Received February 15, 2020, accepted March 2, 2020, date of publication March 5, 2020, date of current version March 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2978614

GCBlock: A Grouping and Coding Based Storage Scheme for Blockchain System

BIN QU, LI-E WANG, PENG LIU, ZHENKUI SHI, AND XIANXIAN LI
School of Computer Science and Information Technology, Guangxi Normal University, Guilin 541004, China

Corresponding author: Xianxian Li (lixx@gxnu.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 61662008, Grant 61672176, and Grant 61502111, in part by the Guangxi Bagui Scholar Teams for Innovation and Research Project, in part by the Guangxi Collaborative Innovation Center of Multi-Source Information Integration and Intelligent Processing, in part by the Guangxi Special Project of Science and Technology Base and Talents under Grant AD16380008, in part by the Guangxi Natural Science Foundation under Grant 2018JJA170082 and Grant 2016GXNSFAA380192, and in part by the Postgraduate Education Innovation Program of the Guangxi Zhuang Autonomous Region through Project under Grant Ycsw2019071.

ABSTRACT To achieve the tamper-proof, reliability and traceability of transactions in a trustless environment, the blockchain requires each peer node to store the whole global ledger. However, as transactions keep increasing over time, the storage cost of each node increases. In addition, many schemes have been proposed to boost rapid transactions which will even lead transactions to grow explosively. The problem of storage is becoming one challenge of blockchain since the storage overhead of each node increase rapidly. Reducing the storage overhead of each participant is very urgent and worthy. In this paper, we present GCBlock: a grouping overlay network storage scheme for the blockchain which can reduce the storage overhead of nodes and cut the whole storage cost of the blockchain greatly while keeping the underlying protocols. In our scheme, we try to group the nodes according to their physical fuzzy distance to reduce the overall delay when tracing. We set rules of autonomous check to deal with evil behavior within the group. To further enhance the stability of our scheme, we propose the *transcript fractional repetition* code which is newly constructed based on the *fractional repetition* code to encode data. Finally, we make a comprehensive evaluation of GCBlock and the results show that it is workable and reasonable.

INDEX TERMS Blockchain, distributed storage, overlay network, network coding.

I. INTRODUCTION

Since the birth to Bitcoin [1], blockchain technology has attracted great attention on both academic and industry for its trustworthy, tamper-proof, traceable, and decentralized. People began to explore the application mode of blockchain. As an example, Ethereum [2] even can run apps written in Turing-complete languages. Blockchain is a P2P distributed ledger technology, which maintains a global ledger through each node. However, to prevent malicious tampering with accounts, data redundancy has to be infinitely expanded. As a result, the storage cost of nodes is rising. Taking Bitcoin as an example, the size of a block data is about 1MB, and a new block is created every 10 min. Each individual node increases storage costs nearly 52GB per year.

In addition, fast transactions are a general trend to increase system throughput and many recent schemes have been proposed [3]–[5]. When the transaction throughput of a blockchain system increases to the level of VISA, the blockchain system needs to increase the storage cost of 214PB each year [6]. Therefore, with the development of the blockchain, the storage cost will become a problem that cannot be ignored.

Pruning data and sharding storage are popular candidates to reduce data redundancy in blockchains. Pruning data may lose the traceability of the blockchain. A blockchain system without traceability will lose many application scenarios, such as auditing, product traceability [7]–[9], and so on. Sharding storage is to disperse store data in different storage sharding to reduce redundancy. The sharding blockchain system modifies the protocol of the traditional blockchain. Therefore, different committees need to constantly interact to verify new transactions. It can reduce redundancy, but changes the underlying storage and verification modes. Modifying the original protocol of an existing blockchain system requires community consensus, which requires a lot of work

The associate editor coordinating the review of this manuscript and approving it for publication was Giacomo Verticale.

TABLE 1. The differences between GCBlock and blockchain protocols.

Name	Operating range	Function
Blockchain protocols	Blockchain system itself	Ensure the normal operation and system security of the blockchain sysytem.
GCBlock	In an overlay network using an existing blockchain system as the underlying network	Reduce storage of nodes in existing blockchain systems.

and risks ecological damage. Therefore, this paper tries to solve the storage redundancy problem without modifying the existing blockchain system.

Here, we propose GCBlock: a scalable and tolerant disperse storage scheme by leveraging grouping and a new created network coding architecture. The grouping idea is somewhat similar to the existing sharding system [4], but the application model and structure are different. First, grouping is performed on the overlay network layer and will not change the existing underlying architecture of the blockchain. Our approach is only to provide an option to reduce storage cost in existing blockchain systems. It differs from the sharding blockchain system which changes the protocols and architecture of the most existing system. Then, in terms of disperse data, the sharding blockchain system needs to store data instantly, while we only disperse the data after the transaction consensus is reached. Because the data after the consensus is not directly used to verify new transactions, it will not break the underlying running mechanism. For example, in Bitcoin, new transactions are verified by a copy of UTXO [10]. In addition, GCBlock is not a blockchain protocol, but a scheme to reduce storage for existing blockchain systems. The differences between GCBlock and blockchain protocols are shown in Table 1.

To reduce the communication delay between nodes within a group, we group based on fuzzy distance. To further enhance stability and reduce redundancy, we construct the *transcript fractional repetition* (TFR) code, which is a new code scheme using erasure code within a group. For the blockchain system is a complete P2P network and the join and departure of nodes is uncertain, traditional erasure code schemes cannot handle this instability very well. TFR code is based on *fractional repetition* (FR) code [11] to add layers, which will enhance the fault tolerance of the system. In addition, because the nodes in the blockchain network are not always benign, our distributed storage scheme may also suffer attacks from malicious nodes. To address these challenges, we introduce a way to independently detect malicious and selfish behaviors within the group. Malicious and selfish peers can be removed from the group by a node's autonomous check. Specifically, our contributions can be summarized as follows:

1) In this paper, we present a scheme to reduce the storage cost for the existing blockchain system by leveraging grouping overlay network and a new network code architecture.
2) To shorten the communication delay when tracing, the grouping criteria is based on fuzzy distance. The grouping is over overlay network which can keep the original protocols of blockchain systems.
3) We construct a new coding scheme - TFR code which aims to adapt the scalability of the number of nodes in the group. To further enhance system stability, we propose countermeasures to identify evil behaviors and nodes within the group.
4) We evaluate our scheme comprehensively. The results show that the communication overhead and system latency are reasonable.

The rest of the paper is organized as follows. Section II is related work. Section III gives the basic structure model, the threat model and the design goals. Section IV proposes our scheme, including network grouping and network coding. Section V analyzes of the security of GCBlock. Section VI evaluates the performance of GCBlock. Section VII concludes the paper.

