Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.044418

ARTICLE

A Novel High-Efficiency Transaction Verification Scheme for Blockchain Systems

Jingyu Zhang[1,2], Pian Zhou[1], Jin Wang[1], Osama Alfarraj[3], Saurabh Singh[4] and Min Zhu[5,*]

[1]School of Computer & Communication Engineering, Changsha University of Science & Technology, Changsha, 410015, China

[2]Science and Technology on Information Systems Engineering Laboratory, School of Systems Engineering, National University of Defense Technology, Changsha, 410073, China

[3]Computer Science Department, King Saud University, Riyadh, 11437, Saudi Arabia

[4]Department of Artificial Intelligence and Big Data, Woosong University, Daejeon, 34606, South Korea

[5]College of Information Science and Technology, Zhejiang Shuren University, Hangzhou, 310015, China

*Corresponding Author: Min Zhu. Email: zhumin@zjsru.edu.cn

Received: 29 July 2023 Accepted: 30 October 2023 Published: 29 January 2024

ABSTRACT

Blockchain can realize the reliable storage of a large amount of data that is chronologically related and verifiable within the system. This technology has been widely used and has developed rapidly in big data systems across various fields. An increasing number of users are participating in application systems that use blockchain as their underlying architecture. As the number of transactions and the capital involved in blockchain grow, ensuring information security becomes imperative. Addressing the verification of transactional information security and privacy has emerged as a critical challenge. Blockchain-based verification methods can effectively eliminate the need for centralized third-party organizations. However, the efficiency of nodes in storing and verifying blockchain data faces unprecedented challenges. To address this issue, this paper introduces an efficient verification scheme for transaction security. Initially, it presents a node evaluation module to estimate the activity level of user nodes participating in transactions, accompanied by a probabilistic analysis for all transactions. Subsequently, this paper optimizes the conventional transaction organization form, introduces a heterogeneous Merkle tree storage structure, and designs algorithms for constructing these heterogeneous trees. Theoretical analyses and simulation experiments conclusively demonstrate the superior performance of this scheme. When verifying the same number of transactions, the heterogeneous Merkle tree transmits less data and is more efficient than traditional methods. The findings indicate that the heterogeneous Merkle tree structure is suitable for various blockchain applications, including the Internet of Things. This scheme can markedly enhance the efficiency of information verification and bolster the security of distributed systems.

KEYWORDS

Blockchain architecture; transaction verification; information security; heterogeneous Merkle tree; distributed systems

1 Introduction

Blockchain technology originally emerged from the Bitcoin system designed by Satoshi Nakamoto. As a unique distributed ledger, blockchain has become the core underlying technology of emerging virtual digital currencies [1]. Blockchain addresses the challenge of constructing a trustworthy ecosystem to meet user needs in situations without a trusted central institution [2]. The Bitcoin system stores dynamically updated encrypted transaction information. Using peer-to-peer transmission technology, the payment parties can transact directly without the verification of a central organization [3]. As a self-managed ledger, it ensures information immutability in a distributed network, making it favorable for numerous security applications [4]. Given the rapid development of digital currency in recent years, blockchain has garnered increased attention from researchers and companies [5]. The implementation of blockchain technology has begun to influence various sectors, such as electronic finance, electronic healthcare, smart homes, social security, and logistics. Compared to traditional centralized databases, blockchain's advantages encompass improved security, transparency, and greater traceability. These attributes position blockchain as an effective technological enabler capable of addressing the security and privacy concerns encountered by researchers in many disciplines [6,7].

The blockchain system fundamentally operates as a peer-to-peer network in a fully distributed environment. User nodes in the system ensure the legitimacy and accuracy of transaction information using digital signature technology. Nodes that attempt to secure bookkeeping rights through competition mechanisms, like the Proof of Work, become miner nodes [8]. These miner nodes conduct hash operations on verified transactions, package them into blocks, and add them to the chain storage structure. Every node in the blockchain can access the transaction content and must achieve consensus to determine which will be recorded in this distributed ledger [9]. The basic components of the blockchain are depicted in Fig. 1. Because the system relies on distributed storage, all nodes synchronize all block information and update data promptly [10].

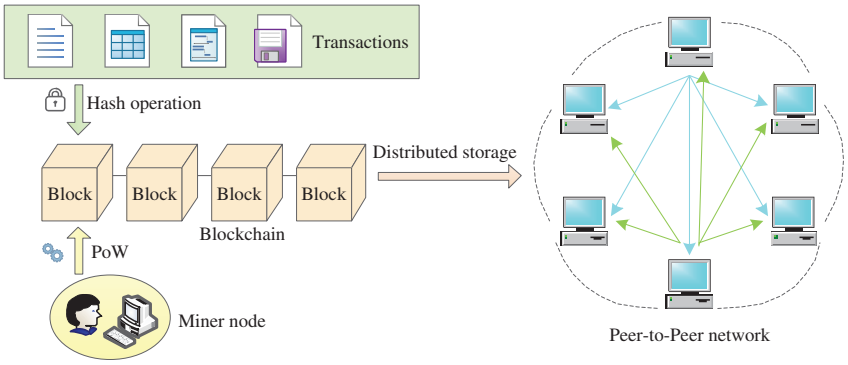

Figure 1: The basic composition of the blockchain system

Merkle trees greatly assist in verifying the integrity of block data. Comparing the previous hash in the block with a hash operation ensures that all blocks have not been tampered with before the current block is generated. If any of the blocks has been tampered with, it results in a chain change in all subsequent blocks. This is the basis for which blockchain data cannot be tampered with. All nodes involved in the system jointly maintain the normal operation of the blockchain through the established consensus. Modifying ledger data can only be realized with the consent of the majority of nodes, making tampering extremely difficult [11]. When a few nodes malfunction or there are a few

malicious nodes, the entire blockchain network system remains unaffected. Every transaction node has the right to access the data in the ledger, and because the ledger is open, it operates without the control of third-party organizations [12,13]. Generally, certain digital signature technologies, such as asymmetric encryption algorithms, are applied to the blockchain, playing roles in anonymity and user privacy protection [14].

Blockchain technology has garnered widespread attention from the academic community due to its unique storage structure and tamper-proof characteristics [15]. Existing research has proposed some blockchain-based information verification methods [16]. These methods predominantly focus on the application of blockchain in various industry scenarios, utilizing the Merkle tree structure to provide information security protection in specific fields. However, these solutions do not enhance the underlying data structure. As the scale and application scope of blockchain continues to grow, significant challenges arise from data transmission and hash operation overhead. When the system handles a large volume of transaction data, existing verification schemes prove insufficiently efficient. Based on these challenges, the aim of this study is to introduce innovative improvements to the storage structure of transactions to bolster verification efficiency. In this paper, the primary contributions are as follows:

- A novel blockchain information verification scheme is proposed in which data is organized in blocks in the form of transactions. All bottom-level transactions are no longer treated equally, and miner nodes differentiate transactions based on probability by recording the verification of historical cycles. Both the full node and the lightweight node collaborate to handle data transmission and hash calculation processes, achieving the verification of specified information.
- A new heterogeneous Merkle tree structure and design-associated algorithms are introduced for constructing this heterogeneous tree. Transactions with varying verification probabilities are stored on different layers of the Merkle tree. In this manner, distributed nodes that frequently execute transactions can readily obtain the hash values of the transactions they produce. Mathematical models are developed to quantitatively analyze the main factors influencing verification efficiency.
- Simulation experiments are designed to compare the cost of the scheme with existing methods in verification. Results indicate that the storage structure based on the heterogeneous tree structure reduces the amount of transmitted data, offering superior efficiency and performance compared to the traditional structure. There are fewer hash values to send and fewer hash operations to perform when verifying a set quantity of transactions.

The remainder of this paper is organized as follows: The subsequent section presents current research work relevant to this study. Section 3 introduces pertinent preparatory knowledge. The proposed verification scheme is detailed in Section 4. Section 5 evaluates the efficiency of transaction verification in the heterogeneous Merkle tree through the designed simulation experiments. The concluding section encapsulates the research presented in this paper.

