IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 September 2023, accepted 11 October 2023, date of publication 18 October 2023, date of current version 27 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3325748

RESEARCH ARTICLE

Multi-State Merkle Patricia Trie (MSMPT): High-Performance Data Structures for Multi-Query Processing Based on Lightweight Blockchain

VIDDI MARDIANSYAH, (Member, IEEE), ABDUL MUIS, (Member, IEEE), AND RIRI FITRI SARI, (Senior Member, IEEE)
Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia, Depok City 16424, Indonesia

Corresponding author: Riri Fitri Sari (riri@ui.ac.id)

This work was supported in part by the Ministry of Education and Culture, Indonesia, through the Directorate General of Research and Development under Penelitian Disertasi Doktor (PDD) Research Scheme under Grant NKB-336/UN2.RST/HKP.05.00/2021. The work of Viddi Mardiansyah was supported in part by Lembaga Pengelola Dana Pendidikan (LPDP), Ministry of Finance, Indonesia, under Contract 2020032102333.

ABSTRACT Blockchain technology has emerged as a promising solution to secure and decentralized platforms. However, blockchain technology has high computational requirements, latency, and low throughput, particularly for single or multi-query processing. Lightweight blockchain has emerged as a solution to overcome these problems. It addresses performance and efficiency issues and can provide convenience in the query process. This paper proposed a novel high-performance data structure for multi-query processing based on a lightweight blockchain, namely Multi-State Merkle Patricia Trie (MSMPT). MSMPT combines Merkle Patricia Trie (MPT) based indexing and linked-list storage to achieve high performance. MPT has been used on the Ethereum network with a Key-Value database approach. The key field in this proposal is used as crucial user data. The value field is changed to the head of the linked list, and the following data elements will store a summary of the data based on the specified category. In this paper, a blockchain simulator was built to discover the performance of the proposed systems. This simulator will simulate creating blocks in a blockchain network using existing and modified blockchain data structures. The blocks created will be compared using the query process from the conventional and proposed systems. The experimental findings demonstrate that MSMPT outperforms existing blockchain-based data structures by requiring only about one millisecond in query processing performance and less than 500 bytes of additional storage. The MSMPT provides a promising solution for efficient and scalable data management in lightweight blockchain, particularly for multi-query processing.

INDEX TERMS Blockchains, blockchain data structure, lightweight blockchain, linked lists, multi-state Merkle Patricia Trie.

I. INTRODUCTION

Evolution in the digital world has made data an indispensable commodity for all activities [1]. Data is used as a primary source in various application areas such as healthcare [2], [3], [4], transportation [5], [6], economy [7], [8] and industry [9], [10]. Because this data is considered valuable, it creates an interest for criminals to exploit it for personal acquisition or interest [11]. Destruction or minimal alteration of data to not arouse suspicion from data owners is one of the most feared things [12]. Another area for improvement with data is the availability of the data itself [13]. Data is available from only

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

one source, so in the event of a denial-of-service attack on the data provided, the availability of data will be detrimental to the owners and users of the data.

Data has become the most valuable asset for organizations of all sizes and types in the digital era. From financial transactions to medical records and personal information, data is being generated and collected at an unprecedented scale. However, this massive amount of data also challenges traditional storage systems and data management. Blockchain technology has emerged as a potential solution to these problems. The emergence of blockchain technology as a potential remedy to these problems.

Since being introduced by Satoshi Nakamoto in 2008 and implemented in 2009 as Bitcoin's underlying technology [14], blockchain in recent years has become popular because of the increasing success of some cryptocurrencies. According to Forbes, the two most popular cryptocurrencies in May 2022 [15] are Bitcoin and Ethereum [16]. Blockchain technology also offers a solution to these two problems of data security. First, it is an immutable and tamper-resistant data structure on blockchain technology replicated among nodes in a network. This append-only system made the data cannot be changed. Second, it is a distributed database characterized by decentralization [17], [18]. This distributed system resolves data availability restrictions because data can be accessed from various locations. Because this data is considered valuable, it creates an interest for criminals to exploit it for personal acquisition or interest. In the last several years, blockchain technology has been used in many industries, including healthcare, public critical infrastructures, supply chain and logistics management, access control, and medical records. Blockchain was originally designed for recording financial transactions or asset transfers. Thus, blockchain is currently used to store valuable data, including ownership or identity certificates, financial accounts, medical data, and even data transactions from the Internet of Things (IoT).

The following disadvantages of the conventional blockchain, which is based on the Bitcoin system: First off, the transaction is open and entirely transparent, and the transaction latency is significant [14] because of the shortcomings of the consensus process. Decentralized design is the second disadvantage. All existing nodes can join the blockchain or leave.

However, the blockchain is a chain structure database, which will cause the query inefficiently regarding the increasing number of blocks grown. As of March 31, 2023, the block height of Bitcoin reached 783,277. It means that when the blockchain systems need to query historical data, even for a single data, systems will need to traverse hundreds of thousands of blocks. The Bitcoin blockchain does not hold account balances. It only holds keys to Unspent Transaction Outputs (UTXOs). An entire UTXO is spent (sometimes partially received back as ''change'' as a brand new UTXO) and is included in a transaction.

Ethereum has proposed an account-based data architecture, which can manage transactions better to increase the overall effectiveness and accessibility of blockchain. To accelerate the query, Ethereum created three index trees [19], [20].

The world state in Ethereum updates the status of all deployed and active smart contracts and all accounts, including balances. The whole node's state tree comprises an RLP-encoded global account [21], [22]. It handles account data, creates transaction set hashes using the Merkle Patricia Trie, and organizes and manages account data [23].

