Consortium Blockchain Storage Optimization Based on Fountain Codes

Jianhong Li[1,2], Qi Chen[1,2(✉)], Xianmin Wang[1], Guoyu Yang[1], Zihan Jiang[1], Teng Huang[1], and Li Hu[1]

[1] Institute of Artificial Intelligence, Guangzhou University, Guangzhou 510006, China
{2112106026,hl_27}@e.gzhu.edu.cn, chenqi.math@gmail.com, xianmin@gzhu.edu.cn, huangteng1220@buaa.edu.cn

[2] State Key Laboratory of Integrated Service Networks (Xidian University), Xi'an 710071, China

Abstract. Blockchain is a distributed digital ledger with tamper-proof and privacy-preserving features. However, its use of fully replicated data storage mechanism results in high storage cost and poor scalability. This paper proposes a consortium chain encoding technique based on LT code and an LT code dual-fault-tolerant stripping decoder scheme. By combining LT code block encoding technology, the storage costs for nodes can be effectively reduced, and the dual-fault-tolerant stripping decoder scheme ensures that newly joined nodes can safely recover the complete transaction history. We simulated the proposed solution in this paper with node numbers of 20, 40, 80, 100, 200, and 300, and compared the encoding time at the same storage optimization cost with the existing popular blockchain ledger Reed-Solomon (RS) code encoding technique. We also simulated the decoding time of the proposed LT code dual-fault-tolerant stripping decoder and the RS code decoding technique in the presence of different numbers of malicious nodes interference, in order to compare their ability to securely recover the ledger. Finally, we compared the minimum number of encoding blocks required for decoding in both schemes. The results indicate that the encoding and decoding time of the RS code scheme almost linearly increase, while the proposed solution exhibits a generally flat and slow growth. Therefore, both encoding and decoding in the proposed solution are superior to the RS code scheme. In terms of the minimum number of encoding blocks required for decoding, the proposed solution requires slightly more blocks than the RS code encoding scheme. However, in the RS code scheme, the dynamic addition or removal of nodes causes other nodes in the blockchain network to perform re-encoding, that increases the bandwidth of the blockchain network. Therefore, the proposed solution effectively improves the accuracy of node recovery in the ledger, maintains the ledger's security in the consortium blockchain network, and enhances the scalability of the consortium blockchain network.

Keywords: Blockchain · Storage optimization · Erasure codes · Fountain codes

J. Cai et al. (Eds.): SPNCE 2023, LNICST 525, pp. 35–53, 2025.
https://doi.org/10.1007/978-3-031-73699-5_3

1 Introduction

Blockchain, as the foundational technology for cryptocurrencies such as Bitcoin and Ethereum, has played an important role in facilitating decentralized and trusted transactions. Moreover, it is also gradually having a transformative impact on different fields such as IoT, medicine, healthcare, and supply chain. The decentralization, security and scalability of blockchain make it have great potential.A blockchain node independently verifies each block added to the chain, stores the entire blockchain ledger, and helps newly joined nodes recover the entire blockchain ledger. Nodes with these functions are called "full nodes", which are the backbone of the blockchain network. However, running a full node on a resource-constrained device comes at a heavy price in terms of storage and computational costs. In particular, the storage requirements of the blockchain are growing at a near-exponential rate. Due to limited computing, storage, and bandwidth resources, it is difficult for resource-constrained devices to bear the consensus process of the blockchain and the storage costs of ledgers. For example, Bitcoin's total block data is currently over 200 GB, and the Ethereum network generates around 0.2 GB of data per day. And for high-throughput blockchains like Ripple [3], storage costs will be a pressing concern in the near future. The Ripple ledger is already 8.4TB in size and growing at a staggering 12GB per day.

To address the issue of storage costs, some mainstream permissionless blockchains like Bitcoin and Ethereum have introduced various technologies such as lightweight clients [22,27] and the Lightning Network [6]. Lightweight clients only store block headers to verify the block bodies received from full nodes, that effectively reduces storage overhead on the client side. The Lightning Network's fundamental idea is to facilitate multiple small-scale payments between two accounts and minimize the uploading of final balances to the blockchain, thus reducing the volume of block data and alleviating storage pressure on each node. Erasure code has also shown great performance in permissionless blockchains and is widely adopted as a new method for storage reduction [14]. In fact, it is not limited to permissionless blockchains; there have been numerous works on permissioned blockchains that leverage erasure codes for blockchain storage optimization. The authors of [16] considered a fragmented blockchain and reduced storage by computing encoded fragments through linear combinations of uncoded fragments. The authors of [25] employed key-based encryption for dynamic partitioning of the blockchain network. Reed-Solomon (RS) code [20], as one type of erasure code, has also demonstrated excellent performance in consortium blockchains. RS code can effectively reduce the storage costs of consortium blockchains and improve the efficiency of recovering historical blockchain ledger data.

However, there is still room for optimization in these approaches. While lightweight clients can reduce storage costs for clients, full nodes on the blockchain still follow the method of complete replication. Therefore, this approach is not effective in terms of storage savings. While the Lightning Network can alleviate storage pressure on each node, it does not fundamentally disrupt

the mechanism of full replication. Although previous works using erasure codes have shown excellent performance in storage optimization, especially with RS codes meeting our storage reduction requirements and breaking the mechanism of complete replication, these erasure codes involve complex decoding within a finite field, resulting in high communication costs for recovering the original data.

Therefore, in the face of the increasing cost of blockchain storage on the permissioned blockchain [21], there are three main problems as follows:

- The blockchain ledger data is redundant, and the cost of node storage of blockchain ledger data is high.
- Malicious nodes will attempt to destroy the blockchain ledger by any means. For newly added nodes, malicious nodes will attempt to prevent the newly added nodes from obtaining the complete blockchain historical ledger.
- The decoding of RS code will increase the cost of computing resources of nodes.