II. RELATED WORK

Recently, some researchers have focused on pruning old data and designing new system architectures to reduce blockchain data redundancy. In the Bitcoin white paper [1], Nakamoto has proposed "restoring disk space" and "simplifying payment verification (SPV)". SPV nodes need to rely on other nodes to verify transactions, which reduces security. Restoring disk space is to prune old transaction data. Similarly, in sharding blockchain system, E. Kokoris-Kogias *et al.* [5] also chose the way of pruning old data to reduce data redundancy. However, with the pruning of transaction data, the blockchain also loses the properties of data traceability. R. K. Raman *et al.* [12] designed a new blockchain that uses Shamir secret sharing coding schemes to allow each node to store only a portion of the data to reduce storage costs. M. Zamani *et al.* [4] designed a new sharing blockchain protocol and proposed a sharding storage scheme that allows each committee to store a portion of the data to reduce data redundancy. Our scheme is different from the above. We do not change the original blockchain system, but only provide a solution to solve data redundancy for the existing blockchain system. We propose to add an overlay layer for the existing blockchain network, allowing the nodes to choose whether to join the grouping overlay network. Nodes joining a grouping rely on data encoding to reduce data redundancy.

Z. Xu *et al.* [13] proposed a scheme to reduce data redundancy in the private/permissioned blockchain by assigning blocks in a Consensus Unit. In their scheme, the Consensus Unit comprises a group of nodes which work together and just need to maintain at least an entire copy of Blockchain data. This may be impractical in public blockchain for nodes in public blockchain are independent of each other and the assignments may be unequal for both storage costs and query costs. Besides, they should estimate the communication

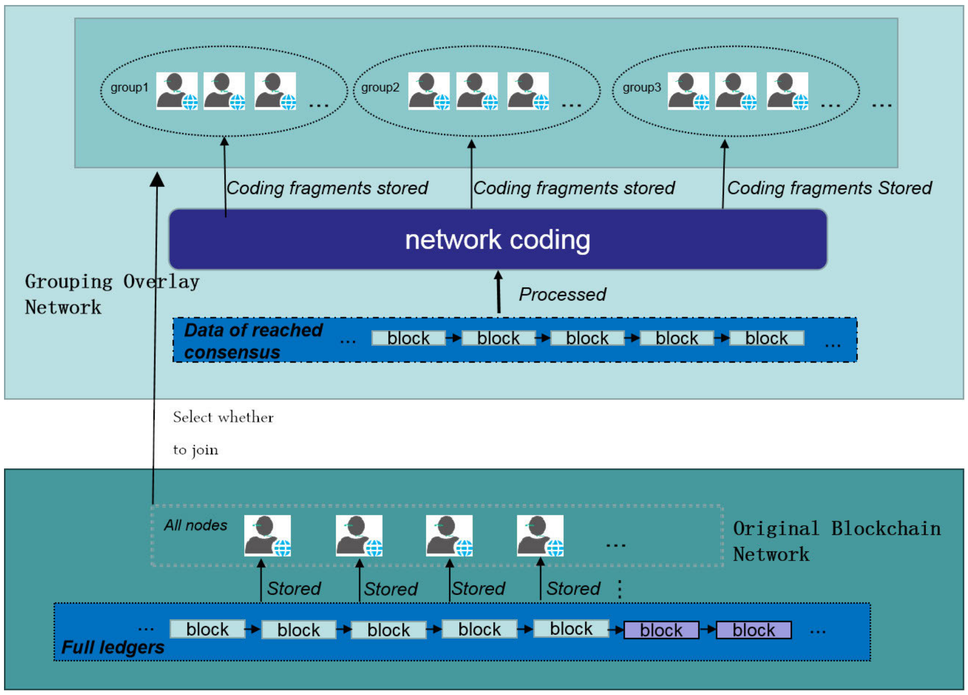


FIGURE 1. The basic structure model.

costs and the frequently access set periodically and adjust to reassign the blocks which may be unstable and can be affected by multiple factors. In this paper, we focus on the redundancy problem in the public blockchain rather than the private/permissioned blockchain. We use an erasure code scheme to reduce node redundancy instead of the replication scheme. It has been demonstrated [14] that erasure coding has greater advantages in terms of fault tolerance, lower bandwidth and lower storage.

III. PRELIMINARIES

In this section, we define the basic structure model, the threat model and on that basis present our design goals.

A. STRUCTURE MODEL

To show our scheme more clearly, we draw the original blockchain network model and grouping overlay network model as shown in Figure 1.

Original Blockchain Network: A common blockchain network, such as Ethereum, Bitcoin, etc.

Full ledgers: The global ledger data of the blockchain includes the data of reached consensus and the data of waiting for consensus. The full ledgers are stored synchronously in each node.

All nodes: All nodes that store the full ledgers (e.g., the full node of Bitcoin). All nodes can choose whether to join the grouping overlay network.

Grouping Overlay Network: A grouping network built on top of the original blockchain network.

Data of reached consensus: The data of reached consensus cannot be changed. (e.g., Bitcoin transactions that have been confirmed after 6 blocks are almost impossible to tamper with.) GCBlock only operates on the data after the consensus is reached.

Network coding: Erasure code is used to encode block data and store them in groups. Our coding scheme is described in Section IV.

Groupi: Groups formed by nodes that have joined the overlay network. Specific grouping schemes will be described in Section IV.

Next, we briefly describe how nodes reduce storage costs by joining the grouping overlay network. When a node joins the GCBlock, the node preferentially joins the closest group. After successfully joining the group, it accepts encoded fragments within a group and deletes data of reached consensus stored by itself. In this way, the node only stores a part of the encoded fragments, which reduces the storage costs. When data needs to be recovered, the node can download enough encoded fragments from peer nodes within the group to decode recovery.

B. THREAT MODEL

Since the GCBlock only provides a service to reduce data redundancy of the original blockchain, we do not consider the case where the blockchain system is compromised. In addition, we also do not consider irrational large-scale attacks.

In the threat model, we mainly consider the *selfish behavior* of nodes and the *malicious behavior* of nodes. The *selfish behavior* of nodes is mainly *free-riding*. *Free-riding* means that the node does not store any encoded fragments and gets encoded fragments from other nodes in the group when needed. The *malicious behavior* of a node is that the node provides spurious encoded fragments to peers within the group. Sometimes, *malicious behavior* is a disguise of *selfish behavior*. For example, selfish nodes provide spurious encoded fragments for others. We collectively refer to selfish behavior and malicious behavior as evil behavior.

C. DESIGN GOALS

The main purpose of GCBlock is to reduce the storage costs of blockchain systems. As we all know, the blockchain network is a P2P network. Therefore, we considered the characteristics of P2P networks to design GCBlock. Our design goals are:

High tolerance: In a P2P network, nodes may quit and join at any time. Our coding scheme must be highly flexible to accommodate the constant quitting and joining of nodes while ensuring extremely low data recovery failure rates.

Security: For the evil behaviors proposed by the threat model, we need to formulate strategies to remove the evil nodes from the group.

Efficiency: Nodes within a group should be able to transfer data with low latency.

IV. PROPOSED SCHEME

A. OVERLAY GROUPING NETWORK

An overlay network is a virtual network built on top of an existing network through logical links of nodes, designed to extend services that are not available on existing networks [15]. There is no need to changes the underlying network, overlay network looks like a normal user-level application. Constructing GCBlock in the overlay layer can maintain the running mechanism and architecture of the original blockchain system. There have been many researches on the application of overlay network in P2P network, which can be divided into structured overlay network [16], [17] and unstructured overlay network [18], [19]. A structured overlay network maintains a specific structure that enables reliable routing between a limited number of nodes. Compared with the unstructured overlay network, the nodes of the structured overlay network can choose the neighbors that meet certain constraints in a set of nodes.