Ethereum used LevelDB [24] as the world state database in the blockchain system and stored the block data in simple Key-Value (KV) queries, not relational queries. LevelDB is an open-source on-disk key-value store used in various operating systems like Windows, macOS, Unix-based systems, and Android. LevelDB enables database to record the keys and the values in arbitrary byte arrays and organizes data according to the key. This database supports data compression, grouping writes, and forward and backward iteration.

Below is a summary of this article's contributions.

- This paper proposed a high-performance data structure for a multi-query processing model based on a lightweight blockchain, which improves the multi-query search method on the lightweight blockchain.
- A Multi-State Merkle Patricia Trie (MSMPT) is proposed based on a modified Merkle Patricia Trie with a Linked List. Each linked list associated with a critical client account may have different state information from other key client accounts.
- MSMPT improves query search from state storage. This structure can improve the performance of blockchain information retrieval while ensuring that each account's different states remain known.

The remaining sections are organized as follows. Section II discusses related concepts and works, while Section III provides an overview of the proposed methodology. The details about multi-state queries are presented in Section IV. The experiment results and discussion are covered in Section V, while Section VI provides a summary of the work.

II. PRELIMINARY CONCEPTS

Since its beginnings in 2009 as the technology underlying Bitcoin transactions, blockchain is a distributed ledger system that has been through substantial development. With the capability to provide security, immutability, accountability, and transparency through a distributed network, it is highly suited for use cases that conventional infrastructure cannot serve. This section will briefly describe specific approaches linked to the proposed system and explain why it is essential to the architecture of the proposed system to comprehend the blockchain infrastructure.

A. BLOCKCHAIN ARCHITECTURE

Blockchain architecture describes the layout and organization of a blockchain network, a distributed, decentralized database used to store data and track transactions. A combination of different technologies, such as cryptography,

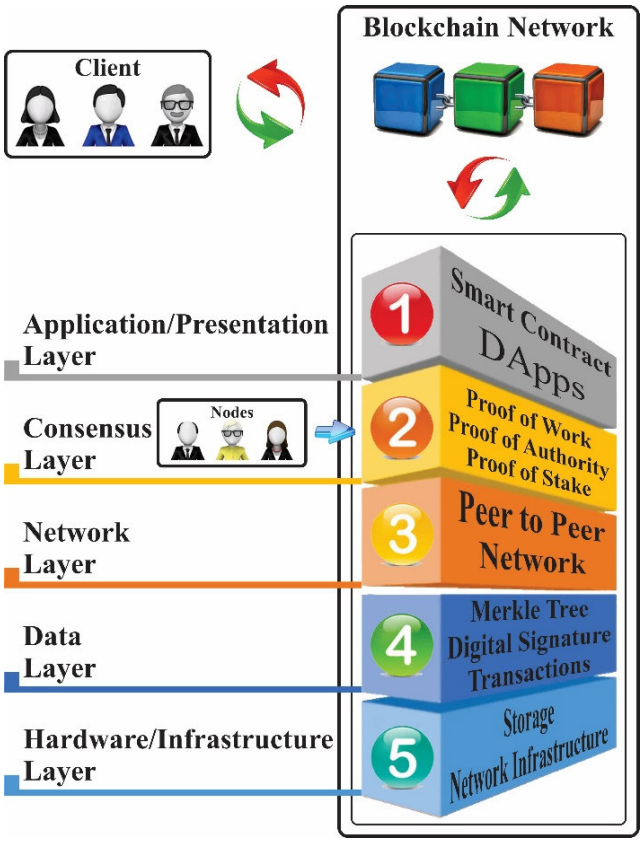

FIGURE 1. Blockchain architecture layer.

distributed ledger technology, networking, and game theory, works together to enable trust and consensus on the chain [25]. The architecture preserves the same structure across all nodes and is optimized as a monetary system. A public blockchain system is neutral because it does not depend on trust, unlike traditional ledgers managed by centralized entities [26]. The blockchain architecture is composed of different layers, each of which serves a specific purpose in the operation of the blockchain network. A blockchain architecture layer is presented in Fig. 1, and this structure is also used to describe different layers and components.

The application/presentation layer, shown in Fig. 1, consists of the programs end users use to communicate with the blockchain network. The application and execution layers comprise this stratum. Layer of application may consist of user interfaces, software, application programming interfaces (APIs), and frameworks. At the execution layer, smart contracts and decentralized applications (DApps) are examples of this layer. Despite the fact that transactions pass from the application to the execution layer, they are validated and executed at the semantic layer. The application provides instructions to the execution layer, which implements transactions and guarantees the blockchain's deterministic nature.

The second layer, or consensus layer, shown in Fig. 1, is an essential part of the layer in the existence of the blockchain platform. This layer creates a new block, validates the newly created block, and ensures that all nodes involved in the blockchain network approve or disapprove of the activities that occur. Several existing cryptocurrencies have consensuses, such as Bitcoin and Litecoin, using a Proof-of-Work (PoW) consensus, while Ethereum uses a Proof-of-Stake (PoS) consensus.

In Fig. 1, the communication between nodes in the third or network layer uses a client-server topology. However, in blockchain technology, the topology is known as peer-to-peer to interact between nodes in the network. Block discovery, transactions, and propagation are all handled by this layer. The propagation layer is another name for this layer. This layer ensures that nodes can find each other and interact, propagate, and synchronize to maintain the blockchain network in a legal state. Nodes perform activities and transactions that occur on the blockchain.