In summary, to meet the high demands for computing power, network bandwidth, and storage costs in permissioned blockchains, this paper proposes an optimized block storage approach based on LT code [17]. On one hand, we employ the fountain code technique with a no code rate to reduce the storage space of the blockchain. On the other hand, we design and optimize fountain code encoding strategies and secure decoding methods specifically tailored for recovering blocks damaged by malicious nodes. Additionally, we propose an LT code double fault-tolerant stripping decoder for consortium blockchain [8] to reduce recovery latency and improve decoding accuracy.

The main contributions of this paper are as follows:

- An optimization scheme of consortium blockchain storage based on fountain code is proposed to reduce the burden of blockchain ledger storage on nodes. When enough encoded blocks are generated in an epoch, each node will encode the blocks in this epoch using fountain code and store them locally. By encoding and storing blocks within an epoch locally, the storage overhead of resource-constrained nodes can be reduced. After experimental verification, it was found that LT codes have better performance, which can reduce the storage cost of the historical ledger of blockchain from $O(n)$ to $O(m)$, where m is a constant much smaller than n.
- During the block recovery process, the recovery node can visit other nodes and request to download the corresponding encoded blocks until there is a sufficient number of encoded blocks to restore the block. During the block recovery process, historical blocks can be recovered with a high probability of success by accessing a sufficient amount of data from honest nodes and using the LT code double fault-tolerant stripping decoder for safe decoding. Furthermore, experimental results have shown that LT codes require less communication cost during block recovery process compared to RS codes.
- The LT code dual fault-tolerant stripping decoder is used to decode the encoded blocks. This decoding method identifies and discards tampered

encoded blocks by matching whether the head of a single encoded block (droplet) is the same XOR value as the corresponding block head in the header chain before decoding. After decoding, the LT code dual fault-tolerant stripping decoder will verify again whether a single decoding block has been tampered with. This dual integrity check mechanism improves the efficiency and success rate of decoding.
– The performance of the proposed algorithm is verified through extensive experiments. The experimental results show that the proposed algorithm can effectively reduce the communication overhead in the consensus process and reduce the blockchain storage cost.

The rest of this paper is arranged as follows: Sect. 2 introduces the backnowlegde about erasure codes, consortium blockchain and blockchain storage. Section 3 introduces the threat model and design goals. Section 4 introduces the specific design of the protocol. Section 5 conducts the security and privacy analysis of the protocol. Section 6 introduces the simulation experiment. Finally, we summarize our contributions in Sect. 7.

2 Background

2.1 Consortium Blockchain

Permissioned blockchains [21] are blockchains that are closed or have an access control layer. This extra layer of security means that permissioned chains can only be accessed by authorized users. Permissioned users are only able to perform blockchain operations within the strict confines of roles assigned to them by the ledger administrators and require that they authenticate themselves through certificates or digital identifier methods. This additional permission restriction provides permissioned blockchains with high levels of privacy and security and the flexibility of decentralization. As a form of permissioned blockchain, the consortium blockchain provides significant control and faster processing speed while ensuring privacy and security, which makes it more efficient and secure in many aspects. The existing four major consortium blockchain are Hyperledger Fabric [2], FISCO BCOS [13], EEA Quorum [10] and R3 Corda [5]. Among them, Hyperledger Fabric, EEA Quorum and R3 Corda are foreign consortium blockchain, while FISCO BCOS is a domestic consortium blockchain.

2.2 Erasure Codes

Erasure code is a common storage system fault-tolerant technique, and Reed-Solomon (RS) code [26] is the most common one. There are two configurable parameters, K and M, in RS code. RS divides the data into K equally sized segments and encodes them to generate M redundant segments called parity. $K + M$ segments are called chunks. RS encoding guarantees that any K out of $K + M$ chunks are sufficient to recover the original data. It is important to note that erasure coding itself cannot check the correctness of the chunks, and it may recover incorrect data based on invalid chunks. Therefore, each node in the blockchain needs to verify the correctness of the received chunks.

2.3 Blockchain Storage

Storage optimization has always been one of the hotspots of blockchain network research. For this reason, many scholars have proposed many methods to reduce the storage cost of nodes. The most common techniques include lightweight clients or simplified payment verification methods, in which nodes do not need to store the entire blockchain ledger, only need to store a copy of the block header. Unlike full nodes, light clients do not participate in validating transactions and do not interact directly with the blockchain. Instead, they rely entirely on full nodes in the blockchain network. Light clients may help reduce the cost of storage, but they will increase the bandwidth of the blockchain network, and may even lead to privacy leaks, and are vulnerable to attacks such as denial of service. Another popular technique is block pruning, where nodes delete old blocks and store only the most recently committed blocks [1]. Compared with light clients, it performs better in terms of security. However, they only store a small part of the blocks and cannot help newly joined nodes restore historical blocks, which seriously damages the decentralization feature.

Erasure codes [19] are excellent at reducing the storage cost of blockchain network nodes without compromising reliability. Therefore, this paper also adopts erasure codes to reduce the storage cost of blockchain nodes. References [23] proposed low-storage nodes that split each block into small segments of fixed size and store only encoded segments. Coding fragments are obtained by linearly combining block fragments with random coefficients. Their main limitation is that they only consider the situation that a node can exit the blockchain network or the node is inaccessible, and do not consider the situation that malicious nodes provide maliciously formed coding fragments to interfere with the recovery of historical blocks. The authors in [25] considered the problem of blockchain networks with confidentiality and reduced storage costs. They propose to dynamically partition the network, then encrypt each block using a specific key for each region, and distribute the encrypted blocks among nodes in each region using a distributed storage code approach such as [9,12]. The authors in [16] consider fragmented blockchain networks and propose to compute encoded fragments by linearly combining unencoded fragments. Reed-Solomon codes (see, e.g. [20]) are the most commonly used encoding method for generating encoded fragments.Reed-Solomon codes can be used to recover the original data in the presence of a limited number of adversarial nodes that provide malicious data [20]. All these coding schemes such as random linear codes, distributed memory codes, and Reed-Solomon codes need to operate over sufficiently large finite fields and lead to high computational complexity for decoding. Therefore, the improved method based on fountain codes in this paper, especially the LT codes used, is significantly better in terms of computational cost.

