction always involves multiple accounts, and similar data are easier to share.

To prevent double-spend attacks, we added a limited number of nodes using category-based sharding. Category nodes that are not enough to form a shard are temporarily stored in the root shard. When a category shard exceeds the standard quantity, our sharding protocol divides it to prevent network redundancy. To ensure data consistency in the category-based DAG blockchains, we used a consensus mechanism composed of "root shard" and "shard main chain": (1) the time series for all nodes of root shard are marked and stored orderly to achieve global sorting among shards and (2) the main chain of each subshard is used to achieve local sorting of one single shard. We also introduced a smart contract-driven crowdsourcing mechanism to prevent DDoS attacks, providing stronger financial controls and risk management capabilities.

Having low performance is a core issue of all sharded blockchains. Disordered sharding mechanisms may cause considerable transaction delay and network congestion due to frequent cross-shard communication and data verification. Our category-based shard protocol and redundant node generation strategy ensure that users can quickly find the latest states in the whole network of target data by loading several specific shards, which substantially improves transaction processing performance.

Using simulation tests and analyses, we were able to verify the feasibility and superiority of our design in terms of algorithm performance and system availability.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The research leading to these results received funding from the National Natural Science Foundation of China under grant no. 61772352 and the Science and Technology Planning Project of Sichuan Province under grant nos. 2019YFG0400, 2018GZDZX0031, 2018GZDZX0004, 2017GZDZX0003, 2018JY0182, and 19ZDYF1286.

References

[1] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proceedings of the International Conference Financial Cryptography Data Security*, pp. 136–149, Springer, Tenerife, Spain, January 2010.

[2] S. Nakamoto, *Bitcoin: A Peer-To-Peer Electronic Cash System*, http://bitcoin.org/bitcoin.pdf, 2008.

[3] V. Jacynycz, A. Calvo, S. Hassan et al., "Betfunding: a distributed bounty-based crowdfunding platform over Ethereum," in *Proceedings of the 13th International Conference Distributed Computing and Artificial Intelligence*, pp. 403–411, Springer, Sevilla, Spain, June 2016.

[4] M. A. Ferrag, M. Derdour, M. Mukherjee et al., "Blockchain technologies for the internet of things: research issues and challenges," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2188–2204, 2018.

[5] S. Aggarwal, R. Chaudhary, G. S. Aujla, N. Kumar, K.-K. R. Choo, and A. Y. Zomaya, "Blockchain for smart communities: applications, challenges and opportunities," *Journal of Network and Computer Applications*, vol. 144, pp. 13–48, 2019.

[6] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, vol. 107, pp. 841–853, 2020.

[7] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: using blockchain to protect personal data," in *Proceedings of the 2015 IEEE Security and Privacy Workshops*, May 2015.

[8] D. Shrier, W. Wu, and A. Pentland, "Blockchain & infrastructure (identity, data security)," *Massachusetts Institute of Technology-Connection Science*, vol. 1, no. 3, pp. 1–19, 2016.

[9] V. B. Khedekar, S. S. Hiremath, P. M. Sonawane, and D. S. Rajput, "Protection to personal data using decentralizing privacy of blockchain," *Transforming Businesses with Bitcoin Mining and Blockchain Applications*, IGI Global, Hershey, PA, USA, 2020.

[10] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K.-K. R. Choo, "Blockchain: a panacea for healthcare cloud-based data security and privacy?" *IEEE Cloud Computing*, vol. 5, no. 1, pp. 31–37, 2018.

[11] S. Lokhande, S. Mukadam, M. Chikane, and M. Bhonsle, “Enhanced data sharing with blockchain in healthcare,” in *Proceedings of the 2nd International Conference on Communications and Cyber Physical Engineering*, Pune, India, January 2019.

[12] S. Huh, S. Cho, and S. Kim, “Managing IoT devices using blockchain platform,” in *Proceedings of the 2017 19th International Conference on Advanced Communication Technology (ICACT)*, February 2017.

[13] G. Yu, X. Zha, X. Wang et al., “Enabling attribute revocation for fine-grained access control in blockchain-IoT systems,” *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1213–1230, 2020.

[14] A.-S. Kleinaki, P. Mytis-Gkometh, G. Drosatos, P. S. Efraimidis, and E. Kaldoudi, “A blockchain-based notarization service for biomedical knowledge retrieval,” *Computational and Structural Biotechnology Journal*, vol. 16, pp. 288–297, 2018.

[15] M. Takemiya and B. Vanieiev, “Sora identity: secure, digital identity on the blockchain,” in *Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, July 2018.