2 Related Work

With the increase in cryptocurrency participants, the scale of transactions based on cryptocurrencies has also experienced rapid growth. The time cost of a transaction being packaged into a block is becoming increasingly important, and users need to predict the time when the transaction will be packed [17]. A model was established based on modern machine learning technology [18] to predict the approximate interval range in which miner nodes accept transactions and include them in

blocks. Research [19] proposed a model to study the impact of different miner incentives on Bitcoin transaction confirmation time, such as fee-by-byte, fee-based, and first-in-first-out. Analysis indicates that even with larger block sizes, transactions with lower fees wait longer. The average confirmation time can be derived by considering the relationship between transaction volume and past service time in the system [20,21]. It has been observed that the miner's transaction selection strategy affects the final revenue. Zhang et al. [22] introduced two methods to calculate the time of validating a single transaction. The first method estimates time-based on the previous processing time cycle, while the other approach utilizes a neural network considering various factors, such as the state of the memory pool. Zhao et al. [23] developed a non-exhaustive queue model and calculated a static probability distribution of the transaction quantity using generating functions. Consequently, the average quantity of transactions and the average validation time in the queue model can be derived based on the probability distribution. Modeling the mining procedure using a queuing system with a batch service can also be employed to calculate confirmation time [24].

Improving node verification mechanisms is essential to enhance the security of data entering the blockchain system and prevent malicious tampering of information. Research [25] proposed a lightweight payment validation protocol specifically for blockchain transactions of Internet of Things (IoT) devices. This ticket-based protocol comprises two independent logical modules: a device manager and a transaction validator, which reduces the processing and network overhead of other IoT devices. Liu et al. [26] developed a blockchain device management framework to ensure secure data exchange for IoT devices. The cloud server oversees IoT devices as domains and maintains data ledgers in each domain. Through the proposed management protocol, the framework effectively mitigates the impact of malicious nodes. Another study [27] utilized blockchain technology to develop a voting mechanism in the Internet of Vehicles, enhancing data feedback efficiency. The blockchain verifies the voting information of vehicles, meeting the criteria of high security and low latency. Sharding presents a viable solution to enhance blockchain scalability. Every node must validate all transactions in its shard. To optimize shard efficiency, Ren et al. [28] introduced a cooperative transaction verification method that enables nodes to confirm fewer transactions by sharing results. This approach guarantees that nodes achieve identical conditions under various transaction execution sequences. A new interactive signature method was utilized between transaction parties to validate transactions [29,30]. This ensures that transactions are incorporated into the blockchain in an undeniable and uncontroversial manner, preventing data falsification. Research [31] presented a blockchain-backed federated learning scheme designed for healthcare information systems. The scheme introduces a secure consensus algorithm to consistently generate blocks and optimize federated learning efficiency. Lastly, Ren et al. [32] suggested a smart home data storage framework using blockchain, which significantly minimizes storage requirements and bolsters storage performance. Within this framework, an efficient identity-based digital signature mechanism reduces the communication expense during the verification phase.

Furthermore, as blockchains encompass more nodes, alleviating load pressure has emerged as an essential priority. Without such measures, the network can quickly become overwhelmed [33,34]. Some studies have integrated the Merkle tree with other technical frameworks to enhance data storage efficiency and diminish system pressure. Zhu et al. [35] proposed a blockchain storage mechanism grounded in a convolutional tree. This mechanism employs a convolutional layer instead of the binary tree structure, effectively boosting storage efficiency. Experiments demonstrate that the number of storage nodes decreases markedly with a fixed amount of input data. The encoding Merkle tree presents as a novel hash accumulator [36]. It offers a robust defense function to thwart network data attacks, even when numerous nodes exhibit malicious intent [37]. This method allows for compact proofs of data availability attacks on any layer and also permits all nodes to validate any falsification

of the encoded tree by storing minimal bytes. Wang et al. [38] introduced a blockchain-based cloud platform data verification mechanism. This solution leverages the decentralization of the blockchain system, fusing hash operations with Merkle tree structures for verification. It effectively augments data security and verification speed on cloud platforms.

The evolution of blockchain has progressed through several phases, from the foundational Bitcoin system to the subsequent Ethereum application platform and onward to the integration of blockchain in diverse sectors. In the era marked by the swift advancement of artificial intelligence, blockchain assumes a pivotal role in various contexts, including big data, distributed computing, and construction work, among others [39–41]. Blockchain applications have grown more intricate and varied, holding the promise to invigorate emerging industries. It warrants mention that, given the influx of transaction data, safeguarding information has become paramount, and the operational demands on nodes are unparalleled. Research pertaining to information storage and verification remains a focal concern. This paper examines the verification processes of both full nodes and lightweight nodes, refining the original Merkle tree to enhance the efficiency of blockchain security verification.

3 Preliminaries

3.1 Block Structure and Merkle Tree

Each block generally consists of two parts: the block header and the block body. As shown in Fig. 2, the block header assembles information, including the version of the current block, the previous block hash, timestamp, nonce, difficulty target value, and the root hash. The specific transaction content is stored in the block body, and transactions are accessible to all nodes. Every block hash value can be obtained by performing a specific hash operation (such as SHA256) on key information in the block header, such as the Pre-hash, Nonce, and Merkle root. All blocks are linked together using hash pointers to form an irreversible chain.

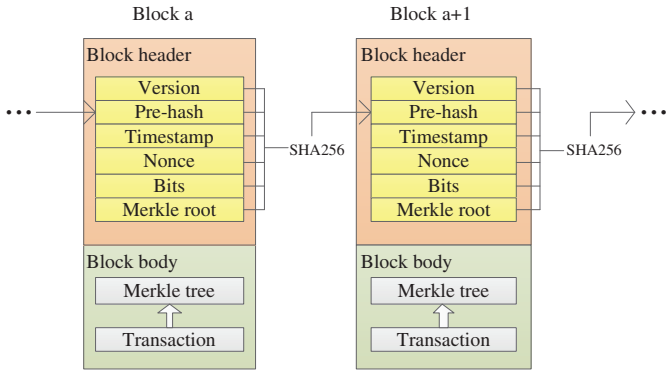

Figure 2: The composition of a block

The Merkle tree is the most typical data organization form in blockchain, designed to store and verify the integrity of block data. The Merkle tree typically contains the specific transaction content at the bottom, the root hash value at the top, and branches along the data layer to the root hash. To construct a Merkle tree, underlying transactions are generally grouped and hashed first. These generated hash values are inserted into the leaf nodes, and the hash operations are repeated until only one hash value remains.

Constructing a Merkle tree involves repeated hash operations. SHA256 is a commonly used hash algorithm known for its high computational efficiency and modest performance requirements.

It can significantly enhance operational efficiency. The Merkle root represents a summary of all transaction information, simplifying the management of the entire blockchain system. Merkle trees can segregate transaction data from hash values, thereby reducing storage requirements and pressure. When blockchain technology is applied to lightweight devices with limited storage capabilities, such as smartphones, storing the underlying data in the block body is unnecessary, which can reduce resource consumption.

3.2 Full Nodes and Lightweight Nodes

A full node contains the complete ledger data, including all block information. The responsibility of full nodes is to verify the accuracy of miner nodes in the consensus mechanism to address computing power challenges. They also verify the new blocks and transactions packaged by the miner nodes. Upon completing the verification, the full nodes broadcast the transaction information to all network nodes and eventually package it into the block. Due to the decentralized nature of the blockchain, every full node possesses a set of ledger data. If a few nodes malfunction, the security of the entire system remains unaffected. As the number of full nodes increases, more copies of the ledger information exist, making data tampering more challenging.

Full nodes contain the entire contents of all data blocks. In contrast, lightweight nodes only store the block header, significantly reducing storage space requirements. Lightweight nodes do not need to remain online continuously. They save only the transaction data relevant to them based on their needs. Since they lack the block body containing specific transactions, a lightweight node must first send a confirmation request to an adjacent full node when it needs to ascertain whether a transaction has been altered. Upon receiving the request, the full node provides hash values. The lightweight node then performs the necessary calculations and compares the result with the value provided by the full node to confirm authenticity.

4 Proposed Efficient Verification Scheme

4.1 Problem Definition

In the original tree structure, all transactions are packaged at the bottom. Transactions generated by different nodes with varying verification demands are treated similarly. Since the hash values related to a transaction must be obtained from the bottom to the root during verification, and corresponding hash operations must be performed at each layer, each transaction consumes the same amount of data transmission and hash operation overhead during its verification. This leads to significant verification costs, which affect the rate of block generation and overall system performance. Thus, the central issue is improving the verification efficiency for traditional blockchain systems in terms of data transmission and hash operation overhead.

As illustrated in Fig. 3, the underlying transaction (e) and transaction (f) are hashed to produce hash (e) and hash (f). They are packaged into the two leaf nodes of the original tree. Subsequently, hash (ef) is derived by hashing these two values again. The system repeats the hash operations for all stored transactions. Ultimately, the final hash(abcdefgh) is stored in the root node. When the transaction information (e) is selected for confirmation, the full node must provide hash(f), hash(gh), hash(abcd), and hash(abcdefgh) to the lightweight node. Through necessary calculations, it can be ascertained that transaction (e) remains untampered.