Fountain codes are designed to handle random erasures. While it is possible to decode from random errors (see, for example, [11,15,18]), perturbing decoding of erroneous data can be difficult to handle. Typically, the iterative decoding algorithm of fountain codes easily propagates any errors in the received data and amplifies them in the recovered data. This is because fountain codes do not

provide any mechanism for checking the integrity of the decoded data. Therefore, compared with the previous work, the advantages of the proposed blockchain storage algorithm are as follows:

- For optimizing storage cost, the proposed blockchain storage optimization algorithm based on fountain codes not only considers the storage cost, but also comprehensively considers the selection of coding parameters and the decoding success rate of coding blocks, so as to build a blockchain network suitable for PBFT consensus mechanism. In addition, the use of fountain codes for encoding and decoding has better performance and higher security than RS codes running on finite fields. Experimental results show that the proposed blockchain storage optimization algorithm based on fountain code can effectively reduce the blockchain storage overhead and block recovery delay.
- For the reliability of block recovery and the security of the blockchain network. Fountain codes use a robust soliton distribution algorithm to select the encoded blocks, which ensures that they can be decoded with negligible failure probability when decoding. When a block needs to be recovered, the node collects the number of encoded blocks needed for recovery from other nodes to successfully decode the historical block with negligible failure probability. Moreover, after each node recovers the historical block, it verifies the Merkle root of the block header in the historical block, which effectively prevents the historical block tampering attack caused by some malicious nodes providing maliciously tampered coding blocks, and ensures the reliability of block recovery and the security of the blockchain network.

3 Threat Model and Design Goals

For blockchain storage optimization, our work is aimed at the PBFT [7] consensus on the consortium blockchain, and our main goal is to reduce the storage cost on each blockchain node. Moreover, in our designed protocol, blockchain nodes store even less data while being able to recover the complete blockchain ledger. Furthermore, they can assist newly joined nodes in recovering historical ledger data. We refer to the nodes with reduced storage cost as droplet nodes and the nodes newly joining the blockchain network as bucket nodes.

3.1 Threat Model and Assumption

Threat Model. Consider a Byzantine adversary that can control an arbitrary subset of droplet nodes. These malicious nodes may collude with each other and may disrupt the blockchain protocol in arbitrary ways. For example, sending maliciously tampered data to bucket nodes, or keeping silent. The remaining droplet nodes are honest and participate in the consensus process of the blockchain and the encoding and decoding process of historical blocks. Our hypothetical adversary randomly controls malicious nodes. That is, before selecting

the node to control, it does not know the content stored in the droplet node. Our goal is to design a protocol, as long as there are no more than f malicious nodes, the bucket node can recover the historical blocks. This paper measures the security performance of the encoding scheme by the minimum number of honest droplet nodes, which are enough to recover the historical blocks with negligible failure probability.

Assumption. The consortium chain storage optimization method which based on fountain code assumes that bucket nodes can obtain the correct block header chain first. Therefore, we assume that the majority of nodes are honest and actively participate in the consensus. Because the PBFT consensus is not like the POW consensus [4], the leader mechanism of the PBFT consensus can ensure that the blockchain will not fork.

3.2 Design Goals

For storage optimization, our main goal is to design that can achieve an optimal balance between storage savings and recovery cost in the PBFT consensus protocol. In addition, we expect the protocol to be able to have smaller bandwidth overhead and computational cost. Besides, the decentralized encoding scheme of fountain code enables droplet nodes to generate droplets without knowing the storage contents of other nodes in the blockchain network.

We measure encoding performance using the following metrics:

1. The storage saving factor of a node is the ratio of the total size of a blockchain block to the size of its stored droplets.
2. The recovery cost of the encoding scheme is measured by the minimum number of honest droplet nodes that a bucket node needs to visit to ensure that historical blocks can be recovered with a negligible probability of failure. Note that the recovery cost of an encoding scheme reflects its security performance. This is because the recovery cost can be regarded as the minimum number of honest droplet nodes that the system must contain to guarantee that historical blockchain data can be recovered. The smaller the recovery cost of an encoding scheme, the better the security performance of a system using that scheme.
3. Bandwidth overhead refers to the overhead of the amount of droplet data that the bucket node needs to download. Our goal is to control the bandwidth overhead to an appropriate threshold and restore the blockchain with a high probability.
4. The computational cost refers to the time it takes for all nodes in the entire blockchain network to decode and re-encode when a new node joins, which is a measure of the required computing resources for the blockchain network.

4 Protocol Design

4.1 System Architecture

Our coding scheme is able to enable nodes to code independently without relying on other nodes and to save storage space by storing only a small number of coding blocks. Recall that we refer to a coded block as a droplet, the node storing the coded block as a droplet node, and the newly added node to recover the historical block as a bucket node.

Encoding. We encode the droplets by the number of block growth, where the encoding period is defined as the time when the blockchain grows by k blocks. Let $B = \{B_1, B_2, ..., B_k\}$ denotes a set of k original blocks from the blockchain. This set of k original blocks will be encoded into s droplets by each node independently. Because PBFT consensus does not have a forked chain, when the number of blocks in a group does not reach k, this group of blocks will be directly stored in the local node. Once k blocks are reached, the node encodes the block to reduce the storage cost of the node. Moreover, for the encoding period, we denote by l.