To group nodes based on delay is an effective approach for organizing and managing nodes that join GCBlock. The delay between nodes of the same group is less than that between nodes of different groups, to reduce network traffic and improve routing efficiency. Usually, the delay between nodes is proportional to their physical distance. Therefore, we can construct a structured overlay network based on grouping according to the underlying physical distance. In addition, the physical scope within the grouping should be reduced and the upper bound of this range should be specified to meet the requirements between nodes.

1) BEGINNING GROUPING

The algorithm to group nodes by geographical location on overlay network is usually called landmark clustering. In the grouping algorithm, we need to select the landmark node and then optimize the topology based on the landmark node. It is an NP-hard problem to find landmark node that can minimize the total communication delay in the group. Therefore, the algorithm for beginning grouping should be able to achieve our requirements approximately and get a grouping topology. In the beginning state, GCBlock needs a bootstrap node. The bootstrap node is only for boot system startup and will stop working after system startup.

The bootstrap node asks joining nodes for routing messages, and then obtains a similar distance among the nodes according to count the IP (The IP in this paper is the IP address.) hop in route messages. Therefore, we can easily get a similar distance matrix. The network nodes grouping also becomes points grouping in the matrix. This idea of network grouping is called fuzzy clustering [20], [21] and has been used in many fields [22]–[24].

For the uncertainty of the number of groups, the hard clustering algorithm that directly specifies the number of groups is not suitable. Thus we introduce the ISODATA algorithm [25], [26] and make adaptive modifications for network grouping in Algorithm 1. To implement the algorithm, we need to set the following parameters:

(1) s–The number of centers that were selected randomly;
(2) D_{min}– The minimum distance between groups;
(3) D_{center}–The distance between groups;
(4) σ_{max}–The maximum variance within the group;
(5) n–The minimum number of nodes in a group;
(6) σ_{λ}– The variance of nodes within a group;
(7) $Sum[G_i]$–The number of nodes in a group;
(8) v–The number of iterations;

In order not to affect the nature of the algorithm, the delay between nodes is represented by the Euclidean distance. In the first step, the algorithm randomly selects s center points as group centers. The second step is to compare the distance from each point to the centers and add the node to the nearest group. The third step is to calculate the number of points in each group and remove the group that the number of points less than n. The fourth step is to recalculate the coordinates of each group center and σ_{λ} of each group. If $\sigma_{\lambda} > \sigma_{max}$ and $Sum[G_i] > 2n$, then the group will be split two group according to the standard deviation as an offset value. The fifth step is to calculate the D_{center}. If $D_{center} < D_{min}$, then the two groups will be merged and the center of the new group will be calculated. The sixth step is to calculate the ungrouped nodes' set U and the number of running Q. When $Q > v$, the calculation stops and the grouping result is outputted, otherwise it returns to the second step.

After outputting the grouping result, the bootstrap node will send the number of a corresponding group and the IP list of group members to nodes and complete the beginning grouping phase. For the randomness of the algorithm, the optimal result cannot be guaranteed. Some nodes that too scattered may not join a group during the beginning grouping phase. Therefore, these nodes need to wait for new nodes to join and form new groups.

Algorithm 1 Beginning Grouping

```
Input: Nodes set $U$ and distance matrix $M_{|U|*|U|}$
Output: The result of initial grouping $G = \{G_1 G_2 \ldots G_i\}$
    Step 1:
 1: Randomly selected $s$ nodes from $U$ as the centers of initial groups: $C = \{c_1 c_2 \ldots c_s\}$;
    Step 2:
 2: for each $u_i \in U$ do
 3:     computing $d(u_i, c_j), j = 1, 2 \ldots s$;    ◁ $u_i$ represents a node.
 4:     Add $u_i$ to the group-$G_a$ and satisfies d $(u_i, c_a) = \min \{d(u_i, c_j)|j = 1, 2, ..s\}$;
 5: end for
    Step 3:
 6: Recomputing the number of each group Sum[$G_i$], i $= 1, 2 \ldots$ s;
 7: while $Sum[G_i] < n$ do
 8:     Get rid of the $G_i$;
 9:     $s \leftarrow s - 1$;
10: end while
    Step 4:
11: Updating the grouping centers    ◁ Take the mean of each dimension in the group
12: Computing the variance within the group $\sigma_\lambda$, $\lambda = 1, 2 \ldots s$.
13: if $\sigma_\lambda > \sigma_{max}$ and $sum[G_i] > 2n$ then
14:     Split into two different groups
        $New_1$ center $\leftarrow$ center $+ sqrt(\sigma_{max})$
        $New_2$ center $\leftarrow$ center $- sqrt(\sigma_{max})$
15:     $s \leftarrow s + 1$
16: end if
    Step 5:
17: Computing the distance between centers-$D_{center}$
18: if $D_{center} < D_{min}$ then
19:     Merge $G_i$ and $G_j$
        Computing the new center-$G_{ij}$
        $s \leftarrow s - 1$
20: end if
    Step 6:
21: $U \leftarrow U - \sum_{i}^{s} G_i$
22: $Q \leftarrow Q + 1$    ◁ Q is the iteration count.
23: if $Q > v$ then
24:     goto Step 2
25: else
26:     output the $G = \{G_1 G_2 \ldots G_i\}$
27: end if
```

Remark: In cases where regional faults (network fault and power fault) need to be considered, our scheme can be easily extended to a simple grouping way in which nodes are randomly divided into groups. However, this will lose the efficiency of transferring data among nodes. Actually, our scheme can provide alternative node-dividing methods to be compatible with the different requirements.

2) DUTY NODE

For out of group communication and intragroup tasks, each group needs to select a duty node at every T interval. The duty node is selected by random voting within the group.

a: SELECTION RULES

Scope: Each node randomly selects a node from the 1/3 nodes that joined the group for the longest time. *Method:* Each selection result will be broadcast within the group, the node with the highest number of votes becomes the duty node. Therefore, the longer the joining time, the easier it is to be selected as the duty node. We also believe that the stability and reliability of the node with a long joining time are higher. *Balance:* When a node is selected as the duty node, its join time will be reset. This ensures a more balanced chance of becoming a duty node.

When the duty node does not send the encoded fragments within M time, the group members will remove the IP of duty node from the IP list of group members and reselect a new duty node.

b: RESPONSIBILITY

During the tenure, the duty node is responsible for encoding the data and sending it to the peers within the group according to encode rules. The duty node is also responsible for checking malicious and selfish behavior and alerting peers within the group. Additionally, the duty node is also responsible for the tasks of contacting new nodes and checking nodes exit.

c: DEAL WITH EVIL BEHAVIOR

When the node finds malicious or selfish behaviors, it can remove the evil nodes from the IP list of group members and report these behaviors to the duty node. The duty node will verify the report. If an evil behavior is verified, it will broadcast the alerting signal to the peers within the group. The peers self-verify evil behaviors by the alerting signal, and remove the node from the IP list of the group members when the evil nodes are identified. All IP of evil nodes will be added to a blacklist of the group by the peers, and the IP in the blacklist will be forbidden to join the group.

d: IDENTIFY EVIL NODE

After encoding, the duty node hashes the encoded fragments, forms a Merkle tree structure and broadcasts it to all nodes in the group. When decoding is required, the node can verify the obtained encoded fragments according to the Merkle tree, and can quickly identify the source of the pollution. As shown in Figure 2, a schematic diagram for the generation of a Merkle hash tree with eight encoded fragments.