[16] Y. Hao, C. Piao, Y. Zhao, and X. Jiang, “Privacy preserving government data sharing based on hyperledger blockchain,” in *Proceedings of the International Conference on E-Business Engineering*, pp. 373–388, Springer, Shanghai, China, October 2019.

[17] ALIPAY, *World Record!! We’ve Processed 256,000 Payment Transactions Per Second (tps)*, ALIPAY, Shanghai, China, 2017, https://twitter.com/alipay/status/929123909970153472.

[18] S. Popov, *The Tangle*, Whitepaper, 2018, http://tanglereport.com/wp-content/uploads/2018/01/IOTA_Whitepaper.pdf.

[19] H. Pervez, M. Muneeb, M. U. Irfan, and I. U. Haq, “A comparative analysis of DAG-based blockchain architectures,” in *Proceedings of the 2018 12th International Conference on Open Source Systems and Technologies (ICOSST)*, pp. 27–34, Lahore, Pakistan, December 2018.

[20] B. Cao, Y. Li, L. Zhang et al., “When internet of things meets blockchain: challenges in distributed consensus,” *IEEE Network*, vol. 33, no. 6, pp. 133–139, 2019.

[21] C. Bai, “State-of-the-art and future trends of blockchain based on dag structure,” in *Proceedings of the International Workshop on Structured Object-Oriented Formal Language and Method*, pp. 183–196, Springer, Shenzhen, China, November 2018.

[22] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, “A secure shard protocol for open blockchains,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 17–30, Vienna, Austria, October 2016.

[23] A. Churyumov, “Byteball: a decentralized system for storage and transfer of value,” 2016, https://byteball.org/Byteball.pdf.

[24] C. LeMahieu, *Nano: A Feeless Distributed Cryptocurrency Network*, Nano Foundation, London, UK, 2018, https://nano.org/en/whitepaper.

[25] K. A. Hua and C. Lee, “An adaptive data placement scheme for parallel database computer systems,” in *Proceedings of the VLDB*, pp. 493–506, Brisbane, Australia, August 1990.

[26] Zilliqa is the World’s First Public Blockchain Built Entirely on a Sharded Architecture, 2017, https://www.zilliqa.com/platform.

[27] QuarkChain TPS Competition Final Leaderboard Announced with 310,000 TPS Achievement, 2020, https://community.quarkchain.io/t/quarkchain-tps-competition-final-leaderboard-announced-with-310-000-tps-achievement/106.

[28] Elrond: 3750+ TPS in a Single Shard, 30x Improvement in Throughput Compared to our Prototype, 2019, https://twitter.com/elrondnetwork/status/1094962922210627585.

[29] J. Wang and H. Wang, “Monoxide: scale out blockchains with asynchronous consensus zones,” in *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pp. 95–112, Boston, MA, USA, February 2019.

[30] Investment Analysis and Research Report of Near Protocol, 2019, https://www.8btc.com/media/515887.

[31] Z. Team, “The zilliqa technical whitepaper,” 2017, https://docs.zilliqa.com/whitepaper.pdf.

[32] The QuarkChain Whitepaper, 2018, https://www.quarkchain.io/wp-content/uploads/2018/11/QUARK-CHAIN-Public-Version-CN-0.3.4.pdf.

[33] Elrond, *A Highly Scalable Public Blockchain via Adaptive State Sharding and Secure Proof of Stake*, Whitepaper, 2019, https://www.chainwhy.com/upload/default/20190705/75d146eec7c85680f34461a0fe8a621b.pdf.

[34] A. Skidanov and I. Polosukhin, “Nightshade: near protocol sharding design,” 2019, https://nearprotocol.com/downloads/Nightshade.pdf.

[35] Coins max TPS, 2020, https://bitcointalk.org/index.php?topic=3026750.0.

[36] J. R. Douceur, “The sybil attack,” in *Proceedings of the International Workshop on Peer-To-Peer Systems*, pp. 251–260, Springer, Cambridge, MA, USA, March 2002.

[37] V. Katkar and S. Bhirud, “Novel DoS/DDoS attack detection and signature generation,” *International Journal of Computer Applications*, vol. 47, no. 10, pp. 18–24, 2012.

[38] Hyperledger Project, 2015, https://www.hyperledger.org.

[39] W. Zhang and Y. Ge, “Improvement of DPoS consensus based on block chain,” in *Proceedings of the 2019 4th International Conference on Intelligent Information Processing*, pp. 352–355, Kuala Lumpur, Malaysia, December 2019.

[40] R. Morais, P. A. Crocker, and S. M. D. Sousa, “A tool for implementing privacy in nano,” in *Proceedings of the 2020 IEEE International Conference on Decentralized Applications and Infrastructures*, IEEE, Oxford, UK, August 2020.