The fourth layer, or the data layer in Fig. 1, is a data structure used by the blockchain and is illustrated by a series of interrelated blocks. All data or transactions recorded in a block are stored in a Merkle Tree (MT). MT is a binary tree of hashes. Each successfully mined block contains the root hash of the MT and other information, such as block version number, the previous hash block, nonce, and time-stamp. MT provides security and integrity because all transactions in a block are recorded in a root hash. So it is challenging to change the existing transaction data because the slightest change will affect the owned root hash value.

The last layer in Fig. 1, the infrastructure or hardware layer in the blockchain architecture, is a data storage medium known as a server. In the blockchain mechanism, all data is stored not in a server but in several nodes in a distributed network. When clients request content or data stored in a node, applications search for data using a client-server mechanism. This data search mechanism on the blockchain is conducted on the nearest or trusted node through a peer-to-peer network mechanism. The client-to-client or peer-to-peer blockchain network nodes are also tasked with computing, validating, and recording transactions in a shared ledger.

B. BLOCKCHAIN STORAGE

Blockchain storage is an innovative method of securely storing data in a decentralized network. Traditional centralized cloud storage platforms have loopholes that can compromise data security. Blockchain storage can solve the shortcomings inherent in traditional centralized storage infrastructures. To manage the storage in blockchain, usually, the data are stored in a data structure that allows for the integrity of the data to be verified (e.g., Merkle Tree [27] or Merkle Patricia Trie [23]). The block is kept on a storage device in a log-like structure, with the information stored in a state database for quick access. In the event of a failure, this log is mostly used to verify or rebuild the state database [28].

As a decentralized technology, blockchain features stores facts and statistics. A series of blocks hold the entire transaction report [29]. As illustrated in Fig. 2, every block in the chain contains transactions, block version, time-stamp, previous hash, nonce, and Merkle Tree.

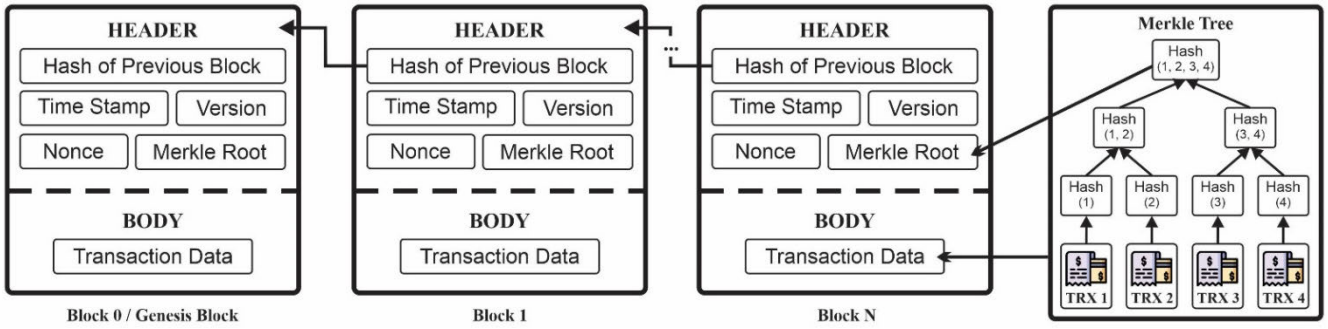

FIGURE 2. Blockchain data structures in general.

A block is the fundamental unit of the blockchain and consists of numerous interconnected pieces with substantial value. The block contains the previous hash block, time-stamp, block version, nonce, Merkle root, and list of transactions (which comprise a transaction and the transactions counter). The previous hash block and Merkle root hash tree in a block primarily use encryption hashing algorithms to safeguard data and transactions.

Cryptocurrencies use specific hashing algorithms that implement cryptographic hash functions and maintain blockchain functionality and transaction processing. Bitcoin, for example, employs the SHA256 algorithm, Ethereum employs the Ethash method, and Litecoin employs the Scrypt (pronounced es-crypt) algorithm. SHA-256 is more complex, secure, and consumes more energy, making it suitable for large-scale mining operations. Scrypt is faster, simpler, and less energy-consuming, making it more accessible to individual miners. Ethash is a memory-hard algorithm specifically designed for Ethereum, making it more resistant to ASICs and fostering decentralization.

The Merkle root tree hash logs all transactions. Every new transaction made by a user on the blockchain network will be stored on the Merkle Tree. The Merkle root hash created from all transactions at that moment will be saved and formed into a block. When a new transaction attempts to change the contents of a block, all fields must be updated to reflect the changes. Digital signatures mostly rely on the Merkle root tree hashes to prevent malevolent users from purposefully changing data. Digital signatures employ a variety of encryption algorithms. These digital signatures complicate the conversion process for hackers; to escape detection, they must destroy the block containing the record and those related to it.

Blockchain is a network of interconnected blocks by cryptographic signatures, with each block containing data. The genesis block is the initial block in a blockchain network. The new block created will include the header data from the previous block and all the transaction data itself. After this new block is connected to the previous block and informed of all connected nodes in the blockchain network and as confirmation of approval for adding this block from other nodes, the block can no longer be changed or tampered with. If new data needs to be stored using the append-only mechanism, a new blockchain block must be introduced to the network [30].

C. BLOCKCHAIN QUERY

Blockchain query refers to retrieving data from a blockchain. There are several ways to query blockchain data, depending on the blockchain platform, the type of data, and the use case. Blockchain query is essential to build decentralized applications, analyze blockchain data, and explore blockchain transactions.

A query on the blockchain is an operation used to get information from a transaction or block on the blockchain network. Queries on the blockchain usually refer to requests for specific information sent to the blockchain network. This operation can be performed by anyone connected to the blockchain network, provided they have access to the required private key.