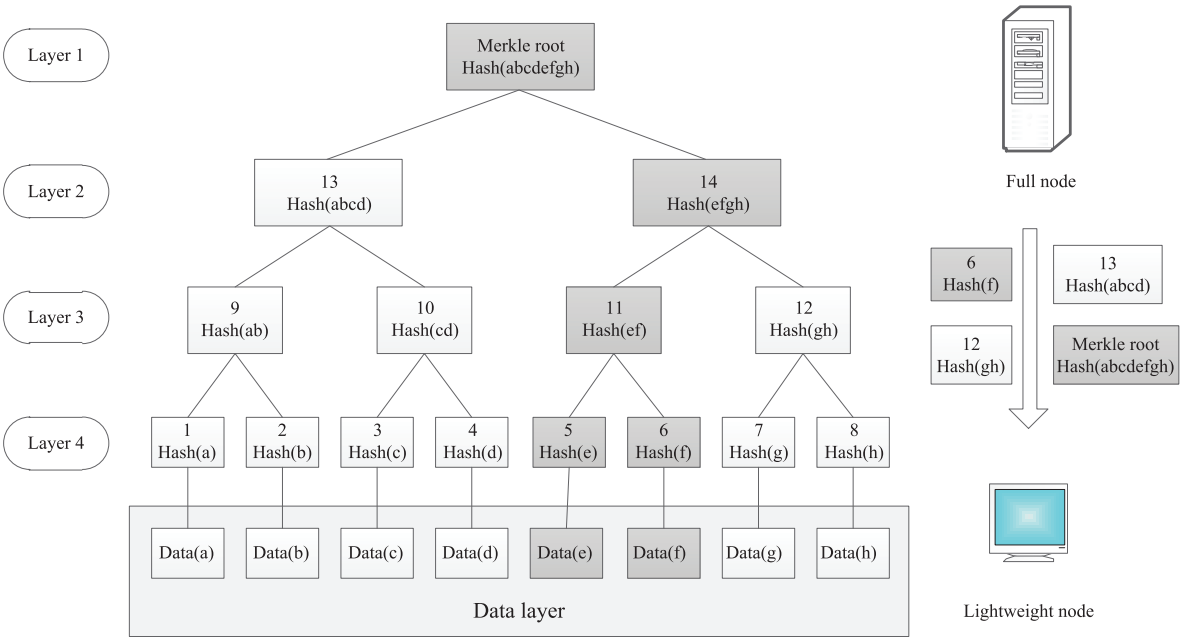

Figure 3: Transaction verification process

As transaction volume grows rapidly, the number of layers continues to expand. All transactions incur similar costs during verification. It is important to note that variations exist in the activity levels of user nodes within the blockchain system. Some active users participate in transactions frequently, signifying their frequent transaction verifications. Additionally, these active nodes comprise a small fraction of the entire system. Such observations lead to considerations of differentiating transactions based on node activity. Hence, the aim is to facilitate active users, enabling them to complete the verification process more efficiently and swiftly. As a result, the structure of the Merkle tree is made unbalanced to achieve these objectives, which include minimizing the hash values transmitted and hash operations conducted.

4.2 Overview of Proposed Scheme

As depicted in Fig. 4, the scheme consists of a node evaluation module, a verification module based on full nodes and lightweight nodes, and a heterogeneous Merkle tree storage module. For transactions packaged into the block, their verification probabilities are assessed based on historical records. Given these varying verification probabilities, transactions are stored in a novel structure called the heterogeneous Merkle tree. When a user in the blockchain system seeks to verify that a transaction has not been tampered with, the full node and the lightweight node collaborate to execute data transmission and hash operations, and the result is relayed to the user.

4.3 Node Evaluation Module

In the blockchain system, miner nodes continuously package newly arrived transactions into new blocks and verify them. The time to generate a new block is assumed to be exponentially distributed, and transactions arrive following a Poisson distribution. The node evaluation module focuses on the transaction senders and receivers. For transactions generated by the same node, the probability of being verified is equal. If certain payment parties verify more frequently during a specific period,

these user nodes are considered active nodes. The likelihood of their transactions being verified in the subsequent period is relatively high. For transaction nodes that have not verified for a duration, the transactions they produce have a reduced likelihood of verification in the next period. Before packaging a new block, the miner node first differentiates and evaluates the transaction nodes and their generated transactions. The evaluation procedure is depicted in Fig. 5.

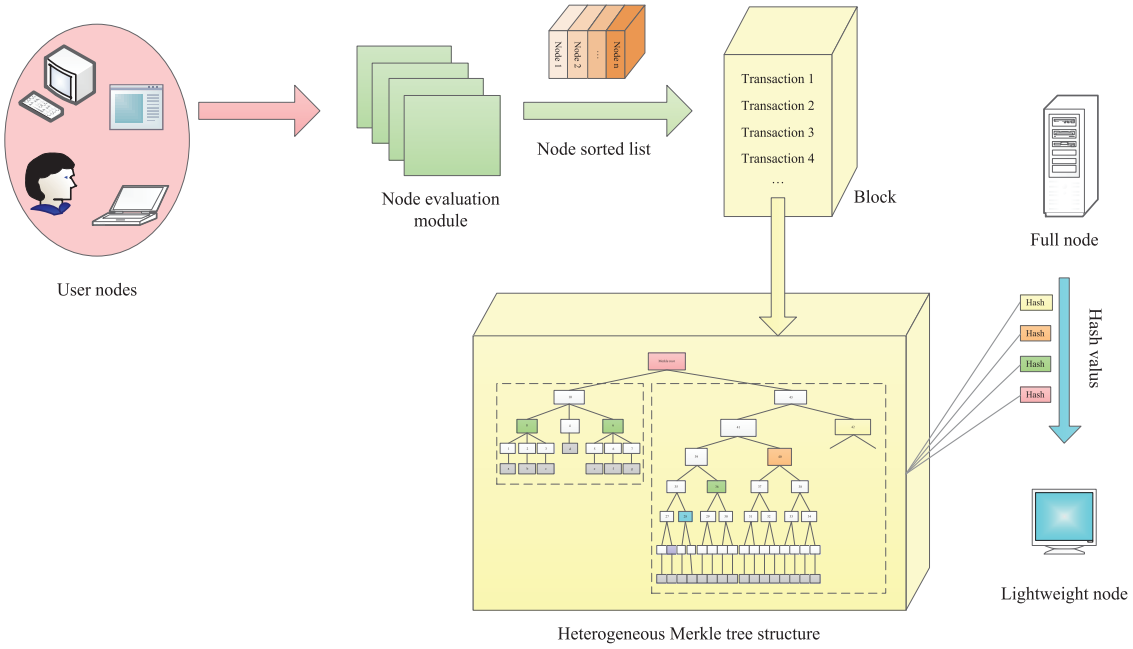

Figure 4: Workflow of our proposed scheme

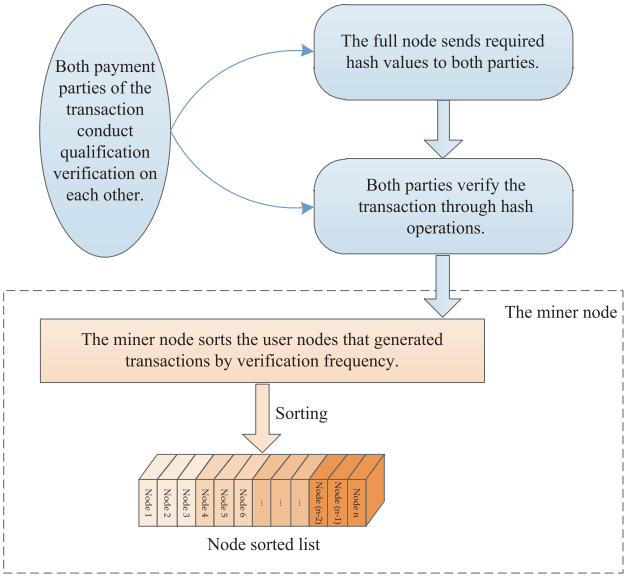

Figure 5: Node evaluation mechanism

The evaluation mechanism proceeds as follows: When a user node conducts transactions with other nodes in the blockchain network, both parties must verify each other's transaction qualifications. At this juncture, the transaction information generated by both payment parties within a certain period requires verification. Using the Merkle tree storage structure, full nodes in the network relay the necessary hash values to both parties. Each party then verifies the information's accuracy independently using hash operations. Each user node maintains a record of its verification frequency.