Remark: The duty node does not have the authority to remove other nodes from the group. Removing an evil node from the IP list of the group members are nodes' self-behavior. When most nodes within a group remove the evil node from the IP list of the group members, the evil node

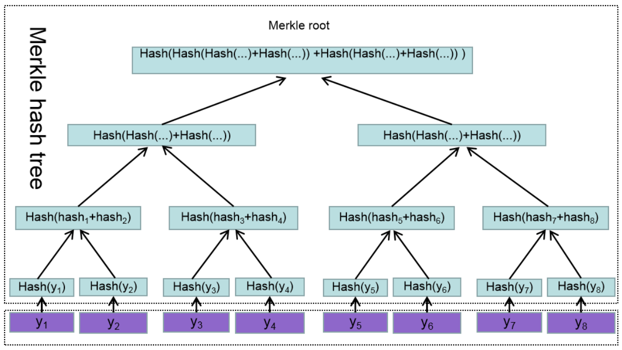

Merkle hash tree
Merkle root
Hash(Hash(Hash(...)+Hash(...)) +Hash(Hash(...)+Hash(...)))
Hash(Hash(...)+Hash(...))
Hash(Hash(...)+Hash(...))
Hash(hash1+hash2)
Hash(hash3+hash4)
Hash(hash5+hash6)
Hash(hash7+hash8)
Hash(y1) Hash(y2) Hash(y3) Hash(y4) Hash(y5) Hash(y6) Hash(y7) Hash(y8)
y1 y2 y3 y4 y5 y6 y7 y8

FIGURE 2. An eight-leaf Merkle hash tree. The leaves of a tree are hashes of encoded fragments. The Merkle hash tree is formed by recursively hashing the hashes of the leaves.

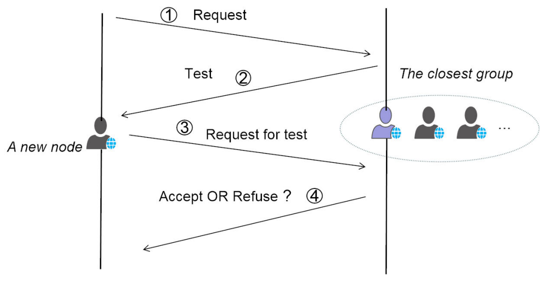


FIGURE 3. The request process for a new node to join a group.

cannot get data from these nodes within the group. In this way, the evil node is removed from the group.

3) JOINING AND EXITING

The joining and exiting of nodes in a P2P network are unpredictable. For joining and exiting nodes, we give the following scheme:

New node joining: When a new node joins the network, it sends a notification message to the entire network. The duty nodes reply with the confirmation message to the new node, indicating that the new node has been known to join the overlay network. The new node estimates the distance according to the delay of the confirmation message, and then requests to join the group with the closest distance. The process of request to join a group is shown in Figure 3.

Describe the process:

(1) The new node sends a request join message to the closest duty node.
(2) After receiving the request message, the duty node sends a test message to the new node.
(3) When the new node receiving the test message, it immediately sends the request test message.
(4) After receives the request test message, the duty node estimates the distance of the new node according to the delay. The duty node sends an accept message to the new node when the distance of the new node does not exceed the required maximum distance within the group, otherwise sending a refuse message. The distance within a group is estimated by communication delay.

When the duty node sends the acceptance message, it also broadcasts the information of the new node to the nodes within the group. Each node within the group will add the information of the new node. If refused, the new node needs to wait for other nodes to join and form a new group.

Node Exiting: Through the heartbeat mechanism, the duty node can detect the node exit and broadcast a message within the group. The exit node will be removed from the group and the IP of the node will be added to a limit list. The limit list is designed to prevent the intermittent offline behavior of the nodes. If the node exits, it will be restricted to rejoin for a long time.

B. NETWORK CODING

The joining and exiting of the nodes is uncertain in the blockchain network. Therefore, our coding scheme has to meet the following two requirements:

(1) It is able to adapt to the constant changes in the number of nodes within the group.
(2) It has the ability to quickly repair the fault nodes and higher fault tolerance.

In faulty node repair, most erasure code systems use coding repair. Surviving nodes and new nodes need to calculate a linear combination of stored symbols for regeneration, which undoubtedly adds additional computational overhead. To reduce the delay of repairing failed nodes, our scheme adopts the idea of the FR code. FR code is a non-coding repair based on a table, which reduces the complexity of additional coding repair than the *regenerating code* [27].

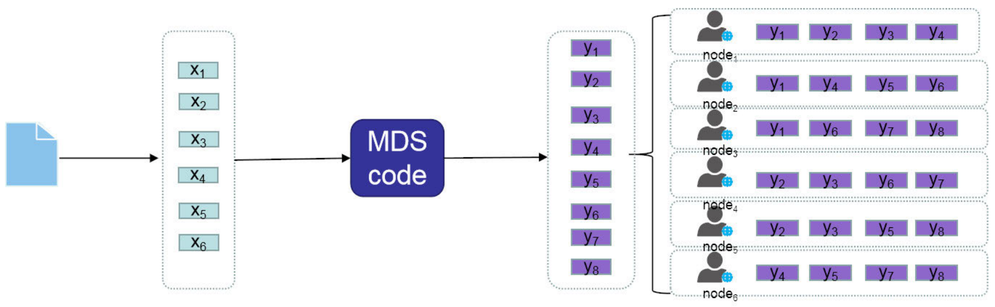
However, once the parameters of FR code are set, the number of nodes are fixed. This cannot adapt to the constant changes in the number of nodes in the blockchain network. To meet the change in the number of nodes, we propose the *transcript fractional repetition* code which is newly constructed based on the FR code.

1) REVIEW FRACTIONAL REPETITION CODES

The FR code is constructed based on the *Maximum Distance Separable* (MDS) [28] code. The data is encoded into θ fragments by MDS, and then each node in the system stores d encoded fragments. Each encoded fragment is placed in ρ different nodes. By an index table, the FR code can achieve the purpose of exact repair.

Here we need to describe the MDS code: The MDS code with the parameter (θ, k) means that the M size file/data is divided into k pieces, and then the k pieces are encoded into θ fragments (of the same size) by MDS code. Where $\theta > k$, the number of the parity-check fragments is $r = \theta - k$. Any k encoded fragments in θ can be used for decoding to reconstruct data.

FR code can be described as follows: The MDS encoded fragments $\{1, \ldots \theta\}$ are repeatedly placed in n nodes, each node stores d encoded fragments and each encoded fragment must belong to ρ nodes. Figure 4 shows a schematic diagram of FR code. The parameters $(n, d, \rho) = (6, 4, 3)$, the data is



x_1
x_2
x_3
x_4
x_5
x_6
MDS code
y_1
y_2
y_3
y_4
y_5
y_6
y_7
y_8
$node_1$ y_1 y_2 y_3 y_4
$node_2$ y_1 y_4 y_5 y_6
$node_3$ y_1 y_6 y_7 y_8
$node_4$ y_2 y_3 y_6 y_7
$node_5$ y_2 y_3 y_5 y_8
$node_6$ y_4 y_5 y_7 y_8

FIGURE 4. An example of a simple FR code. First, a data block is divided into six equal pieces, and then encoded into eight equal pieces by a MDS code (8,6). Finally, these encoded fragments are repeatedly placed to form an FR code with a repetition degree $\rho = 3$.