In the blockchain, two types of queries are commonly performed. The first is a single query, an operation performed to get information from a single block or transaction on the blockchain. An example of a single query is getting information about an account's balance or checking the transaction's confirmation status. The second query is a multi-query, an operation performed to simultaneously get several blocks or transactions on the blockchain. An example of a multi-query is getting information about several transactions that occurred in a certain period or checking information about several addresses simultaneously.

In processing queries on the blockchain, several protocols, such as consensus and encryption protocols, are used to ensure data security and validity. This query process ensures that the information the blockchain provides is reliable and valid.

The blockchain systems query process has at least four layers: storage, transaction, consensus, and network. Every layer is entirely adjustable for the specific demands of various technological variants, with all their benefits and drawbacks [31]. Fig. 3 shows the simplified architecture of a blockchain for querying data using four layers.

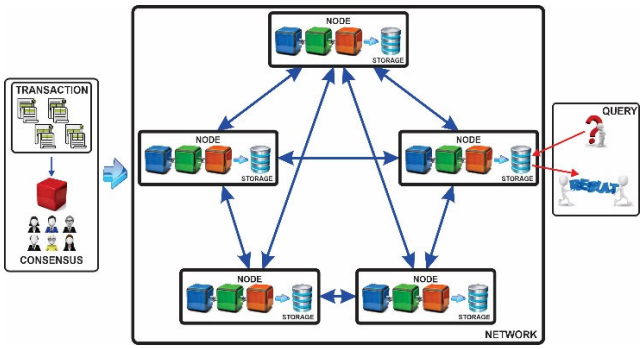

FIGURE 3. Simplified architecture of a blockchain for querying data using four layers.

In Fig. 3, when a transaction occurs, and a block with a certain consensus is generated, the newly created block is linked to the existing block and stored in distributed storage. When a query transaction is requested by a user/node, the system will read all data stored in storage to respond to the query request. Searching for this data will depend on the size of the data saved in storage. The bigger the data stored, the query process will be very long. The more data is saved, the longer the query process. Because the system is decentralized, data can be searched by reading the storage closest to the user who requested the query.

Challenges with querying blockchain data include limited APIs, decentralization, lack of query language, and data confusion and entanglement [32]. The more blockchain-based applications, the more data need to process and store. As a result, eventually, there will be a problem with enough data flowing and bandwidth increasing.

D. LIGHTWEIGHT BLOCKCHAIN

As a new technology approach, a lightweight blockchain is a simplified or modified version of a traditional blockchain that enables trustless, secure data transactions between nodes. The architecture of blockchain technology demands heavy computation and energy consumption, making it difficult for resource-constrained devices to participate in the network. Lightweight blockchain is simplified and designed to be less resource-intensive than traditional blockchain, but data security is not compromised. This strategy is suitable for applications and devices, including Internet of Things devices, that require data integrity but have limited computational resources [33]. An experiment demonstrated, for instance, that a lightweight blockchain-based network could accommodate up to 1.34 million authentication and verification processes per second, which is sufficient for use in IoT networks with limited resources, especially in the healthcare industry [34]. To better understand the research opportunities and restrictions, the characteristics of current research and the limitations of blockchain for IoT on lightweight blockchain for IoT proposals must be rigorously studied [35].

A lightweight distributed blockchain comprises several modules, including a consensus process, a ledger synchronization protocol, and shared storage persistence. The distributed blockchain module makes memory restrictions on computing nodes easier [36]. However, using a distributed blockchain module may increase consensus's complexity and communication costs. Therefore, selecting a consensus mechanism and query process is crucial in designing a lightweight blockchain.

E. RELATED WORKS

A query process can be an exponential mechanism [37]. The exponential mechanism defines a probability distribution over all outcomes and generates a random one from the distribution. Many researchers have conducted appropriate research in these fields to understand better how to increase the efficiency of the blockchain query process.

EtherQL was initially developed by Li et al. [38] as an effective querying layer for Ethereum. Range queries and top-k queries, two highly effective query primitives that may be quickly integrated with other applications, are provided by this technology for analyzing blockchain data. Different levels of abstraction are provided by EtherQL, making it suitable for application developers, researchers, and data analysts.

Yue et al. [39] demonstrate their experience using SQL-based databases for blockchain technology analysis, detail the features of the Bitcoin blockchain that make it a compelling database case, compare and contrast the three approaches, and offer recommendations for practical applications. They discovered that querying blockchains with SQL can be a valuable instructional strategy.

Riadi et al. [40] proposed a system framework design using the Inter-Planetary File System (IPFS) and Blockchain technology with a case study of COVID-19 data. Combining these technologies allows the data stored in the Electronic Health Record to be safe and always available. All data is secured with blockchain cryptographic algorithms and can only be accessed using their user private key. This system can maximize the use of private keys as access rights to maintain the integrity of the COVID-19 diagnosis and certificate data.

Jiahe et al. [41] suggested a blockchain-based intelligent manufacturing security model and implemented a novel Merkle Patricia Trie to enhance the blockchain's structure and enable quick node status queries. Because the Merkle Patricia Trie degrades the performance when supporting the data operation with high data volume and concurrent operation. Jiahe et al. propose a cache-based Merkle Patricia Trie and lock-free concurrent that can reduce the time complexity of data injection and improve the acceleration of block generation and query process.