Next, we define the number of nodes as n, and we select the node numbered j and observe its encoding process. For the current blockchain of height t, where t is not an integer multiple of k, j encodes k blocks in every period of k blocks, outputs s droplets, and then deletes k blocks, and the process continues. Denote the encoded output droplet of node j as $C = \left\{C_{1,1}^{(j)}, ..., C_{1,s}^{(j)}, ..., C_{l,1}^{(j)}, ..., C_{l,s}^{(j)}\right\}$, node j finally also stores $t - lk$ uncoded blocks. The specific encoding is shown in Fig. 1.

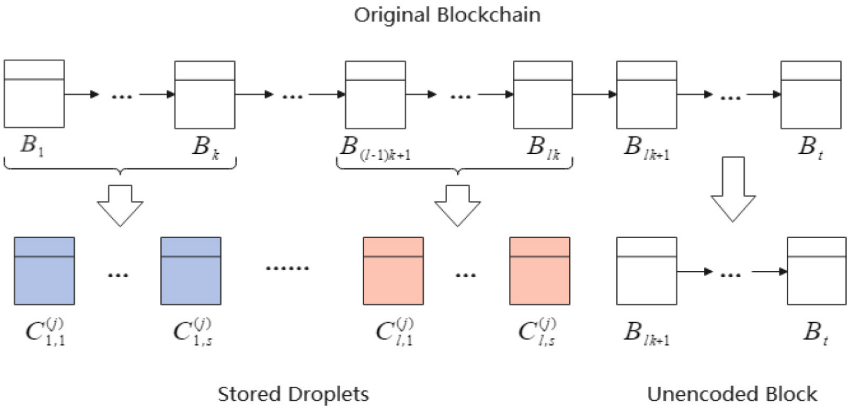

Fig. 1. Example of Node j Encoding.

Decoding. Assuming the current block height is t, after a new bucket node is added, the bucket node visits any m nodes in order to download the encoded blocks needed to restore the blockchain. We define the longest period of block encoding to be $e = \lfloor \frac{t}{k} \rfloor$; therefore, we collect droplets from the interval of periods $1 \leq l \leq e$. In addition, bucket nodes also download an additional $t - lk$ uncoded blocks from the m nodes. Bucket nodes can also access the headchains stored in other droplet nodes to verify the correctness of the decoded blocks.

4.2 Protocol Workflow

The RS code scheme is commonly used for encoding in existing permissioned blockchains. RS code is a type of erasure code that encodes a given set of K data blocks with M parity blocks, known as "parity check", resulting in $N = K + M$ encoded blocks. RS encoding ensures that any K blocks among the N blocks are sufficient to recover the original data. However, the dynamic addition or removal of nodes in the RS code scheme requires all nodes in the entire blockchain network to re-execute the encoding process, which greatly increases the bandwidth overhead of the blockchain network. In this paper, LT code is used for encoding. LT code is a type of fountain code that can generate infinite outputs from k inputs. The most important feature of fountain codes is their ability to recover k inputs with high probability from $K\,(\geq k)$ output data. Unlike the RS code, the dynamic addition or removal of nodes in the LT code scheme does not affect the nodes in the original blockchain network. Each node's encoding and decoding is independent of other nodes, so there is no need for all nodes in the blockchain network to re-encode due to the dynamic addition or removal of nodes.

The decoder for LT codes is called a stripping decoder (also known as belief propagation), and its decoding process has a high computational efficiency. However, the stripping decoder cannot handle the maliciously tampered output data. The output data method to solve this malicious tampering mainly uses the header chain of the blockchain as auxiliary information, and uses the Merkle root stored in the header chain to determine whether the output data has been maliciously tampered.

LT Code Encoder. In each encoding period, the droplet node encodes droplets according to the following steps. The node first select a positive integer d from 1 to k based on the degree distribution as the degree of the encoding block. Then, it uniformly randomly selects d blocks out of k blocks. Finally, it computes the bitwise XOR of the d blocks to obtain the droplet. Nodes store not only the generated droplets but also the index of d blocks. Nodes repeat this process to continuously generate droplets.

In the terminology of LT codes, d is called the degree of the droplet and the blocks used to compute the droplet are called neighbors. The term comes from a bipartite graph with k original input blocks as left vertices, s droplets as right vertices, and an edge connecting the block to the droplet if one of the blocks is

used to do XOR to generate the droplet. Moreover, the probability distribution of the degree of sampling on the set $\{1, 2, ..., k\}$ is called the degree distribution. This is shown in Fig. 2.

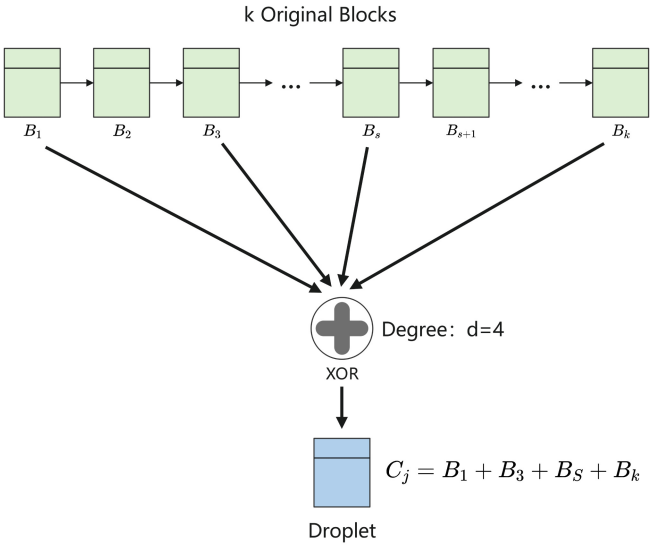

Fig. 2. An example of LT code encoding.