TABLE 2. A level index table of the TFR code.

Fragment number	Leve 0	Level 1	...
(1,2,3,4)	ID-IP	ID-IP	...
(1,4,5,6)	ID-IP	ID-IP	...
...	...	...	...

encoded by (8, 6) MDS code (parity-check fragments $r = 2$), and then according to the rules of the FR code to store the encoded fragments in six nodes. The parameters in the FR code satisfy the following formula [11]:

$$\theta\rho = nd \tag{1}$$

For detailed rules of construct FR codes, the reader is urged to read [11].

2) TRANSCRIPT FRACTIONAL REPETITION CODES

To meet the fluctuation of the number of nodes, we constructed the TFR code. The TFR code extends the flexibility of the FR code by adding replication levels to cope with the fluctuation of the number of nodes within the group.

Construct the TFR code: First construct an initial FR code as level 0 and then add levels based on the level 0. All levels form a level index table. The level index table is dynamic and changes as nodes within the group change. Each complete level is a copy of the initial FR code. In Figure 5, an example of the TFR code, the initial FR code with n = 6 is shown$\{node_1, node_2, node_3, node_4, node_5, node_6\}$, which are the nodes of the level 0. The nodes of level 1 are $\{node'_1, node'_2, node'_3, node'_4, node'_5, node'_6\}$. As the number of nodes within a group increases, the number of levels can be expanded.

The encoding process: The duty node encodes the block data using the MDS code, and performs encoding fragments distribution according to the level index table of the TFR code. For example, send a set of encoded fragments (y_1, y_2, y_3, y_4) to a set of nodes $(node_1, node'_1, node''_1)$. As shown in Table 2, a level index table of the TFR code. ID-IP is the unique identifier for each node.

Verification encode fragments: After receiving the encoded fragments, the nodes perform to decode verification. After determining the trueness of the data encode, the nodes delete the original block data to save the encoded fragments.

Repair rules: Priority to repair the node of the upper level. We specify that the level 0 has the highest priority, so the integrity of level 0 needs to be maintained in priority. Repair the fault node of a higher level, first find a copy of the fault node from the lowest level and move it to the higher level of the level index table. If the lowest level does not have a copy of this failed node, it executed up one level. In the worst case, the fault node has no copy and needs to wait for the new node to join to repair it.

Restore data: A node should preferentially obtain the encoded fragments from the nodes of the same level for decoding. If the encode fragments of the same level are not enough, then to access the upper level to get the encode fragments.

Dynamic level index tables:

(1) Remove node: When a node is removed by the IP list of the group members, the node will be removed from the level index table. After the node is removed, repair it according to the repair rules.

(2) Add node: When a new node joins a group, the level index table needs to be updated. First, the new node gets the level index table of the group from the duty node. Then, according to the level index table and repair rules, the new node repairs the fault node. If all levels are complete, the new node will form a new level. After the new node updates the level index table, it will broadcast the content of update within the group. When the update is received, the members of the group will update the level index table.

V. SECURITY ANALYSIS

We analyze GCBlock security in terms of grouping security and coding availability.

A. GROUPING SECURITY

GCBlock serves the public blockchain, so the data is public and there are no access control security issues. GCBlock is essentially a mutually beneficial storage solution. GCBlock is essentially a mutually beneficial storage solution. Due to

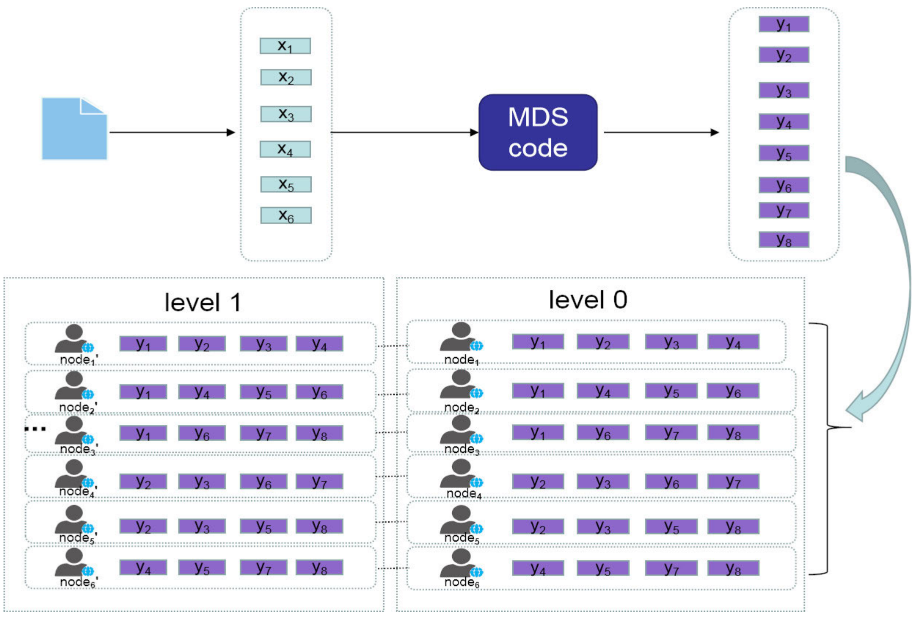


FIGURE 5. A example of TFR code. TFR code is formed by adding levels of replication to the basis of FR code.

TABLE 3. The profit of games between nodes.

		P_a	
		evil	not evil
P_b	remove	$(k-d)L/k, -(k-d)L/k$	$-(k-d)L/k, -(k-d)L/k$
	not remove	$-(k-d)L/k, L$	$(k-d)L/k, (k-d)L/k$

the GCBlock is built on the overlay, we do not consider the case where the underlying layer is being attacked on a large scale. Implementing a large-scale attack against GCBlock does not bring any direct benefits, and the attack cost are comparable to attack the original blockchain (for example, 51% attack).

We will discuss the evil behavior presented in the threat model. The nodes that join the group are only interested in reducing storage. According to the game, we represent the reduced storage overhead as profit. So, the stable profit of per node within the group are $(k-d)L/k$. L is the block data originally stored by each node. An evil node will not store any data, so its profit becomes L. If the evil node are not removed from the IP list of group members, the profit of other nodes in the group will be lost. The biggest loss is $(k-d)L/k$. If the evil node are removed, the other nodes will retain the stable profit. If a node removes an honest node, it will reduce the chances of getting the fragments. This causes a loss to itself and to the removed node, and the biggest loss is $(k-d)L/k$. Table 3 shows the profit state of the game. In Table 3, four basic game types as follows. P_a represents the evil node and P_b represents the honest node.