Miner nodes typically operate on high-performance hardware with the aim of competing for bookkeeping rights. A miner node is selected through a designated consensus mechanism to package transactions into the block body. Typically, miner nodes possess all ledger information. For pending transactions, the miner node ranks the user nodes that originated these transactions by verification frequency and then assigned probabilities to each transaction. Subsequently, it arranges the transactions based on verification likelihood from highest to lowest, preparing to insert them into the heterogeneous Merkle tree.

Algorithm 1 describes the node evaluation process. Assume that there are N user nodes in the current system, and the verification frequency of the i-th user node is denoted as $f[i]$. There are some user nodes that have not been verified, which means that their frequency is 0. According to the probability theory, it is not impossible for an event with a frequency of 0 to occur. Transactions generated by these users are still likely to be verified during the next block generation cycle. The algorithm assigns probability values to transactions generated by user nodes based on their verification status. The time complexity of the algorithm is O(n).

Algorithm 1: Node Evaluation Algorithm

Input: N: The number of user nodes
$f[i](1 \le i \le N)$: Verification frequency of the i-th node
Output: $P[i](1 \le i \le N)$: Probabilities of transactions from the i-th node

Initialize $Node_0 \leftarrow 0$, $Fre \leftarrow 0$
for each $i \in [1, N]$ **do**
    Count the total verification frequency
    $Fre \leftarrow Fre + f[i]$
    if $f[i] = 0$ **then**
        Count the nodes with a frequency of 0
        $Node_0 \leftarrow Node_0 + 1$
    end if
end for
for each $i \in [1, N]$ **do**
    if $f[i] \neq 0$ **then**
        Assign probability to transactions of the node whose frequency is not 0
        $P[i] \leftarrow (f[i] + 1)/[Fre + (N - Node_0) + 1]$
    else
        Assign probability to transactions of the node whose frequency is 0

(Continued)

Algorithm 1 (continued)

$P[i] \leftarrow 1/[Fre + (N - Node_0) + 1] * (1/Node_0)$
end if
end for
return $P[i](1 \leq i \leq N)$

4.4 Heterogeneous Merkle Tree Structure

In the heterogeneous tree proposed, the part below the root node divides into two subtrees. The structure is presented in Fig. 6. The left subtree has fewer layers, while the right subtree has more layers. Because the structure of both sides is unbalanced, it is possible to utilize the probabilistic characteristics of transactions to distinguish the transactions and place them in the appropriate subtrees. Transactions with relatively low probabilities are inserted into the data nodes of the right subtree, whereas those with high probabilities are inserted into the left subtree. The nodes of the left subtree are higher up and closer to the root node.

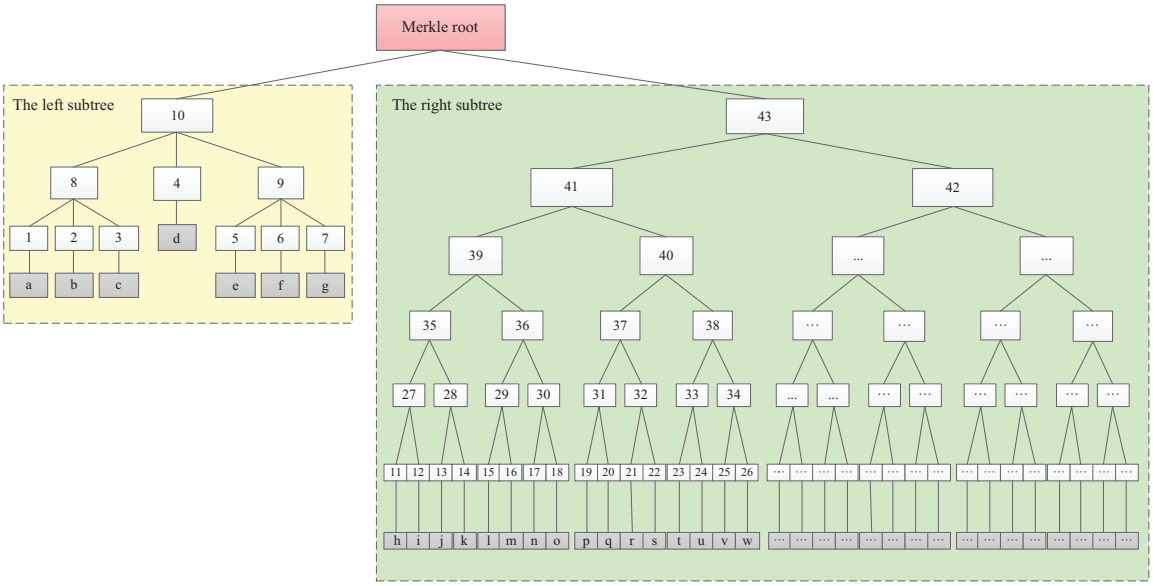

Figure 6: The heterogeneous Merkle tree

When transactions stored in the left subtree are verified, the number of hash operations that lightweight nodes must perform to obtain the root hash reduces due to the closer proximity. From another perspective, the number of hash values that need to be sent by full nodes is also likely reduced because of the higher location in this structure. For instance, when the transaction (c) is chosen by a user node for validation, the full node must provide the contents stored in nodes {1, 2, 4, 9, 43} and the root node. When transaction (k) is chosen for validation, the contents stored in nodes {13, 27, 36, 40, 42, 10} and the root node must be provided.

The layers of the two subtrees have a quantitative relationship, which is explained in subsequent analyses. Thus, it is necessary to construct subtrees with appropriate layers for data blocks containing transactions of different magnitudes. For a data block with 1088–4223 transactions, a heterogeneous Merkle tree with a left subtree of 6 layers is constructed, and the maximum number of transactions

that the left subtree can hold is 63. For a data block with 288–1087 transactions, a heterogeneous tree with a left subtree of 5 layers is recommended. The maximum number of transactions in this left subtree is 31. For data blocks with more than 4224 transactions, the corresponding numbers are 7 and 127. Based on the new probability density function of block transactions from the Bitcoin system, over 50% of blocks contain between 2000 and 3000 transactions. The previously mentioned three data block scenarios encompass most actual blockchain cases.

Algorithm 2 describes the construction process of heterogeneous trees. After processing by the node evaluation module, all transactions set to be packaged are assigned verification probabilities. Initially, the sizes of the left and right subtrees in the heterogeneous tree are determined based on the number of transactions. Then, based on the verification probability, transactions with higher probabilities are inserted into the upper nodes of the left subtree. The left subtree is filled from top to bottom. Once the left subtree is full, the remaining transactions are inserted into the lower nodes of the right subtree. The algorithm determines the number of hash values and the number of hash operations required for a single transaction's verification based on its layer. The time complexity of this algorithm is O(n).

Algorithm 2: Building Heterogeneous Merkle Tree

Input: Tx: The quantity of transactions to be packaged
$P[i](1 \le i \le Tx)$: Probabilities of the i-th transaction
Output: $Layer\_Tx[i](1 \le i \le Tx)$: Layer number of the i-th transaction
$Hash[i](1 \le i \le Tx)$: Hash values to be sent
$Operation[i](1 \le i \le Tx)$: Hash operations to be performed

Initialize $(Layer\_Tx[], R, L) \leftarrow 0$
/* Determine the layer number of the subtree */
$R \leftarrow \log_2 Tx + 1$ // right subtree
$L \leftarrow \text{int}(R/2)$ // left subtree
/* Store transactions based on probability*/
for each $i \in [1, Tx]$ **do**
  if $i \le 2^L - 1$ **then**
    Store transactions in the left subtree
    if $i \le 2^{L-1} - 1$ **then**
      Store transactions on the upper layers
      $Layer\_Tx[i] \leftarrow \text{int}(\log_2 i + 3)$
    else
      Store transactions on the lowest layer
      $Layer\_Tx[i] \leftarrow L + 1$
    end if
    $Hash[i] \leftarrow 2(Layer\_Tx[i] - 1)$
    $Operation[i] \leftarrow Layer\_Tx[i] - 1$
  else
    Store transactions in the right subtree
    $Layer\_Tx[i] \leftarrow R + 1$
    $(Hash[i], Operation[i]) \leftarrow Layer\_Tx[i]$
  end if
end for
return $Layer\_Tx$, $Hash$, $Operation$

In the heterogeneous Merkle tree proposed, the right subtree is a conventional binary tree, while the structure of the left subtree is distinct. The left subtree is a Merkle tree that performs hash operations on three hash values at once. The advantage of this rule is that when the number of transactions to build a block body remains constant, the layers of the left subtree are fewer than those of the binary tree. However, not every hash value adheres to this rule. Some fast nodes in the left subtree sit between the two values generated by the lower layer and store the hash of only one transaction. Compared to the transactions at the bottom of the left subtree, those in the fast nodes are closer to the root. When verifying the transactions in these fast nodes, both the number of operations performed and the hash values needed for transmission decrease. For instance, when the transaction (e) is chosen by a user node for validation, the full node must provide the contents stored in nodes {6, 7, 4, 8, 43} and the root node. To verify the data (d)'s accuracy, the values in nodes {8, 9, 43} and the root node are necessary.