The description of the LT encoding process used by blockchain nodes will be described in detail below. Coding is the same for all epochs, and we arbitrarily pick an epoch for analysis. A node calculates its jth droplet $C_j (1 \leq j \leq s)$ without interfering with other nodes, the calculation process is as follows:

step 1 The degree d of the droplet is selected from the degree distribution function $\mu(\cdot)$.

step 2 Choose d blocks uniformly at random from the k blocks, and do the bit-wise XOR operation on these d blocks to generate C_j . Define $C_j = \{H_j, T_j\}$, where H_j is the first L_h bit of C_j, called the head of C_j. T_j is the $L - Lh$ bit of C_j and is the transaction data part of C_j.

step 3 Store C_j together with a random number r_j that generates a binary vector of length k through degree d.

In addition to s droplets, each node also needs to store the headchain H of the original blockchain, because the random number r_j is used to generate the binary vector and the headchain H for data integrity verification in the decoding process. The binary vector is used to identify which original blocks are combined to generate C_j, while the headchain H enables the decoder to identify maliciously formed droplets.

Let's review the possible malicious behavior of droplet nodes. In addition to remaining silent when in contact with bucket nodes, malicious droplet nodes will have the following two behaviors:

- Store any values of C_l, r_l, and H. For some encoding epoch, let G be a $k \times l$ binary matrix where the ith row corresponds to the ith block in the block. For an honest node j, the binary vectors v_j and C_j generated by r_j must have $C_j = v_j G$. However, a malicious droplet node l can store any value of C_l and r_l such that $C_l \neq v_l G$ in the binary vector v_l generated through r_l. The droplets generated by such malicious droplet nodes are called turbid droplets.
- Arbitrary degree of selectivity d, and arbitrarily choose d blocks to count droplets. But the malicious droplet node will correctly store the encoded block C_j and the random number r_j. The droplets generated by such malicious droplet nodes are called opaque droplets. The main purpose of this attack is to increase the probability of decoding failure.

A droplet computed by an honest droplet node is called a transparent droplet.

LT Code Dual Fault-Tolerant Stripping Decoder. The LT code decoder we redefined for the consortium chain is a fault-tolerant stripping decoder that can identify malicious data and discard it. Consider a newly added bucket node on the consortium blockchain, which visits m $\left(m \geq \frac{k}{s}\right)$ arbitrary droplet nodes and downloads the corresponding droplet node data. These data include droplets C_j and random numbers r_j. Label the downloaded droplets as $C_1, C_2, ..., C_{ms}$. Since the encoded droplets do not have any useful information about the blocks, bucket nodes cannot distinguish between transparent, opaque, and turbid droplets within a droplet.

We assume that bucket nodes have access to honest headchains. Honest headchains can be obtained by contacting at least $n-2f$ of the n nodes. The acquired headchain will be applied to verify the correctness of the decoded data.

Decoding is performed iteratively. At each iteration, the algorithm decodes at most one block until all blocks have been decoded, otherwise it declares a decoding failure. The decoding process is shown in Fig. 3.

Next, we describe the steps of the decoding process in the LT code dual fault-tolerant stripping decoder in detail. Let $H_1, H_2, ..., H_k$ denote the first k block headers of the honest headchain.

step 1 Initialization: Use k original blocks as left vertices and ms droplets as right vertices to form a bipartite graph Y. If block B_g is used to compute C_j, there is an edge connecting droplet C_j to original block B_g. For $g = 1, 2, \ldots, k$, initialize $\hat{B}_g$ to a null value. Let the number of iterations $i = 1$ and $Y^{i-1} = Y$.

step 2 For the received droplet C_j and random number r_j, first check whether the droplet C_j has been tampered with. The corresponding binary vector v_j is calculated through r_j, and the block head corresponding to the block chain H is XOR operated according to the binary vector v_j. Compared with the block head H_j in C_j, if it is correct, it continues to participate in the decoding process, and if it is wrong, it discards the received droplet.

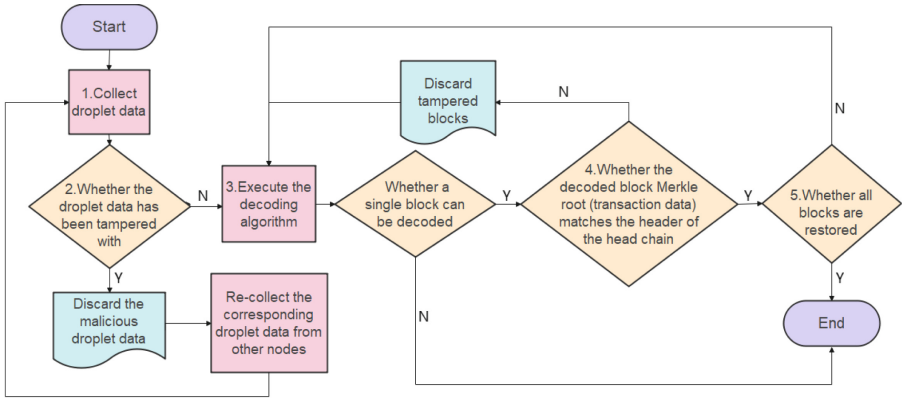

Fig. 3. The Decoding Process.

step 3 Find a droplet C_l in Y^{i-1} that is connected to exactly one block B_g, and such a droplet C_l is called a monomer. If there is no droplet of monomer, then decoding fails and the decoding algorithm is terminated.

step 4 Let H_l and T_l be the block header and transaction data part of C_l, respectively.

1. Calculate the Merkle root (T_l) of T_l. Let $\hat{B}_g = C_l$ if T_l matches the header H_g at the corresponding position in the headchain H and root (T_l) also matches the Merkle root stored in H_g.
2. Otherwise, C_l is discarded and the bipartite graph number Y^{i-1} is updated to Y^i. Let i increase by 1 and go to step 3.

step 5 For all droplet nodes $C_l^{'}$ connected to B_g in the bipartite graph Y^{i-1}, let $C_l^{'} = C_l^{'} \oplus \hat{B}_g$.

step 6 Delete all edges connected to B_g and update the bipartite graph number Y^{i-1} to Y^i.

step 7 Increase i by 1.

step 8 If all the blocks have not been decoded, go to Step 3 and continue the decoding process.