Yang et al. [42] proposed LedgerDB, a centralized ledger database designed for secure and tamper-proof recording of application transactions. LedgerDB provides universal auditing and verification capabilities for all parties requiring transaction recording integration. This LedgerDB is proposed to be deployed on Alibaba Cloud. Another Yang [43] paper tells about the concepts and principles of verification in LedgerDB, which emphasizes the importance of external solid audit capabilities in centralized ledger databases.

Kong et al. [44] proposed a new world state by enhancing the entire node organization mode and the balancing state. State creation's organization and insertion techniques are redefined as a state tree is built for some states in blocks. Their research shows that the suggested strategy lowers memory usage and boosts retrieval effectiveness.

Xing et al. [45] discovered that the current blockchain system only supports traversal queries using transaction hashes as keywords and needs to perform better in data management. The account's past transactions can now be accessed more quickly thanks to a query mechanism based on the account transaction trace chain (ATTC). They suggest an account transaction chain-index structure based on sub-chains. The account transaction chain is broken up into smaller chains, and a hash pointer links the final blocks of each chain. The sub-chain query mode in ATTC shortens the query path by converting the block-by-block query mode to the sub-chain. In order to reduce the cost of building the index and use fewer storage resources, several transactions from the same account are combined and stored in the same block.

III. HIGH-PERFORMANCE DATA STRUCTURES FOR MULTI-QUERY PROCESSING BASED ON LIGHTWEIGHT BLOCKCHAIN

Query processing in the lightweight blockchain is a vital system performance aspect. A lightweight blockchain is a type of blockchain that tries to lower the storage and processing needs of the blockchain system while preserving its security and decentralization. Lightweight blockchain also uses a combination of cryptographic techniques and optimization strategies. In a lightweight blockchain, query processing is conducted by the network nodes responsible for verifying transactions and maintaining the blockchain [46].

The information saved in a block is merely the Merkle root information, as explained in Fig. 2, and the transaction data that takes place will be stored in a Merkle Tree. Each non-leaf node in a Merkle Tree is identified with the hash value of its offspring node labels, while every leaf node is identified with the hash value of a data block. The Merkle Tree considerably lowers the amount of memory needed for verification and aids in confirming the accuracy and authenticity of data.

When a transaction in the Merkle Tree is tampered with unauthorized, the miners can detect it because the transactions stored in the Merkle Tree store the hash value of each node in the position of the parent node above. So every time there is a change in transaction details, such as the amount to be debited or credited or the address where the payment is made, the change will be forwarded up to the hash at the top level and finally to the Merkle root. The Merkle Tree is the best solution regarding time complexity for searching for a transaction on a block. Merkle Tree is designed for single queries only, for example, for finding a transaction history. Unfortunately, currently, the requirement to query process not only for a single transaction but also a query process for a multi-transaction that has been made for a single user, for example [47]. Thus, the current system needs a new

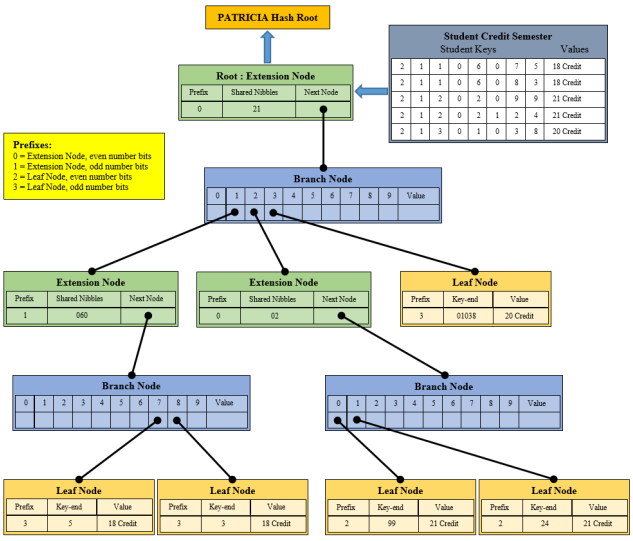

FIGURE 4. Merkle Patricia Trie structure.

or modified blockchain data structure to process multiple queries.

Ethereum uses a Merkle Patricia Trie as its primary data structure for storing the world state (stores the account states, including account balances and nonce values. Each node in this tree represents an account.), transactions (records transactions that update the world state tree. These transactions are stored immutably in the blockchain), and other related data [48]. This specific data structure combines a Merkle Tree and a Patricia Trie, offering benefits of both structures, such as efficient data storage and quick data retrieval.

A data structure known as the Merkle Patricia Trie offers a mechanism to authenticate all (key, value) bindings cryptographically. As a result of its complete determinism, trees with matching (key, value) connections are sure to be identical. These matching connections make inserts, lookups, and deletes efficient, with a time complexity of O(log(n)). Fig. 4 depicts an illustration of the structure of Merkle Patricia Trie employed by the Ethereum network.

Fig. 4 illustrates the three different nodes. The extension node comes first and holds a link to the subsequent node and a portion of the key field shared by several nodes. The branch node is a node that has pointers to separate nodes that share the same key prefix. The last is a leaf node, which contains a value field and the final portion of the key field (also known as the key-end). A prefix from each of the leaf node's parents is concatenated to form the key field to the leaf node, which ends in the leaf node. As mentioned above, Ethereum uses Merkle Patricia Trie on these three trees: the state root, transaction root, and receipts root. Fig. 5 shows the original Ethereum block header.