(1) (remove, evil): When P_b removes the evil node from the IP list of the group members, P_b makes a choice for the stability of the group and will continue to maintain the profit $(k-d)L/k$. For P_a, will lose the profit $(k-d)L/k$ when it is removed.
(2) (remove, not evil): When P_b removes a not evil node from the IP list of the group members, it reduces the chance that gets the encoded fragments. This biggest loss may be $(k-d)L/k$. This is an act that does harm to others but no good to itself.
(3) (not remove, evil): If P_b does not remove the evil node from the IP list of the group members, this will affect the stability of the group and cause a loss to itself. The biggest loss is $(k-d)L/k$. And then the profit of the evil node is L.
(4) (not remove, not evil): When there are no evil nodes, P_b does not remove any nodes from the IP list of the group members. In this state, all nodes maintain the profit $(k-d)L/k$.

The wisest choice of nodes is not to do evil, so as to ensure the balance of interests of all parties. And the honest node does not remove the non-evil node from the IP list of the group members, so that both parties do not lose. When a node is evil, other nodes will remove it from the IP list of the group members and add it to the blacklist to ensure maximum benefit. Therefore, this will reach a state of Nash equilibrium.

B. CODING AVAILABILITY

In [11], we can know that the FR code can maintain the uncoded repair feature when $\rho-1$ nodes are offline. When an encoded fragment in a group is losing, we call the group lose the uncoded repair feature. Since the exit behavior of

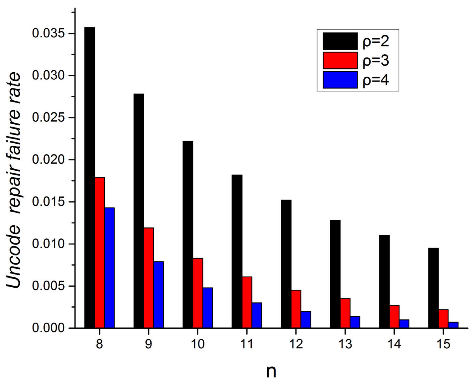


FIGURE 6. $\rho = 2, 3, 4$ and $n = 8, 9, 10, 11, 12, 13, 14, 15$, the uncoded repair failure rate R.

TABLE 4. R^m with different n and ρ.

n	ρ	R	R^2	R^3
8	2	0.0357	0.0012	4.55E-05
9	3	0.0119	0.00014	1.685E-06
10	4	0.0048	2.3E-05	1.1E-07

each node is independent and unpredictable, we assume that the probability of exiting ρ nodes and just losing the same encoded fragment is expressed as the uncoded repair failure rate of FR code. The probability of joining and exiting nodes cannot quantified in the blockchain. Without considering the probability of the nodes exit, we use $R = \frac{1}{\binom{n}{\rho}}$ to represent the uncoded repair failure rate of FR code. Therefore, in a m-levels TFR code, the uncoded repair failure rate is R^m. We analyze the values of R with different ρ and n and the value of R^m with different m. The change of the R is shown in Figure 6, when $\rho = 2, 3, 4$ and $n = 8, 9, 10, 11, 12, 13, 14, 15$. We can find that when the repetition degree ρ is fixed, the more the number of nodes, the lower the uncoded repair failure rate, namely, the higher the fault tolerance. When the number of nodes n is fixed, the greater the degree of repetition ρ, the lower the non-coding failure rate, namely, the higher the fault tolerance. In Table 4, we list the values of R^m for $m = 2, 3$ and different parameters. We can clearly see that the uncoded failure rate is exponentially reduced as the levels of TFR code increases. Therefore, the TFR code is more fault tolerant than the FR code.

VI. PERFORMANCE EVALUATION

We evaluate the performance of the GCBlock based on the effect of grouping and the performance of erasure code. All of our experiments were performed on a Lenovo ThinkStation P910 with two Intel Xeon E5-2620 v4@2.1GHz and 64G DDR4 RAM installed.

A. GROUPING IMPLEMENTATION

In order to get closer to the grouping effect of the original network, we join the Bitcoin network to collect the IP

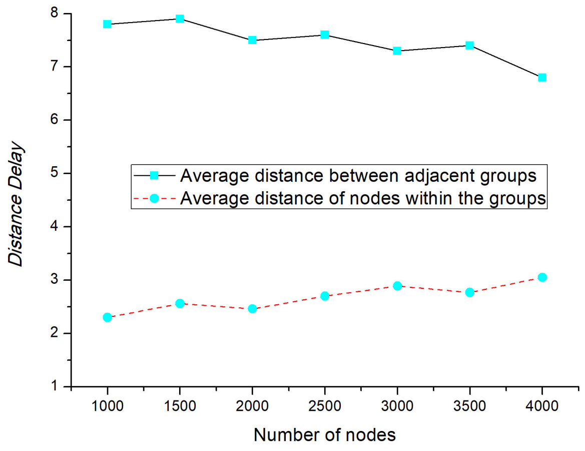


FIGURE 7. The average distance within the group and the average distance between adjacent groups according to the IP hops with the different number of bitcoin network nodes.

addresses by DNS seed [29]. The brief process: To get the IP address from the DNS seed and put it into the seed collection of IP. To create an epoll, constantly extracting IP from the seed collection to establish a TCP connection, and sniff the read and write events. As long as the addr message(a message containing neighbors' IP addresses) is received, the IP addresses are extracted into the IP seed set, and thus loops to collect the IP address in the network. The IP addresses between neighbor nodes are set one distance delay. We collected a total of 27403 IP addresses (including full nodes and light nodes). We extracted 1000, 1500, 2000, 2500, 3000, 3500, and 4000 IP addresses respectively, and constructed the adjacency matrix and distance matrix of these IP addresses to implement the beginning grouping algorithm. We set the initial selection $s = 100$ and the minimum number of nodes in a group is $n = 10$. The minimum distance between groups $D_{min} = 5$. The maximum variance within the group $\sigma_{max} = 2$. The number of iterations $v = 100$. We recorded the average distance of nodes within the group and the average distance between the adjacent groups in the experiment. The results are shown in Figure 7. It can be seen that the average distance of nodes within the groups are less than the average distance of adjacent groups. The delay between nodes is positively related to the physical distance. Therefore, grouping nodes with fuzzy distances can reduce the latency of data interaction between nodes.

B. THE PERFORMANCE OF THE ERASURE CODE

1) ENCODING AND DECODING

In order to test the performance of the TFR code, we use the Reed-Solomon code which is a good quality MDS code. Encoding is essentially a process of calculating the parity-check fragments. As the number of parity-check fragments increases, the complexity of encoding will increase. Decoding complexity depends on the finite field of the encoded fragments. The larger the finite field, the greater the complexity of decoding. We study the Backblaze [30]

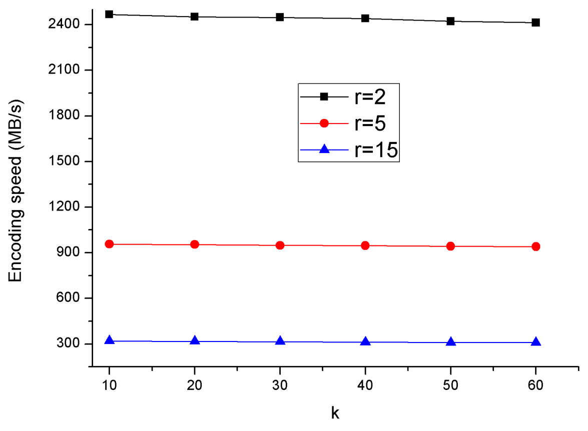


FIGURE 8. Encoding speed when the block data size is 1MB.