Compared with the traditional LT code decoder, the LT code decoder redefined for the consortium blockchain can distinguish the stripped decoder with the ability to detect tampered data from the classic stripped decoder in steps 2 and 4. Specifically, the classical stripped decoder always accepts the decoded block, no matter whether the block has been tampered with or not. However, the LT code decoder redefined for the consortium blockchain performs XOR on the block head at the corresponding position in the header chain through the block index vector in the droplet data in step 2 to first detect whether the droplet data has been tampered with. In this way, the tampered droplet data can be discarded without entering the decoding step, which greatly reduces the complexity of the decoding process. This method ensures that the droplet data is normal before the decoding process starts to increase the probability of successful decoding

during the decoding process. When the droplet is decoded into a single block, step 4 again verifies whether the header of the decoded block matches the corresponding block header in the header chain, further providing a mechanism for integrity checking. If the block header and Merkle root do not match the data stored in the header chain, then the fault-tolerant stripping decoder can discard the tampered data and re-download the drops for decoding to other droplet nodes.

4.3 Key Functions

Degree Distribution. Although the encoder and decoder can choose any degree, the probability of successfully decoding the input data from a given amount of output data depends on the choice of the degree distribution. In this section, we describe the robust soliton distribution proposed by Ludy [17]. In [17], robust soliton degree distributions were shown to have good success probability. The degree distribution function $\mu(\cdot)$ is a discrete probability function in the range of 1 and k integers. To describe the robust soliton distribution, the following notation is introduced in this paper. The function $\rho(\cdot)$ is defined as follows:

$$\rho(d) = \begin{cases} \frac{1}{k} & \text{for} \quad d = 1 \\ \frac{1}{d(d-1)} & \text{for} \quad d = 2, \ldots, k \end{cases} \tag{1}$$

For a given $0 < \delta < 1$ and $c > 0$, define:

$$R = c\sqrt{k} \ln\left(\frac{k}{\delta}\right) \tag{2}$$

Then, define a function $\theta(\cdot)$ as:

$$\theta(d) = \begin{cases} \frac{R}{dk} & \text{for} \quad d = 1, \ldots, \frac{k}{R-1} \\ \frac{R}{k} \ln(\frac{R}{\delta}) & \text{for} \quad d = \frac{k}{R} \\ 0 & \quad d = \frac{k}{R+1}, \ldots, k \end{cases} \tag{3}$$

The parameter δ is a bound on the probability of decoding failure after a certain number of droplets have been downloaded. The parameter c is a custom parameter that can be adjusted to optimize the number of droplets needed to recover the historical blocks of the blockchain. Summing and normalizing $\rho(\cdot)$ with $\theta(\cdot)$ yields the robust soliton distribution as follows:

$$\mu(d) = \frac{\rho(d) + \theta(d)}{\beta}, \quad \text{for} \quad d = 1, \ldots, k \tag{4}$$

where

$$\beta = \sum_{d=1}^{k} \rho(d) + \theta(d) \tag{5}$$

5 Security and Privacy Analysis

This section starts by considering a pair of encoding and decoding schemes (Enc, Dec) according to the performance of the design goals in Sect. 3. Compare the performance indicators of LT code and RS code for PBFT consensus.

5.1 Storage Savings

The storage savings of LT codes is the ratio of the total blockchain size of data to the size of the encoded droplets. The storage saving size of droplet node j is $\gamma = \frac{size(B)}{size(Enc(B,j))} = \frac{k}{m}$. And the storage saving of the RS code [24] is $\gamma = \frac{size(B)}{size(Enc(B,j))} = k$.

5.2 Recovery Cost

Considering the coding scheme with storage savings of γ in LT codes, for a given decoding failure probability $0 < \delta < 1$, the recovery cost of the historical block of the blockchain needs to be measured by the minimum number of honest droplet nodes contacted by the bucket node $K(\gamma, \delta) = \frac{k+O\left(\sqrt{k}\ln^2\left(\frac{k}{\delta}\right)\right)}{m} \leq n - 2f$ [17]. To ensure that the historical blocks of the blockchain can be successfully recovered with probability at least $1 - \delta$. Therefore, when at most $n - 2f$ honest nodes in the blockchain network are visited, the historical blocks can be successfully recovered with probability at least $1 - \delta$ using LT codes, and the recovery cost of the coding scheme reflects the security performance of the system. The recovery cost of RS codes is $K(\gamma, \delta) = \frac{k}{1} = n - 2f$. Theoretically, the recovery cost of RS codes is higher than that of LT codes.

5.3 Bandwidth Cost

Bandwidth cost refers to the minimum number of droplets that a bucket node needs to download to guarantee a high success rate for decoding. Specifically, the bandwidth cost is the number of droplets required to ensure the successful recovery of the blockchain minus the historical block size of the blockchain to be recovered divided by the encoded k block sizes. We use the probability to calculate the visited nodes as honest and malicious. When a bucket node visits a node to download data, the probability that the visited node is a malicious node is $\frac{n-2f}{n}$, and the probability that the visited node is an honest node is $\frac{2f}{n}$. Therefore, the formula of bandwidth cost can be obtained as $\beta(\gamma, \delta) = O\left(\frac{n\ln^2\left(\frac{k}{\delta}\right)}{2f\sqrt{k}}\right)$.