This paper proposes a novel structure of blockchain by modifying the Merkle Patricia Trie structure used on the Ethereum network and combining it with the Linked List structure to be implemented on a lightweight blockchain network. The approach is to use a Merkle Patricia Trie

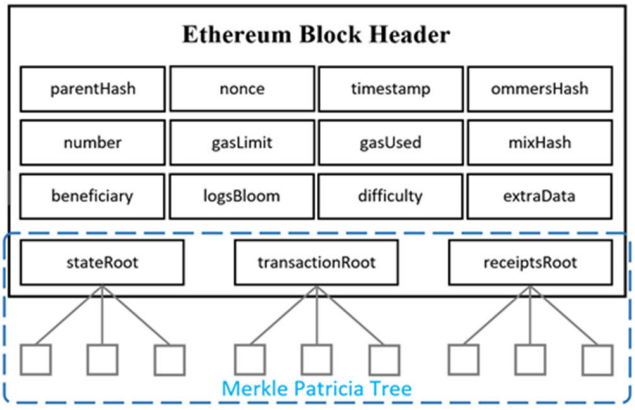

FIGURE 5. Ethereum block header.

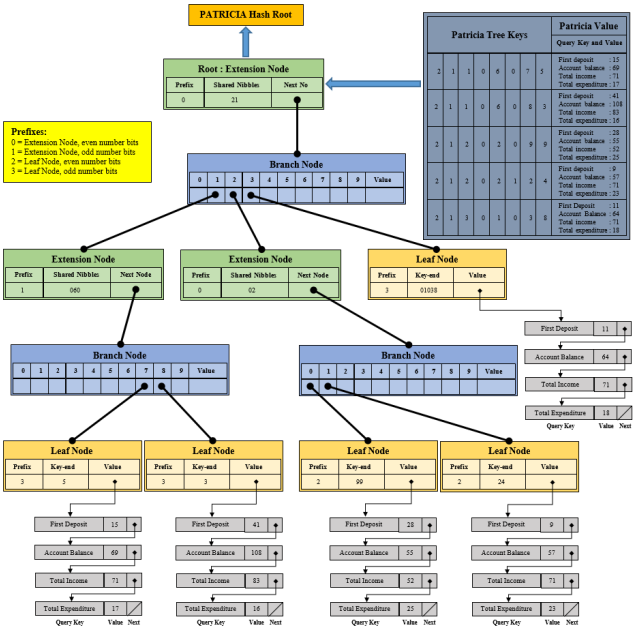

FIGURE 6. Merkle Patricia Trie with Linked Lists structure.

combined with a linked list data structure. The Merkle Patricia Trie in this proposed system uses the same LevelDB (key, value) database; however, the ''value'' field is used as the head value of the proposed linked lists. The next element in the linked lists is a detailed transaction or summary transaction. Therefore, it is expected to have more efficient query processes and data storage than the three Merkle Patricia Tries used in Ethereum. This paper proposed a novel high-performance data structure for multi-query processing based on a lightweight blockchain, the Multi-State Merkle Patricia Trie (MSMPT).

The proposed form of data structure in MSMPT, implemented on a lightweight blockchain by modifying Merkle Patricia Trie structure and combined with Linked List, is depicted in Fig. 6 and Fig. 7. The grey color in Fig. 6 is the additional linked lists structure that combined in Merkle Patricia Trie.

In Fig. 6, the existing Merkle Patricia Trie structure is modified by combining it with the structure of the linked list.

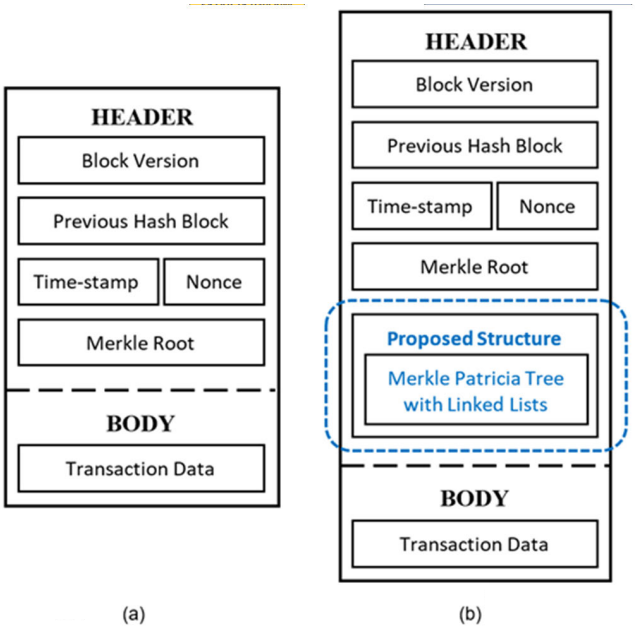

FIGURE 7. Proposed additional Merkle Patricia Trie with Linked Lists into lightweight blockchain structure. (a) blockchain structure, (b) modified blockchain structure.

The unique keys stored in the root, branch, extension, and leaf nodes are still used. However, the value that accompanies the unique Key is changed to a pointer that points to a linked list. Fig. 6 shows that one unique Key will have a pointer pointing to a series of linked lists. Each element in the linked list will store different information according to the system's needs. When a user makes a new transaction, the data stored in the user's element-linked lists will also be updated so that the data stored in the Merkle Patricia Trie with Linked Lists are always the latest.

Fig. 7 shows the modifications made to the blockchain data structure in this paper. A detailed explanation can be seen in Fig. 7 (a), which shows the existing data structure on the blockchain. Fig. 7 (b) shows the proposed Merkle Patricia Trie data structure that has been combined with Linked Lists and will be implemented on a lightweight blockchain.

IV. MULTI-STATE MERKLE PATRICIA TRIE WITH LINKED LISTS

This section will explain the detail of the proposed Merkle Patricia Trie with Linked Lists data structure created to perform multi-query processes. With the multi-query process conducted. It is expected to show multi-states for each user in the lightweight blockchain network.