In practical application scenarios, transaction information is continually produced for packaging into blocks. Thus, the layers of the two subtrees in the Merkle tree will grow. As the number of layers rises, the verification speeds for transactions in the two subtrees will differ more significantly. Transactions in higher-positioned data nodes in the left subtree are verified with greater efficiency.

Most blockchain systems employ Merkle trees to store transactions, suggesting that the heterogeneous tree structure is not limited to the Bitcoin system but is also suitable for various blockchain systems. This structure can enhance the Ethereum system, as Ethereum still uses the Merkle tree structure to package data, even though it employs the Merkle Patricia Tree (MPT) structure for account status recording. Regardless of the consensus algorithm in operation, if a system organizes its transactions via the Merkle tree structure, this method is effective. Thus, this scheme's potential applications include public blockchain systems (e.g., Bitcoin, Ethereum), alliance blockchain systems, and private systems prevalent in fields with frequent transaction verification.

4.5 Modeling and Theoretical Analysis

The verification process involving full nodes and lightweight nodes for the heterogeneous Merkle tree is modeled. Mathematical models are employed to quantitatively analyze the primary factors influencing verification efficiency, including full nodes' data transmission volume and lightweight nodes' hash operation overhead. Table 1 displays the definitions of relevant symbols used in the modeling.

Table 1: Notations

Notation	Description
l	The layers of the left subtree
r	The layers of the right subtree
M	The quantity of hash values sent by the full node
N	The quantity of hash operations carried out by the lightweight node
M_i	The quantity of hash values sent when verifying a transaction at the layer i
N_i	The number of hash operations when verifying a transaction at the layer i
M_p	The number of hash values when verifying p transactions
N_p	The number of hash operations when verifying p transactions
r_k	The layers of the right subtree in the block k

(Continued)

Table 1 (continued)

Notation	Description
p_k	The number of transactions selected for verification in the block k
C_k	The number of fast nodes participating in the verification in the block k

Before building the heterogeneous tree, the miner node first executes the node evaluation mechanism and sorts the nodes according to the possibilities from high to low. The aim is to distinguish the transactions generated by all payment nodes.

To build a heterogeneous tree, determine the layers first. After counting the number of transactions packaged into the current block, the layers of two subtrees can be calculated. The layers of the left subtree are denoted by l, and the layers of the right subtree are denoted by r. The heterogeneous tree can improve the verification efficiency only when the layers of two subtrees follow the relation described in the following Eq. (1):

$$r + 1 > 2l \tag{1}$$

The right subtree is an original binary tree, and transactions are located at the bottom layer. When a single transaction in the right subtree is verified, the quantity of hash values M and the quantity of hash operations N are calculated by Eq. (2).

$$M = N = r + 1 \tag{2}$$

When the transaction in the left subtree is verified, due to the particularity of the structure, the transactions stored at different layers have different verification speeds. If the transaction is at the layer i, the relation between the number of hash values M_i and the value i is expressed by Eq. (3). The relation between the number of hash operations N_i and the value i is expressed by the Eq. (4).

$$M_i = 2\mathrm{i} \tag{3}$$

$$N_i = \mathrm{i} + 1 \tag{4}$$

When p transactions stored in the right subtree are verified at the same time, the number of hash values M_p and the number of hash operations N_p are calculated by the Eq. (5).

$$M_p = N_p = p * (r + 1) \tag{5}$$

When p transactions stored in the left subtree are verified at the same time, if p satisfies Eq. (6), the number of hash values is shown in Eq. (7). The number of hash operations is shown in Eq. (8), where j is the parameter that makes p satisfy the Eq. (9).

$$p < \sum_{i=2}^{l-2} 2^{i-2} \tag{6}$$

$$M_p = 2\left(p - \sum_{i=2}^{j} 2^{i-2}\right)(j+1) + 2\mathrm{i}\sum_{i=2}^{j} 2^{i-2} \tag{7}$$

$$N_p = \left(p - \sum_{i=2}^{j} 2^{i-2}\right)(j+2) + (i+1)\sum_{i=2}^{j} 2^{i-2} \tag{8}$$

$$\sum_{i=2}^{j} 2^{i-2} \le p \le \sum_{i=2}^{j+1} 2^{i-2} \tag{9}$$

If p satisfies Eq. (10), the number of hash values is expressed by Eq. (11). The number of hash operations is expressed by Eq. (12).

$$p \ge \sum_{i=2}^{l-2} 2^{i-2} \tag{10}$$

$$M_p = 2i\sum_{i=2}^{l-1} 2^{i-2} + 2l\left(p - \sum_{i=2}^{l-1} 2^{i-2}\right) \tag{11}$$

$$N_p = (i+1)\sum_{i=2}^{l-1} 2^{i-2} + (l+1)\left(p - \sum_{i=2}^{l-1} 2^{i-2}\right) \tag{12}$$

When multiple transactions in multiple blocks are verified simultaneously, the left and right subtrees also present different situations due to their different structures. If p transactions stored in the right subtree in q blocks are verified at the same time, the quantity of layers in the right subtree in the block k is r_k, and the quantity of transactions selected for verification in the block k is p_k. The quantity of hash values M and the quantity of hash operations N follow Eq. (13).

$$M = N = \sum_{k=1}^{q} p_k\,(r_k + 1) \tag{13}$$

When p transactions stored in the left subtree in q blocks are verified simultaneously, the quantity of hash values M is given by Eq. (14). The quantity of hash operations N is given by Eq. (15). Among them, j_k is the parameter that makes p_k satisfy Eq. (9) in the block k, and C_k is the quantity of fast nodes participating in the verifying process in the block k.

$$M = \sum_{k=1}^{q}\left[2i\sum_{i=2}^{j_k} 2^{i-2} + 2\,(j_k+1)\,(p_k - C_k)\right] \tag{14}$$

$$N = \sum_{k=1}^{q}\left[(i+1)\sum_{i=2}^{j_k} 2^{i-2} + (j_k+2)\,(p_k - C_k)\right] \tag{15}$$

5 Performance Evaluation and Discussion

5.1 Experimental Design

In this section, simulation experiments were carried out on both the original tree and the proposed heterogeneous tree storage structure. Analogous to certain characteristics in the Bitcoin system, the block size remains fixed while the capacity varies. A block was set at 1 MB in the experiments. The number of transactions a block can hold is not constant but varies randomly. Fig. 7 displays the probability density function of block transactions. The data was sourced from nodes monitoring the Bitcoin system and represents information for 516 entire blocks. Experiments were conducted on four different datasets to enhance the credibility of the simulation experimental results. These datasets contained 800, 1500, 3000, and 4500 transactions. During the experiments, the independent variable

was the number of transactions, and the dependent variables were the number of hash values required to be sent and the number of hash operations performed to obtain the root hash.

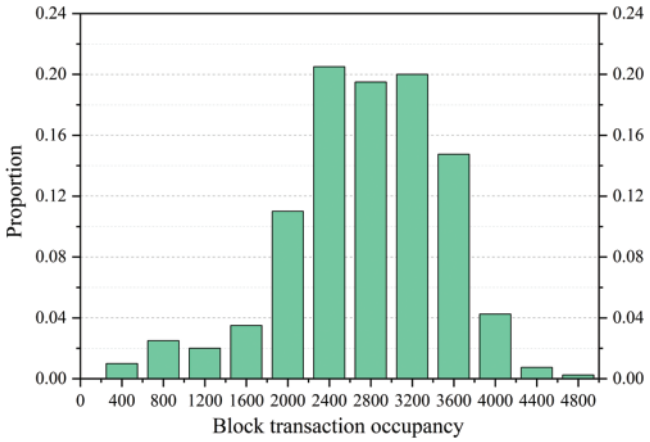

Figure 7: The possibility density function in the Bitcoin system

Based on the changes in the number of transactions, the four datasets were tested with both the original tree and two subtrees of the heterogeneous tree. Through calculations, the original tree structure constructed by the dataset with 800 transactions consisted of 11 layers. The corresponding layers for 1500, 3000, and 4500 transactions were 12, 13, and 14, respectively. In the heterogeneous tree structure, the number of layers of the right subtree constructed by the dataset with 800 transactions was 11, while the left subtree had 10 layers. The layers of the left subtree constructed by the 1500, 3000, and 4500 transactions were consistently one fewer than their corresponding right subtrees. Transactions were then randomly selected from the datasets for verification. Comparisons of the number of hash values and the number of hash operations are shown in Figs. 8 and 9, respectively.