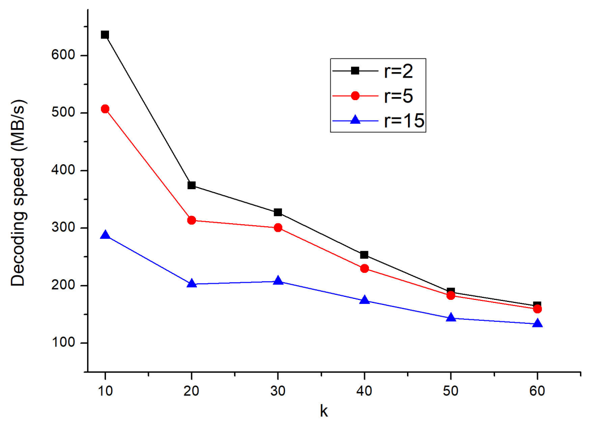


FIGURE 9. Decoding speed when the block data size is 1MB.

and implement Reed-Solomon encoding and decoding with AVX 2.0 instructions and Golang on two Intel Xeon E5-2620 v4@2.1GHz. We tested the encoding and decoding performance of a 1MB block size. The block size we chose to test is based on the standards of Bitcoin [1] and Ethereum [2]. Figure 8 and 9 show the speed of encoding and decoding on different r and k. As shown in Figure 8, the value of k has little effect on the efficiency of encoding, and the speed of encoding depends on the size of r. As shown in Figure 9, it can be seen that the efficiency of decoding depends on the number of encoded fragments. Therefore, we can set the values of k and r according to the system requirements. For example, when the operation of the underlying blockchain system has lower hardware requirements, we can set a smaller r and a smaller k.

In order to test the efficiency of Merkel hash tree generation, we implemented the Merkel hash tree generation process using Golang. We analyzed the generation efficiency of Merkel hash trees with different data sizes and different numbers of encoded fragments. In Figure 10, we can see that the generation of merkel hash tree is extremely efficient, which means that the resource consumption is extremely small. We can also see that when the number of encoded fragments is constant, the larger the data, the slower the Merkel hash tree is generated. When the data size is constant, the number of encoded fragments increases, and the slower the Merkel hash tree is generated. If a system needs to consider reducing the resources consumption of generating Merkle hash tree, we can set a smaller θ or encode smaller data.

2) EVALUATE THE BANDWIDTH CONSUMPTION

The encoded fragments in the TFR are multicast transmitted according to the levels index table, so that the duty node transmits encoded fragments only n times for nodes in different levels. When encoding a Y size data, the bandwidth consumption by the duty node transmits the encoded fragments is $\frac{Y}{k}nd$. The bandwidth consumption by each node stores the encoded fragments is $\frac{Y}{k}d$. Decoding the data requires k encoded fragments. In a decoding process, the bandwidth consumption of a node to obtain the encode fragments

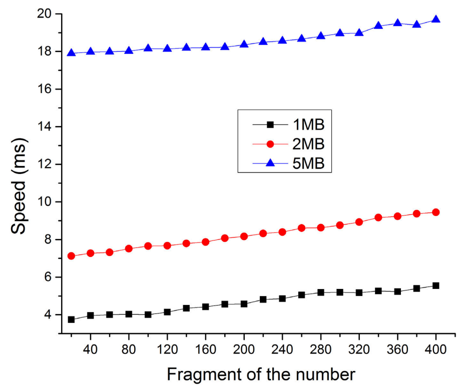


FIGURE 10. The Merkle hash tree generation speed with different data sizes and different number of encoded fragments.

is $(k-d)\frac{Y}{k}$. We evaluated the bandwidth consumption of each node storing d encoded fragments when encoding and decoding 1MB data. For example, when $k = 10, 20, 30, 40, 50, 60$ and $d = 2, 3, 4$, the bandwidth consumption of the encoding and decoding of the node changes. Figure 11 shows the change of bandwidth consumption when a node stores encoded fragments during encoding. Figure 12 shows the change of the bandwidth consumption when a node gets the encoded fragments during decoding. We can know that when d is fixed, the larger k is, the less bandwidth consumption the node stores the encoded fragments, but the more bandwidth consumption the node gets encoded fragments during decoding. When k is fixed, the larger d is, the more bandwidth consumption the node stores encoded fragments, but the less bandwidth consumption the node gets encoded fragments during decoding. Therefore, on the premise of ensuring the stable operation of the system, we can design coding parameters according to the situation of different blockchain systems. For example, when the old data of a blockchain system is used less frequently, we can set a larger k and a smaller d.

3) STORAGE OVERHEAD REDUCTION RATE (SORR)

The storage overhead reduction rate of each node is $1 - d/k$. While considering the reduction rate of storage overhead, we also consider the speed of encoding and decoding and the

TABLE 5. Encoding and decoding speed, R^m and SORR with different parameters.

(n, k, r, d, ρ, m)	Encoding speed(MB/s)	Decoding speed(MB/s)	R^m	SORR
$(22, 20, 2, 3, 3, 2)$	2451.14	373.93	1E-08	0.85
$(32, 30, 2, 3, 3, 2)$	2447.42	327.04	1.1E-09	0.9
$(32, 30, 2, 3, 3, 3)$	2447.42	327.04	3.8E-14	0.9
$(45, 40, 5, 4, 4, 2)$	947.02	229.93	7.8E-14	0.9
$(45, 40, 5, 3, 3, 2)$	947.02	229.93	1.3E-10	0.925

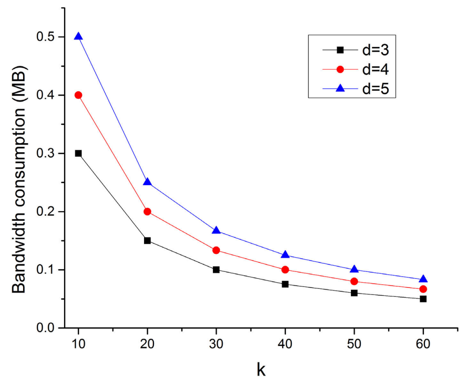


FIGURE 11. Bandwidth consumption of per node for encoding with different *k* and *d*.

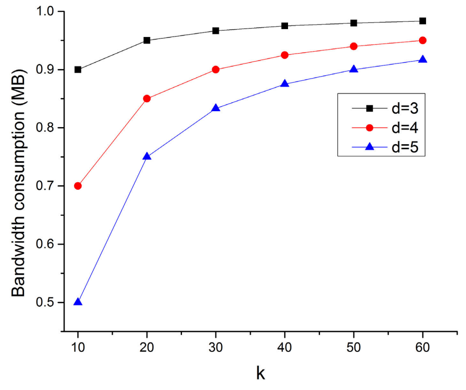


FIGURE 12. Bandwidth consumption of per node for decoding with different *k* and *d*.

uncoding repair failure rate R^m. In Table 5, we take the values of different (n, k, r, d, ρ, m) to evaluate the SORR. The values of k and r affect the speed of encoding and decoding. The values of n and ρ affect the value of R. Another expression of formula (1) is $(k + r)\rho = nd$. These parameters are related to each other. Therefore, while reducing storage, we should set different coding parameters according to different system requirements. For example, in a system that pursue encoding speed and stability, we can set a smaller r and a larger ρ.