In order to get good test results, we built a lightweight blockchain simulator ourselves. The block creation process in the simulator uses the consensus Proof-of-Work (PoW) approach. The PoW consensus refers to a lightweight blockchain using a difficulty level that can be set or adjusted so block creation does not take long [49]. The PoW consensus approach was chosen in this simulator because this consensus is the most popular consensus mechanism used in

Algorithm 1 Creates Blocks With Adjustable Difficulty Levels

Input: Block(B), difficulty
Output: B.compute_hashing and nonce
01: **Define Class** Block:
02: **Define Function** __initialize(Me, index, transactions, time-stamp, previous_hash, merkleRoot, MPTwLLroot, nonce = 0):
03: Me.index ← index
04: Me.transactions ← transaction
05: Me.time-stamp ← time-stamp
06: Me.previous_hash ← previous_hash
07: Me.merkleRoot ← merkleRoot
08: Me.MPTwLLroot ← MPTwLLroot
09: Me.nonce ← nonce
10: **Define Function** compute_hashing(me):
11: block_str ← json.dump(Me.__dict__, sort_key=True)
12: **Return** sha256 (block_str.encode()).hexdigest()
13: **Define Function** proof_of_work(Me, block):
14: nonce← Rnd()
15: compute_hashing ← block.compute_hashing()
16: **While Not** compute_hashing.startswith('0' * difficulty):
17: nonce ← Rnd()
18: compute_hashing ← block.compute_hashing()
19: **Return** compute_hashing
20: **Define Function** add_block(Me, block):
21: block.previous_hash ← block.parentcompute_hashing()
22: block.hash ← Me.proof_of_work(block)

cryptocurrencies, such as Bitcoin, to validate transactions and prevent double-spending. It allows decentralized networks to operate securely, with no centralized authority.

Regarding security, PoW ensures that only legitimate transactions are recorded on the blockchain. It requires miners to solve complex mathematical puzzles, making it difficult for bad actors to tamper with the blockchain. The time, energy, and cost of attempting to alter transactions make it highly unlikely for scammers or hackers to succeed. The major drawback of this consensus is energy consumption, which is why this paper uses the adjustable difficulty level in the simulator.

In the blockchain simulator, several default values must be set before a block can be created, including the maximum number of current users, the total number of those who will make transactions, the number of transactions made for each user, the overall value of activities stored per block, the number of miners that are responsible for creating the block, and the level of difficulty determined when creating the block.

Algorithm 1 shows block generation in the blockchain simulator. Fig. 8 explains the detail of the block creation process using the PoW consensus approach, with the target hash being searched for using an adjustable difficulty level using a sequence diagram. Algorithm 2 shows the creation of the Merkle Tree for all the transactions made by the users/nodes.

In Algorithm 1, the block creation process in the blockchain simulator uses the PoW consensus approach. The simulator user determines the difficulty level used in this algorithm. The difficulty level value entered determines the number of consecutive zeroes that appear sequentially in the

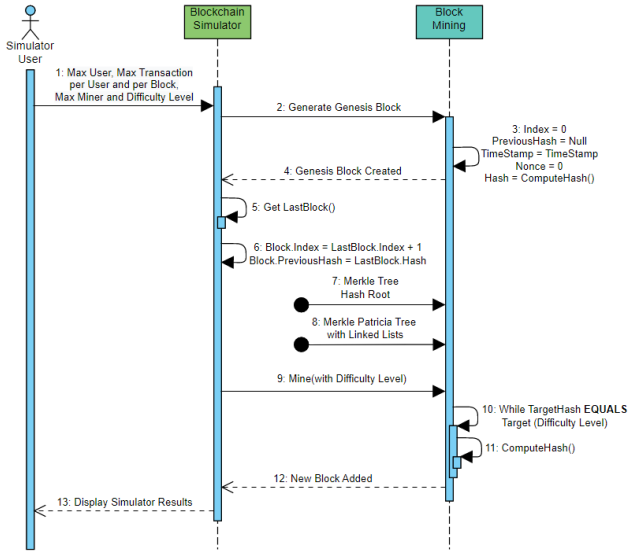

FIGURE 8. Detailed sequence diagram of mining block using new data structure Merkle Patricia Trie with linked lists.

target hash being searched. The greater the specified difficulty value, the more zero values appear sequentially. The mining process for a block will take longer because the possibility of getting several zero values that appear sequentially is less likely. The nonce value is unique as a critical value used to get the hash of the target and can verify the block validity. The nonce value is got randomly and iteratively until a target hash value is obtained with the number of zeros appearing sequentially, and a new block is created.

Fig. 8 shows the block creation process using the proposed new data structure Merkle Patricia Trie with Linked Lists. When the user makes n-blocks, the user must enter several setting parameters expected to produce the desired number of blocks in the simulator application. Then, the blockchain simulator will create an initial block, known as a genesis block, and store it as the first block with a zero index and a zero nonce value because this block only had a hash value without difficulty level. The simulator will create the next block regarding the previous block's hash value (in this case, the Genesis block). When another new block is created, the previous hash value in the new block is referred from the hash value of the previous hash block. When the mining process of a block is started, the simulator will require two additional input values. The first one is the root hash value of a Merkle Tree (the mechanism for finding/generating the hash root value of a Merkle Tree can be seen in Fig. 2 and Algorithm 2) based on the maximum number of transactions to be recorded in a block that the user has determined. The second value of the Merkle Patricia Trie with Linked Lists is the root hash value. The process of the Merkle Patricia Trie with Linked List to obtain the hash root value will be explained using the sequence diagram in Fig. 9.