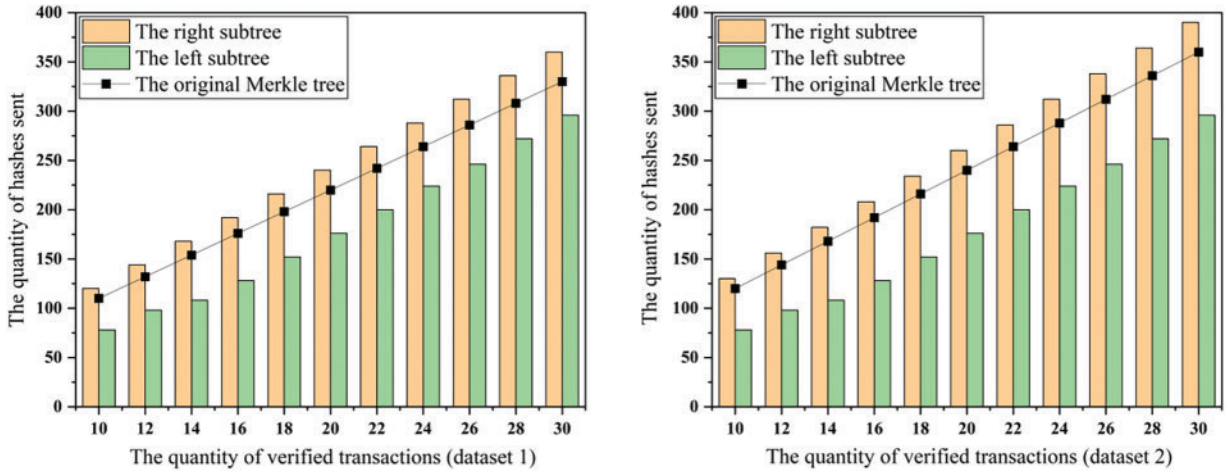

Figure 8: (Continued)

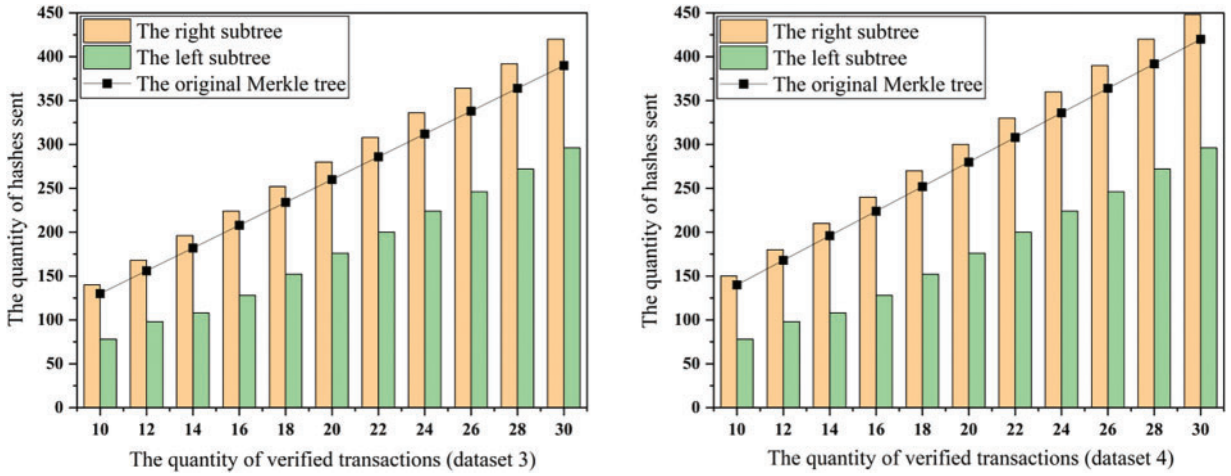

Figure 8: The experimental data for the number of hashes sent

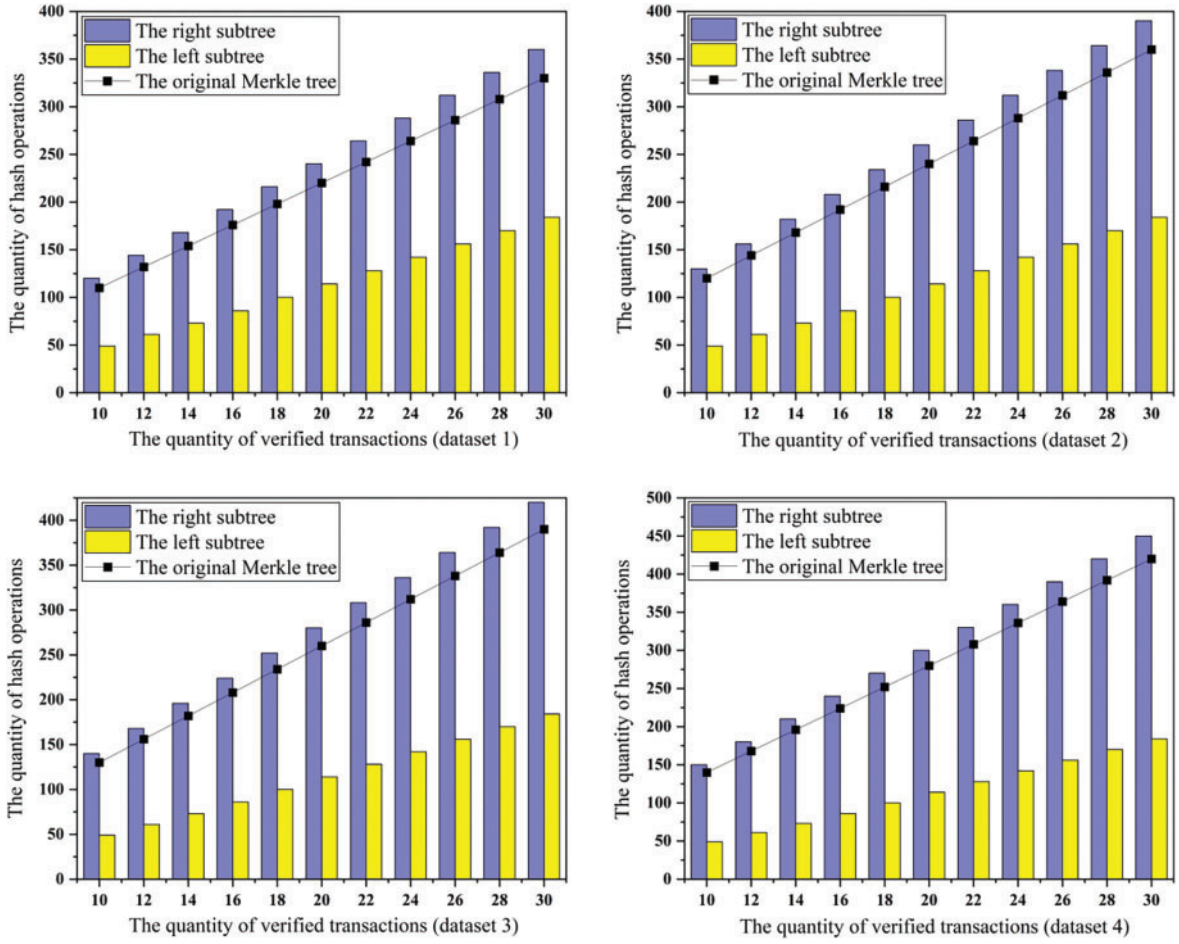

Figure 9: The experimental data for the number of hash operations

To demonstrate the superiority of the proposed verification scheme, comparative experiments were conducted alongside the existing conventional method BMCS [38]. The number of hash values required to be sent and the number of hash operations performed during the verification process in both schemes are depicted in Fig. 10.