- Proof: From the recovery cost, we can see that the amount of downloaded data is $k + O\left(\sqrt{k}\ln^2\left(\frac{k}{\delta}\right)\right)$
- Then the bandwidth cost is $\frac{k+O\left(\sqrt{k}\ln^2\left(\frac{k}{\delta}\right)\right)-k}{k}$

- This reduces to $\frac{O\left(\sqrt{k}\ln^2\left(\frac{k}{\delta}\right)\right)}{k}$
- The amount of data downloaded by the bucket node is $\beta \times \frac{2f}{n} = \frac{O\left(\sqrt{k}\ln^2\left(\frac{k}{\delta}\right)\right)}{k}$
- Then the bandwidth cost is $\beta\left(\gamma,\delta\right) = O\left(\frac{n\ln^2\left(\frac{k}{\delta}\right)}{2f\sqrt{k}}\right)$

5.4 Computational Cost

When a new node joins the blockchain network, both the LT code and the RS code need the new node to restore the historical ledger by collecting the coded blocks of other nodes. We calculate the decoding time $T_{decode-LT}$ and $T_{decode-RS}$. The re-encoding of the LT code scheme is to re-encode new nodes, which is denoted as $T_{encode-LT}$, and the re-encoding of the RS code scheme is to re-encode all nodes, which is denoted as $T_{encode-RS}$. In the LT code and RS code schemes, the time from the new node joining the blockchain network to the end of the blockchain recoding is recorded as T_{LT} and T_{RS}, respectively. We take them as a comparison of the computational cost of the above two schemes.

6 Implementation and Evaluation

Considering the storage savings, the size of the storage savings of the LT code is $\gamma = \frac{k}{s}$ which means a node encodes k original blocks into s droplets. The storage saving size of RS code is γ=k which means a node encodes k original blocks into k droplets by default, but the node only keeps its own numbered droplet, and other droplets will be discarded, so the storage saving size is k. When we use the LT code, the bucket node needs to visit at least m $\left(\mathrm{m} \geq \frac{k}{s}\right)$ nodes in order to recover the historical ledger with high probability. During the encoding process, the larger the encoded droplet s is, the fewer nodes m need to be visited. In order to reflect the contrast of the experiment, we set s to be the same as the RS code, both of which are 1, so that the experiment can more intuitively reflect the gap between the two codes in the PBFT consensus.

We proved the feasibility of LT code applied to the consortium blockchain through theoretical analysis and carried out simulation experiments aiming at the characteristics of PBFT consensus in the consortium chain. We downloaded the blockchain data on the consortium chain and carried out simulation coding and decoding experiments locally. By setting the number of nodes $n = \{20, 40, 80, 100, 200, 300\}$ and then repeat the experiment, taking the average of the data of 100 experiments for statistics.

First, we test the difference in encoding performance between the LT code and the RS code in selecting the original coding block $k = \{8, 14, 28, 34, 68, 102\}$, as shown in Fig. 4.

Second, we test the decoding time of LT code and RS code when there are no malicious nodes and when there are malicious nodes $f = \{6, 13, 26, 33, 66, 99\}$, as shown in Fig. 5.

Third, according to the index of bandwidth overhead, we tested the minimum number of nodes that need to be visited for LT code and RS code decoding. As shown in Fig. 6.

Finally, according to the indicator of computational cost, we tested the time required for the blockchain network from the addition of a new node to the end of the re-encoding, as shown in Fig. 7.

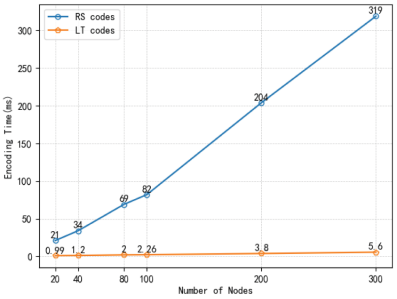

Fig. 4. Encoding Time.

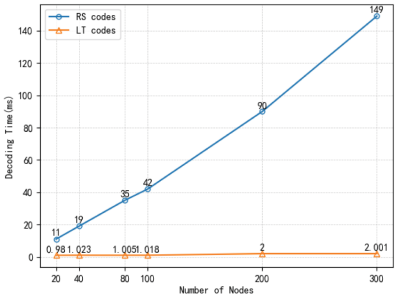

Fig. 5. Decoding Time.

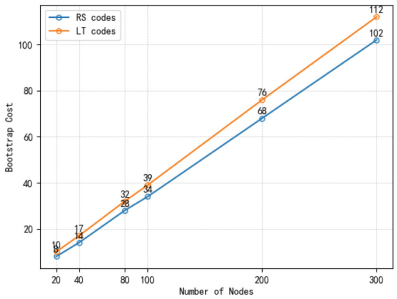

Fig. 6. Bootstrap Cost.

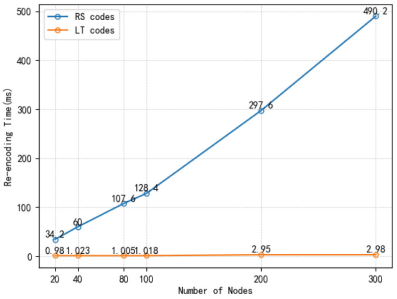

Fig. 7. Re-encoding Time.

Through the above four comparative experiments, it can be seen that when the number of nodes increases gradually, the coding performance of RS code becomes worse and worse in the first experiment, while that of LT code changes little. This is because the encoding of the LT code is an XOR operation, while the encoding of the RS code is a matrix operation. In the second experiment, the decoding performance of the RS code is also worse than that of the LT code because RS code decoding runs on a large enough finite field. Compared with LT code which does not need to run on a finite field, RS code decoding

performance is poor. In the third experiment, there is little difference between the minimum number of nodes that need to be visited by RS code and LT code when decoding. LT code needs to visit a little more nodes than RS code, because LT code decoding is a probability decoder. In order to decode with a high probability of success, the number of nodes to be visited is slightly more than the original calculated minimum number of nodes. In the fourth experiment, comparing the time taken by the RS code and the LT code from the new node joining the blockchain network to the end of the blockchain network re-encoding, we found that the calculation cost of the LT code is lower than that of the RS code. It is because the dynamic addition of nodes will cause all nodes in the RS code scheme to be re-encoded, while in the LT code scheme only new nodes need to be re-encoded, so the calculation cost of the LT code is lower.