VII. CONCLUSION

This paper presents the design of the GCBlock – an open scheme for reducing the nodes storage overhead of blockchain. GCBlock is designed to provide an option to reduce storage costs for existing public blockchain systems. Each node that joins the GCBlock can effectively reduce storage costs by implementing an erasure code scheme within the group. In order to reduce the delay of nodes acquiring data, we try to use the idea of fuzzy distance clustering to group. Within the group, we use a reconstructed TFR code that satisfies the dynamic scalability of the number of nodes to reduce the storage overhead of the nodes. In order to prevent evil behavior, we develop an autonomous check strategy and prove that it satisfies the Nash equilibrium. We have conducted extensive performance evaluation and showed that the scheme is feasible and reasonable.

Since the GCBlock does not modify the original architecture of blockchain, it can be easily used in existing public blockchain systems as a scheme to reduce data redundancy. In addition, the number of data transmission of nodes to serve peers are different in the scheme. It is an interesting topic for nodes to actively serve peers. In the future work, we will carry on the related incentive mechanism research.

REFERENCES

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

[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

[3] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. of Annual International Cryptology Conference*, Springer, Jul. 2017, p. 357—388.

[4] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 931–948.

[5] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 583–598.

[6] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology?—A systematic review," *PLoS ONE*, vol. 11, no. 10, Oct. 2016, Art. no. e0163477.

[7] H. Yu, Z. Yang, and R. O. Sinnott, "Decentralized big data auditing for smart city environments leveraging blockchain technology," *IEEE Access*, vol. 7, pp. 6288–6296, 2019.

[8] Q. Lin, H. Wang, X. Pei, and J. Wang, "Food safety traceability system based on blockchain and EPCIS," *IEEE Access*, vol. 7, pp. 20698–20707, 2019.

[9] K. Salah, N. Nizamuddin, R. Jayaraman, and M. Omar, "Blockchain-based soybean traceability in agricultural supply chain," *IEEE Access*, vol. 7, pp. 73295–73305, 2019.

[10] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Analysis of the bitcoin UTXO set," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2018, pp. 78–91.

[11] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proc. 48th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2010, pp. 1510–1517.

[12] R. K. Raman and L. R. Varshney, "Dynamic distributed storage for blockchains," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2619–2623.

[13] Z. Xu, S. Han, and L. Chen, "CUB, a consensus unit-based storage scheme for blockchain system," in *Proc. Int. Conf. Data Eng.*, Apr. 2018, pp. 173–184.
[14] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Peer-to-Peer Systems* (Lecture Notes in Computer Science), vol. 2429. Berlin, Germany: Springer, 2002, pp. 328–337.
[15] A. Babay, C. Danilov, J. Lane, M. Miskin-Amir, D. Obenshain, J. Schultz, J. Stanton, T. Tantillo, and Y. Amir, "Structured overlay networks for a new generation of Internet services," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 1771–1779.
[16] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc Int. Workshop Peer-to-Peer Syst.* Berlin, Germany: Springer, 2002, pp. 53–65.
[17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 149–160, Oct. 2001.
[18] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," in *Proc. 1st Int. Conf. Peer-to-Peer Comput.*, Aug. 2001, pp. 99–100.
[19] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in bittorrent," in *Proc. NSDI*, vol. 7, 2007, pp. 1–14.
[20] R. Bellman, R. Kalaba, and L. Zadeh, "Abstraction and pattern classification," *J. Math. Anal. Appl.*, vol. 13, no. 1, pp. 1–7, 1966.
[21] J. Nayak, B. Naik, and H. S. Behera, "Fuzzy C-means (FCM) clustering algorithm: A decade review from 2000 to 2014," in *Computational Intelligence in Data Mining*, vol. 2. Berlin, Germany: Springer, 2015, pp. 133–149.
[22] P. Chen, J. Xue, Y. Chen, S. Wei, and Y. Ji, "Resource allocation algorithm based on fuzzy cluster grouping for device-to-device communication," in *Proc. Bio-Inspired Comput.-Theories Appl.* Berlin, Germany: Springer, 2015, pp. 33–44.
[23] M. K. Goyal and V. Gupta, "Identification of homogeneous rainfall regimes in northeast region of India using fuzzy cluster analysis," *Water Resour. Manage.*, vol. 28, no. 13, pp. 4491–4511, Aug. 2014.
[24] A. E. Hassanien, E. Emary, and H. M. Zawbaa, "Retinal blood vessel localization approach based on bee colony swarm optimization, fuzzy C-means and pattern search," *J. Vis. Commun. Image Represent.*, vol. 31, pp. 186–196, Aug. 2015.
[25] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *J. Cybern.*, vol. 3, no. 3, pp. 32–57, Jan. 1973.
[26] J. C. Bezdek, "A convergence theorem for the fuzzy ISODATA clustering algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-2, no. 1, pp. 1–8, Jan. 1980.
[27] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *Proc. IEEE 26th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2007, pp. 2000–2008.
[28] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*, vol. 16. Amsterdam, The Netherlands: Elsevier, 1977, pp. 317–329.
[29] *DNS Seed*. Accessed: Nov. 20, 2018. [Online]. Available: https://bitcoin.org/en/glossary/dns-seed
[30] *Backblaze*. Accessed: Nov. 23, 2018. [Online]. Available: https://github.com/Backblaze

BIN QU received the B.S. degree in network engineering from Henan Polytechnic University, China, in 2016. He is currently pursuing the M.S. degree in software engineering with Guangxi Normal University, China. His research interests include blockchain systems , the Internet of Things, and edge computing.

LI-E WANG received the master's degree in software engineering from Hunan University, China, in 2007. She is currently pursuing the Ph.D. degree with the College of Computer Science and Information Engineering, Guangxi Normal University, China. She is also an Associate Professor with the College of Computer Science and Information Engineering, Guangxi Normal University. Her research interests mainly include data privacy, computer networking, healthcare, and distributed system security. She has published over 20 refereed articles in these areas. She served as a reviewer for several high impact research journals and ACM/IEEE flagship conferences.

PENG LIU received the Ph.D. degree from Beihang University, China, in 2017. He joined the College of Computer Science and Information Technology, Guangxi Normal University, Guilin, China, as an Assistant Professor, in 2007. Since 2015, he has been an Associate Professor. His current research interests include network security, data privacy, and graph mining.

ZHENKUI SHI received the Ph.D. degree in computer science from the City University of Hong Kong, in 2018. He is currently a Lecturer with the College of Computer Science and Information Engineering, and the College of Software, Guangxi Normal University, China. His research interests include cloud security, blockchain technology, network security, and the IoT privacy and security.

XIANXIAN LI is currently a Professor with the College of Computer Science and Information Engineering, Guangxi Normal University, China. His research interests include data security, distributed system security, the Internet of Things, and software theory. He has published over 60 refereed articles in these areas. He served as a Program Co-Chair/Technical Program Committee member for several IEEE conferences and workshops.