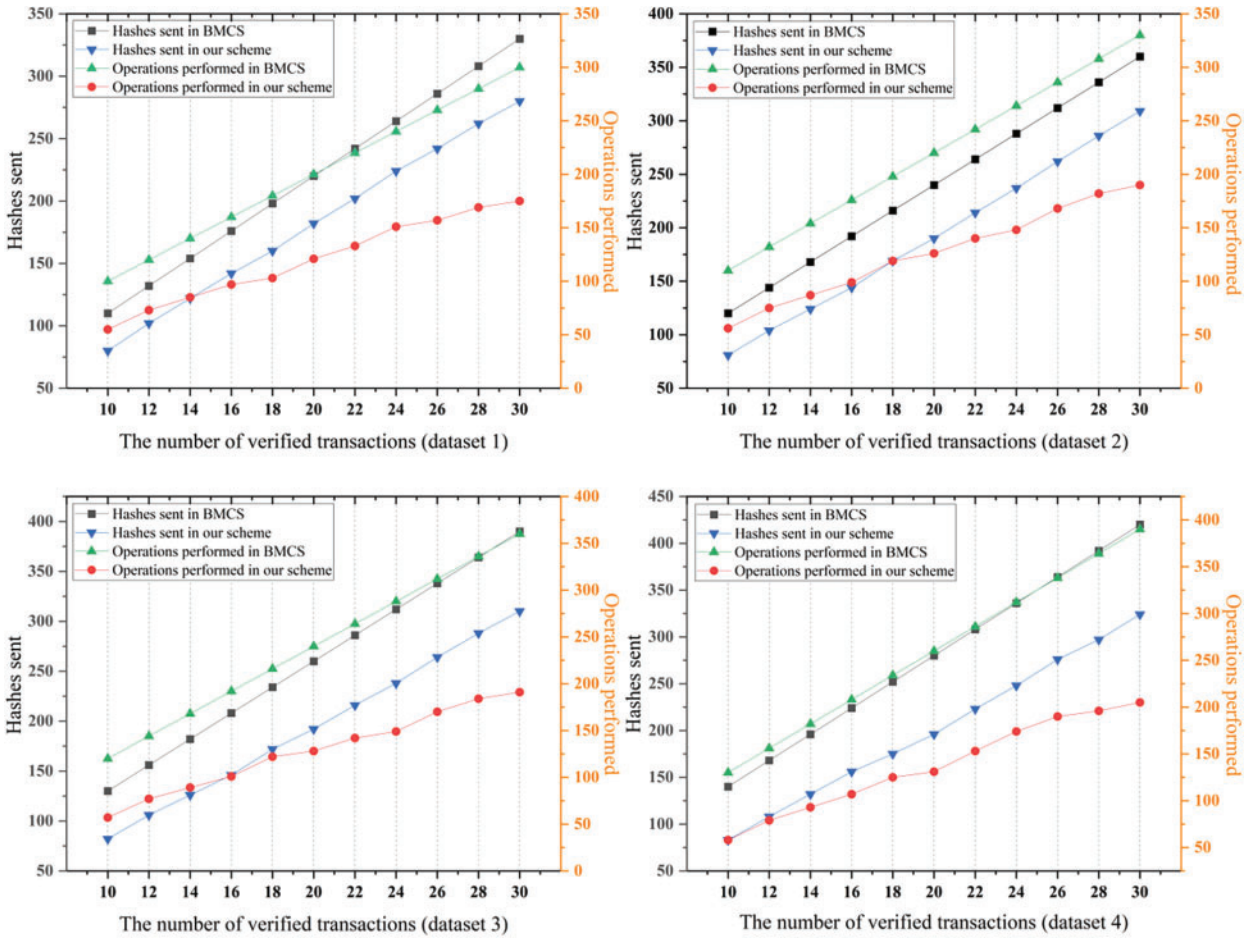

Figure 10: The comparison of verification cost

5.2 Performance Analysis

Since the right subtree adopts the original structure in the heterogeneous Merkle tree, the number of hash values and the number of hash operations should be equivalent. However, due to the unique characteristics of the left subtree, they are generally unequal when transactions from different layers are selected for verification.

It can be seen from Fig. 8 that in both the original tree and two subtrees of the heterogeneous tree, the number of hash values that the full node must send increases as more transactions are verified. When the number of verified transactions remains constant, the number of hash values in the right subtree of the heterogeneous tree is slightly greater than that in the original tree. Yet, the number of hash values in the left subtree for verification is significantly fewer than in the original tree. This discrepancy arises because the layers of the heterogeneous tree formed solely by the right subtree exceed

those of the original tree when the number of transactions remains constant. Since most nodes in the left subtree are formed by three values, the positions of certain transactions in the left subtree are much higher than in the original tree.

Fig. 9 shows that regardless of the structure, the number of hash operations carried out by the lightweight node increases as more transactions are verified. Compared with the original tree structure, the number of hash operations performed by the left subtree is always significantly fewer when verifying the same number of transactions. However, the number of hash operations in the right subtree is slightly greater, aligning with the results of the previous experiment.

As illustrated in Fig. 10, this scheme's data transmission volume and hash operation overhead are significantly less than those in BMCS. A constraint exists between the layers of two subtrees in the heterogeneous tree, ensuring that the left subtree remains smaller when filled with transactions first than the right subtree. Based on the designed algorithm, frequently verified nodes are filtered out, and the transactions they generate are positioned at the higher layers of the left subtree. Verification necessitates obtaining the hash values from the transaction location to the root, with hash operations performed at each layer. Consequently, the number of hash values sent by full nodes in this solution decreases, and the number of hash operations performed by lightweight nodes in the heterogeneous tree is further reduced.

This scheme consistently demonstrates superior efficiency in experimental datasets containing varying numbers of transactions. Verifying a fixed number of transactions requires less data transmission and fewer hash operations. Additionally, as the number of transactions to be verified increases, the difference in the number of hash operations widens, and the performance advantage of this solution becomes more obvious.

6 Conclusion and Future Work

As blockchain becomes more popular and widely used, the transaction data packaged into the block experiences rapid growth. Preventing malicious tampering by attackers and ensuring the security of transaction information verification remains paramount. Enhancing the efficiency of transaction verification by nodes responsible for packaging blocks is crucial. To address this issue, this paper introduces an efficient verification scheme based on a heterogeneous Merkle tree storage structure. Initially, all newly generated transactions are probabilistically evaluated. Subsequently, transactions are stored in distinct locations based on their respective verification probabilities. The performance advantage of this scheme was then confirmed through theoretical analysis and comparative experiments. Due to its unique design, heterogeneous Merkle trees consistently outperform in verification. This proposed scheme minimizes the number of hash values sent and hash operations performed during verification. It can be integrated into distributed systems with blockchain as the foundational architecture, such as the Internet of Things. By storing transactions generated by active nodes that frequently undertake the verification process at elevated positions, the speed of block generation and transaction verification increases. This approach markedly decreases the cost and data transmission demands of information verification, enhancing the performance of blockchain systems. In future research, building on this paper's findings, blockchain technology's features can be further explored to bolster the security and privacy of distributed systems.

Acknowledgement: The authors would like to sincerely thank the editor and reviewers for the valuable comments and suggestions, which significantly improved this paper.

Funding Statement: This work is funded by the National Natural Science Foundation of China (62072056, 62172058) and by the Researchers Supporting Project Number (RSP2023R102) King Saud University, Riyadh, Saudi Arabia. It is also funded by the Hunan Provincial Key Research and Development Program (2022SK2107, 2022GK2019), the Natural Science Foundation of Hunan Province (2023JJ30054), the Foundation of State Key Laboratory of Public Big Data (PBD2021-15), the Young Doctor Innovation Program of Zhejiang Shuren University (2019QC30), and Postgraduate Scientific Research Innovation Project of Hunan Province (CX20220940, CX20220941).

Author Contributions: Study conception and design: J.Z., P.Z.; data collection: J.W.; analysis and interpretation of results: O.A., S.S.; draft manuscript preparation: J.Z., P.Z.; supervision: M.Z. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data supporting the findings of this study are available from the first author upon reasonable request.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Jia, X., Xu, J., Han, M., Zhang, Q., Zhang, L. et al. (2023). International standardization of blockchain and distributed ledger technology: Overlaps, gaps and challenges. *Computer Modeling in Engineering & Sciences, 137(2),* 1491–1523. https://doi.org/10.32604/cmes.2023.026357
2. Han, D., Chen, J., Zhang, L., Shen, Y., Gao, Y. et al. (2021). A deletable and modifiable blockchain scheme based on record verification trees and the multisignature mechanism. *Computer Modeling in Engineering & Sciences, 128(1),* 223–245. https://doi.org/10.32604/cmes.2021.016000
3. Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Technical Report*.
4. Rahman, Z., Yi, X., Mehedi, S. T., Islam, R., Kelarev, A. (2022). Blockchain applicability for the Internet of Things: Performance and scalability challenges and solutions. *Electronics, 11(9),* 1416.
5. Wang, J., Chen, W., Wang, L., Sherratt, R. S., Alfarraj, O. et al. (2020). Data secure storage mechanism of sensor networks based on blockchain. *Computers, Materials & Continua, 65(3),* 2365–2384. https://doi.org/10.32604/cmc.2020.011567
6. Hu, N., Teng, Y., Zhao, Y., Yin, S., Zhao, Y. (2021). IDV: Internet domain name verification based on blockchain. *Computer Modeling in Engineering & Sciences, 129(1),* 299–322. https://doi.org/10.32604/cmes.2021.016839
7. Seok, B., Park, J., Park, J. H. (2019). A lightweight hash-based blockchain architecture for industrial IoT. *Applied Sciences, 9(18),* 3740.
8. Zhang, W., Wu, Z., Han, G., Feng, Y., Shu, L. (2020). Ldc: A lightweight dada consensus algorithm based on the blockchain for the Industrial Internet of Things for smart city applications. *Future Generation Computer Systems, 108,* 574–582.
9. Biswas, S., Sharif, K., Li, F., Maharjan, S., Mohanty, S. P. et al. (2019). PoBT: A lightweight consensus algorithm for scalable IoT business blockchain. *IEEE Internet of Things Journal, 7(3),* 2343–2355.
10. Foytik, P., Shetty, S., Gochhayat, S. P., Herath, E., Tosh, D. et al. (2020). A blockchain simulator for evaluating consensus algorithms in diverse networking environments. *Proceedings of the 2020 Spring Simulation Conference (SpringSim)*, pp. 1–12. Fairfax, USA.
11. Ali, G., Ahmad, N., Cao, Y., Khan, S., Cruickshank, H. et al. (2020). xDBAuth: Blockchain based cross domain authentication and authorization framework for Internet of Things. *IEEE Access, 8,* 58800–58816.