7 Conclusion

In order to solve the problem of RS code's insufficient performance in storage optimization, this paper applies LT code to PBFT consensus for the first time. We analyze the performance of LT codes applied to PBFT consensus through four indicators: storage saving, bootstrap cost, bandwidth overhead and computing cost. We also propose a safe decoding scheme for recovering historical blocks against the problem of malicious nodes destroying bucket nodes and the security of the scheme is analyzed. Also, The integrity checking mechanism provided by the fault-tolerant stripping decoder is used to identify and discard the tampered droplets during the decoding process, which improves the efficiency and success rate of decoding. In addition, we applied the control variable method to do four comparison experiments by setting the number of nodes to compare the performance of LT code and RS code in encoding, decoding, as well as their bandwidth overhead and computational cost. The experimental results show that our proposed scheme performs better in optimizing storage cost, reducing recovery delay, and improving the accuracy and security of decoding.

Acknowledgment. The authors would like to thank the reviewers for their helpful comments and suggestions. This work was supported by the National Key Project of China (No. 2020YFB1005700).

References

1. B. core. (2015) bitcoin core version 0.11.0 released
2. Androulaki, E., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, pp. 1–15 (2018)
3. Armknecht, F., Karame, G.O., Mandal, A., Youssef, F., Zenner, E.: Ripple: overview and outlook. In: Trust and Trustworthy Computing: 8th International Conference, TRUST 2015, Heraklion, Greece, August 24-26, 2015, Proceedings 8, pp. 163–180. Springer (2015)

4. Bano, S., et al.: Consensus in the age of blockchains. arXiv preprint arXiv:1711.03936 (2017)
5. Brown, R.G., Carlyle, J., Grigg, I., Hearn, M.: Corda: an introduction. R3 CEV, August **1**(15), 14 (2016)
6. Burchert, C., Decker, C., Wattenhofer, R.: Scalable funding of bitcoin micropayment channel networks. Roy. Soc. open sci. **5**(8), 180089 (2018)
7. Castro, M., Liskov, B., et al.: Practical byzantine fault tolerance. In: OsDI, vol. 99, pp. 173–186 (1999)
8. Dib, O., Brousmiche, K.L., Durand, A., Thea, E., Hamida, E.B.: Consortium blockchains: overview, applications and challenges. Int. J. Adv. Telecommun **11**(1), 51–64 (2018)
9. Dimakis, A.G., Ramchandran, K., Wu, Y., Suh, C.: A survey on network codes for distributed storage. Proc. IEEE **99**(3), 476–489 (2011)
10. Duplantier, M., Lohou, E., Sonnet, P.: Quorum sensing inhibitors to quench P. aeruginosa pathogenicity. Pharmaceuticals **14**(12), 1262 (2021)
11. Etesami, O., Shokrollahi, A.: Raptor codes on binary memoryless symmetric channels. IEEE Trans. Inf. Theory **52**(5), 2033–2051 (2006)
12. Huang, C., Simitci, H., Xu, Y., Ogus, A., Calder, B., Gopalan, P., Li, J., Yekhanin, S.: Erasure coding in windows azure storage. In: Presented as part of the 2012 USENIX Annual Technical Conference, pp. 15–26 (2012)
13. Huizhong, L., Chenxi, L., Haoxuan, L., Xingqiang, B., Xiang, S.: Anoverview on practice of FISCO BCOS technology and application. Inf. Commun. Technol. Policy **46**(1), 52 (2020)
14. Kadhe, S., Chung, J., Ramchandran, K.: SeF: a secure fountain architecture for slashing storage costs in blockchains. arXiv preprint arXiv:1906.12140 (2019)
15. Karp, R., Luby, M., Shokrollahi, A.: Verification decoding of raptor codes. In: Proceedings. International Symposium on Information Theory, 2005. ISIT 2005, pp. 1310–1314. IEEE (2005)
16. Li, S., Yu, M., Yang, C.S., Avestimehr, A.S., Kannan, S., Viswanath, P.: PolyShard: coded sharding achieves linearly scaling efficiency and security simultaneously. IEEE Trans. Inf. Forensics Secur. **16**, 249–261 (2020)
17. Luby, M.: Lt codes. In: The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings, pp. 271–271. IEEE Computer Society (2002)
18. Luby, M.G., Mitzenmacher, M.: Verification-based decoding for packet-based low-density parity-check codes. IEEE Trans. Inf. Theory **51**(1), 120–127 (2005)
19. Luby, M.G., Mitzenmacher, M., Shokrollahi, M.A., Spielman, D.A.: Efficient erasure correcting codes. IEEE Trans. Inf. Theory **47**(2), 569–584 (2001)
20. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes, vol. 16. Elsevier (1977)
21. Miller, A.: Permissioned and permissionless blockchains. Blockchain distrib. syst. secur. 193–204 (2019)
22. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Decentralized business review p. 21260 (2008)
23. Perard, D., Lacan, J., Bachy, Y., Detchart, J.: Erasure code-based low storage blockchain node. In: 2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data, pp. 1622–1627. IEEE (2018)
24. Qi, X., Zhang, Z., Jin, C., Zhou, A.: BFT-store: storage partition for permissioned blockchain via erasure coding. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 1926–1929. IEEE (2020)

25. Raman, R.K., Varshney, L.R.: Dynamic distributed storage for scaling blockchains. arXiv preprint arXiv:1711.07617 (2017)
26. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. J. Soc. Ind. Appl. Math. **8**(2), 300–304 (1960)
27. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum proj. yellow pap. **151**(2014), 1–32 (2014)