[41] G. Wood, “Ethereum: a secure decentralised generalised transaction ledger,” *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[42] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “An overview of blockchain technology: architecture, consensus, and future trends,” in *Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 557–564, Boston, MA, USA, December 2017.

[43] A. M. Antonopoulos and G. Wood, *Mastering Ethereum: Building Smart Contracts and Dapps: Chapter 4*, O’reilly Media, Newton, MA, USA, 2018.

[44] J. Benet, “Ipfs-content addressed, versioned, P2p file system,” 2014, http://arxiv.org/abs/1407.3561.

[45] B. A. Jantkal and S. L. Deshpande, “Hybridization of B-Tree and HashMap for optimized search engine indexing,” in *Proceedings of the 2017 International Conference on Smart Technologies for Smart Nation (SmartTechCon)*, pp. 401–404, Bangalore, India, August 2017.

[46] D. Vujičić, D. Jagodić, and S. Ranđić, “Blockchain technology, bitcoin, and Ethereum: a brief overview,” in *Proceedings of the 2018 17th International Symposium Infoteh-Jahorina (Infoteh)*, pp. 1–6, IEEE, East Sarajevo, Bosnia-Herzegovina, March 2018.

[47] Merkle Patricia Trie Specification (also Merkle Patricia Tree), 2020, https://github.comlethereum/wiki/wiki/Patricia-Tree.

[48] Q. Qu, I. Nurgaliev, M. Muzammal, C. S. Jensen, and J. Fan, "On spatio-temporal blockchain query processing," *Future Generation Computer Systems*, vol. 98, pp. 208–218, 2019, https://www.r3.com/reports/chain-interoperability/.

[49] V. Buterin, *Chain Interoperability*, R3 Research Paper, 2016.

[50] Cosmos Network, 2020, https://cosmos.network.

[51] V. Buterin, *Ethereum: Platform Review: Opportunities and Challenges for Private and Consortium Blockchains*, 2020, http://r3cev.com.

[52] W. Martino, *Kadena—the First Scalable, High Performance Private Blockchain*, Whitepaper, 2016, http://kadena.io/docs/Kadena-ConsensusWhitePaper-Aug2016.pdf.

[53] C. Ye et al., "Atomicity analysis of service composition across organizations," *IEEE Transactions on Software Engineering*, vol. 35, no. 1, pp. 2–28, 2008.

[54] V. Zakhary, D. Agrawal, and A. El Abbadi, "Atomic commitment across blockchains," 2019, http://arxiv.org/abs/1905.02847.

[55] D. Zhao and T. Li, "Distributed cross-blockchain transactions," 2020, http://arxiv.org/abs/2002.11771.

[56] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.

[57] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[58] S. Van Dongen, "Performance criteria for graph clustering and Markov cluster experiments," Technical Report INS-R0012, National Research Institute for Mathematics and Computer Science, Amsterdam, Netherlands, 2000.

[59] M. Rosvall, D. Axelsson, and C. T. Bergstrom, "The map equation," *The European Physical Journal Special Topics*, vol. 178, no. 1, pp. 13–23, 2009.

[60] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[61] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, Article ID P10008, 2008.

[62] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[63] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger, "Scan: a structural clustering algorithm for networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 824–833, San Jose, CA, USA, August 2007.

[64] T. Chakraborty, S. Patranabis, P. Goyal, and A. Mukherjee, "On the formation of circles in co-authorship net-works," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 109–118, Sydney, Australia, August 2015.

[65] Y. Han and J. Tang, "Probabilistic community and role model for social networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 407–416, Sydney, Australia, August 2015.

[66] S. D. Lerner, "DagCoin: a cryptocurrency without blocks," 2015, https://bitslog.files.wordpress.com/2015/09/dagcoin-v41.pdf.

[67] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[68] C. Pérez-Solà, S. Delgado-Segura, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Double-spending prevention for bitcoin zero-confirmation transactions," *International Journal of Information Security*, vol. 18, no. 4, pp. 451–463, 2019.

[69] Thieman, "Python implementation of directed acyclic graph," 2017, https://github.com/thieman/py-dag.

[70] Graphviz: Graph Visualization Software, 2020, http://www.graphviz.org.

[71] T. T. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K. L. Tan, "Blockbench: a framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, Chicago, IL, USA, May 2017.

[72] E. Adar and B. A. Huberman, "Free riding on gnutella," 2000, https://www.hpl.hp.com/research/idl/papers/gnutella/gnutella.pdf.

[73] S. Rathore, B. Wook Kwon, and J. H. Park, "BlockSecIoTNet: blockchain-based decentralized security architecture for IoT network," *Journal of Network and Computer Applications*, vol. 143, pp. 167–177, 2019.