12. Zhang, J., Zhong, S., Wang, T., Chao, H. C., Wang, J. (2020). Blockchain-based systems and applications: A survey. *Journal of Internet Technology, 21(1),* 1–14.
13. Liao, Z., Pang, X., Zhang, J., Xiong, B., Wang, J. (2021). Blockchain on security and forensics management in edge computing for IoT: A comprehensive survey. *IEEE Transactions on Network and Service Management, 19(2),* 1159–1175.
14. Zhang, J., Zhong, S., Wang, J., Yu, X., Alfarraj, O. (2021). A storage optimization scheme for blockchain transaction databases. *Computer Systems Science and Engineering, 36(3),* 521–535. https://doi.org/10.32604/csse.2021.014530
15. Abdelmaboud, A., Ahmed, A. I. A., Abaker, M., Eisa, T. A. E., Albasheer, H. et al. (2022). Blockchain for IoT applications: Taxonomy, platforms, recent advances, challenges and future research directions. *Electronics, 11(4),* 630.
16. Jiang, L., Xie, S., Maharjan, S., Zhang, Y. (2019). Joint transaction relaying and block verification optimization for blockchain empowered D2D communication. *IEEE Transactions on Vehicular Technology, 69(1),* 828–841.
17. Ahuja, A., Vigneswaran, R., Rajan, M., Lodha, S. (2023). Myochain: A blockchain protocol for delay bounded transaction confirmation. *Proceedings of the 15th International Conference on Communication Systems & Networks (COMSNETS)*, pp. 114–118. Bangalore, India.
18. Singh, H. J., Hafid, A. S. (2019). Prediction of transaction confirmation time in ethereum blockchain using machine learning. *Proceedings of the Blockchain and Applications: International Congress*, pp. 126–133. Avila, Spain.
19. Gebraselase, B. G., Helvik, B. E., Jiang, Y. (2021). Effect of miner incentive on the confirmation time of bitcoin transactions. *Proceedings of the 2021 IEEE International Conference on Blockchain*, pp. 521–529. Melbourne, Australia.
20. Balsamo, S., Malakhov, I., Marin, A., Mitrani, I. (2022). Transaction confirmation in proof-of-work blockchains: Auctions, delays and droppings. *Proceedings of the 20th Mediterranean Communication and Computer Networking Conference (MedComNet)*, pp. 140–149. Pafos, Cyprus.
21. Kasahara, S. (2021). Performance modeling of bitcoin blockchain: Mining mechanism and transaction-confirmation process. *IEICE Transactions on Communications, 104(12),* 1455–1464.
22. Zhang, L., Zhou, R., Liu, Q., Xu, J., Liu, C. (2021). Transaction confirmation time estimation in the bitcoin blockchain. *Proceedings of the 22nd International Conference on Web Information Systems Engineering*, pp. 30–45. Melbourne, Australia.
23. Zhao, W., Jin, S., Yue, W. (2019). Analysis of the average confirmation time of transactions in a blockchain system. *Proceedings of the Queueing Theory and Network Applications: 14th International Conference*, pp. 379–388. Ghent, Belgium.
24. Kawase, Y., Kasahara, S. (2017). Transaction-confirmation time for bitcoin: A queueing analytical approach to blockchain mechanism. *Proceedings of the Queueing Theory and Network Applications: 12th International Conference*, pp. 75–88. Qinhuangdao, China.
25. Pouraghily, A., Wolf, T. (2019). A lightweight payment verification protocol for blockchain transactions on IoT devices. *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, pp. 617–623. Honolulu, USA.
26. Liu, Y., Zhang, C., Yan, Y., Zhou, X., Tian, Z. et al. (2022). A semi-centralized trust management model based on blockchain for data exchange in IoT system. *IEEE Transactions on Services Computing, 16(2),* 858–871.
27. Ren, Y., Zhu, F., Wang, J., Sharma, P. K., Ghosh, U. (2021). Novel vote scheme for decision-making feedback based on blockchain in Internet of Vehicles. *IEEE Transactions on Intelligent Transportation Systems, 23(2),* 1639–1648.

28. Ren, L., Ward, P. A., Wong, B. (2021). Improving the performance of blockchain sharding protocols with collaborative transaction verification. *Proceedings of the 2021 IEEE International Conference on Blockchain*, pp. 462–469. Melbourne, Australia.
29. Zhu, Y., Guo, R., Gan, G., Tsai, W. T. (2016). Interactive incontestable signature for transactions confirmation in bitcoin blockchain. *Proceedings of the 40th IEEE Annual Computer Software and Applications Conference (COMPSAC)*, pp. 443–448. Atlanta, USA.
30. Halunen, K., Vallivaara, V., Karinsalo, A. (2018). On the similarities between blockchains and merkle-damgård hash functions. *Proceedings of the 2018 IEEE International Conference on Software Quality, Reliability and Security Companion*, pp. 129–134. Lisbon, Portugal.
31. Liu, Y., Yu, W., Ai, Z., Xu, G., Zhao, L. et al. (2022). A Blockchain-empowered federated learning in healthcare-based cyber physical systems. *IEEE Transactions on Network Science and Engineering, 10(5),* 2685–2696.
32. Ren, Y., Leng, Y., Qi, J., Sharma, P. K., Wang, J. et al. (2021). Multiple cloud storage mechanism based on blockchain in smart homes. *Future Generation Computer Systems, 115,* 304–313.
33. Tennenhouse, I., Raviv, N. (2023). Transaction confirmation in coded blockchain. https://doi.org/10.48550/arXiv.2305.05977
34. Liu, C., Li, K., Li, K., Buyya, R. (2017). A new service mechanism for profit optimizations of a cloud provider and its users. *IEEE Transactions on Cloud Computing, 9(1),* 14–26.
35. Zhu, H., Guo, Y., Zhang, L. (2021). An improved convolution Merkle tree-based blockchain electronic medical record secure storage scheme. *Journal of Information Security and Applications, 61,* 102952.
36. Yu, M., Sahraei, S., Li, S., Avestimehr, S., Kannan, S. et al. (2020). Coded merkle tree: Solving data availability attacks in blockchains. *Proceedings of the 24th International Conference on Financial Cryptography and Data Security*, pp. 114–134. Kota Kinabalu, Malaysia.
37. Castellon, C., Roy, S., Kreidl, P., Dutta, A., Bölöni, L. (2021). Energy efficient merkle trees for blockchains. *Proceedings of the 20th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 1093–1099. Shenyang, China.
38. Wang, F., Zhou, J. T., Wang, H., Guo, X. (2022). A blockchain-based multi-cloud storage data consistency verification scheme. *2022 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking*, pp. 371–377. Melbourne, Australia.
39. Chen, Y., Luo, L., Ren, B., Guo, D. (2022). Geo-distributed IoT data analytics with deadline constraints across network edge. *IEEE Internet of Things Journal, 9(22),* 22914–22929.
40. Ma, Y., Guo, Z., Wang, L., Zhang, J. (2020). Probabilistic life prediction for reinforced concrete structures subjected to seasonal corrosion-fatigue damage. *Journal of Structural Engineering, 146(7),* 04020117.
41. Chen, Y., Luo, L., Guo, D., Rottenstreich, O., Wu, J. (2021). SDTP: Accelerating wide-area data analytics with simultaneous data transfer and processing. *IEEE Transactions on Cloud Computing, 11,* 911–926.