Merkle Trees are used in blockchain technology for several reasons, contributing to the system's efficiency, security, and scalability. First is data integrity validation. Merkle Trees

Algorithm 2 Create a Merkle Tree to Store Transactions in the Form of a Hash tree

```
Input :  MerkleTree(nodes□MaxUserTrx * MaxTrxPerUser),
         List ← MaxTrxPerBlock
Output :  nodes.getRootHash
01: Define Class Node:
02:   Define Function __initialize(Me, left, right, value):
03:     Me.left ← left
04:     Me.right ← right
05:     Me.value ← value
06:     @staticmethod
07:    Define Function doubleHash(value:str):
08:       Return hashlib.sha256(hashlib.sha256(value.encode()).
                digest()).hexdigest()
09:
10:  Define Class MerkleTree:
11:   Define Function __initialize(Me, values: List[str]):
12:     Me.__buildTree(values)
13:   Define Function __buildTree(Me, values: List[str]):
14:      leaves ← [Node(None, Node,doubleHash(e)) For e In
values]
15:     If Len(leaves) % 2 Equals 1:
16:       leaves.append(leaves[-1:]) # duplicate last element IF
                                       odd number of elements
17:     Me.root ← Me.__buildTreeRec(leaves)
18:   Define Function __buildTreeRec(Me, nodes: List[Node]):
19:     half ← Len(nodes) // 2
20:     If Len(nodes) Equals 2:
21:     Return Node(nodes[0], nodes[1], Node.doubleHash
              (nodes[0].value + nodes[1].value))
22:     left ← Me.__buildTreeRec(nodes[:half])
23:     right ← Me.__buildTreeRec(nodes[half:])
24:     value ← Node.doubleHash(left.value + right.value)
25:     Return Node(left, right, value)
26:   Define Function OutputTree(me):
27:     Me.__ OutputTreeRec(Me.root)
28:   Define Function __OutputTreeRec(Me, node):
29:     If node != None:
30:       Display(node.value)
31:       Me.__OutputTreeRec(node.left)
32:       Me.__OutputTreeRec(node.right)
33:   Define Function getRootHash(Me):
34:     Return Me.root.value
```

can effectively validate data integrity in a blockchain using a cryptographic hash method. This validation ensures that any changes in the data can be easily detected and the original data can be verified with no entire dataset. The second is reduced disk space. Compared to other data structures, Merkle Trees take up very little disk space, as they only store the hash values of the data blocks. This hash value allows for more efficient storage of transaction data. The third is efficient verification. Merkle Trees enable faster and more efficient data verification by only requiring a few hash values to prove the integrity of a transaction. This efficiency of data verification reduces the computational resources needed for verification. The last one is the decreased network traffic. By breaking down the data into smaller pieces and using hash values, Merkle Trees reduce the data sent over the network for verification. This capability to reduce the data leads to reduced network traffic and faster transaction times.

Algorithm 2 shows the process of creating the Merkle Tree. Creating a Merkle Tree in this simulator depends on three data variables inputted into the simulator: MaxUserTrx, MaxTrxPerUser, and MaxTrxPerBlock. MaxUserTrx is the data variable used to determine the number of users/nodes that will actively make transactions on the simulator. The data variable MaxTrxPerUser is the data to determine the number of transactions made by the user/node. While the data variable MaxTrxPerBlock is used to determine the number of transactions that will be recorded in the Merkle Tree, which will be hashed root and stored in a block. In Figure 2, the evidence shows that each transaction will be on a leaf on the Merkle Tree. If the number of transactions is odd or Not divisible by two, then the transaction will be duplicated. So that the hash process at the top level will still be able to create a good root hash value.

Fig. 9 shows the sequence diagram of the proposed Merkle Patricia Trie with Linked List as a new structure of lightweight blockchain. Ethereum blockchain uses the Merkle Patricia Trie as one of its crucial data structures in the storage layer. There are four reasons Ethereum uses the Merkle Patricia Trie. First is efficient and secure data verification. Merkle Trees enable efficient data verification in a decentralized blockchain network, ensuring the shared data can be verified and trusted with little processing power. The second is support for key-value maps. Unlike binary Merkle Trees, which are suitable for authenticating information in a "list" format, the Merkle Patricia Trie handles key-value maps, making it suitable for state representation. Third is the immutable state tree. Merkle Patricia Tries are used in Ethereum to create an immutable state tree, which helps maintain the integrity of the state data and allows for the generation of proofs for including transactions or state changes. The last one is Merkle proofs. Merkle Patricia Tries allow for the generation of Merkle proofs, enabling users to authenticate a small amount of data and extend that authentication to large databases of potentially unbounded size. This feature is helpful for light clients that only download the chain of block headers.

Ethereum uses Keccak to create the Merkle root in the Merkle Patricia Trie since it provides a secure and efficient hashing mechanism for a tree structure that is well-suited to represent and update the state of the Ethereum blockchain. Keccak, also called SHA-3, is a brand-new hash algorithm unrelated to the SHA-1 and SHA-2 families. In this proposed method, the simulator still uses the algorithm. The blockchain simulator uses the function Keccak taken from Crypto.Hash library.

Ethereum uses RLP (Recursive Length Prefix) encoding in its Merkle Patricia Tries to provide a space-efficient and standardized method of transferring data between nodes. RLP standardizes the space-efficient format of data transit between nodes. RLP is the primary encoding technique for serializing things in Ethereum and is used to encrypt arrays of binary data that can be nested in any order. The function RLP in the simulator is imported from the PyPI library.

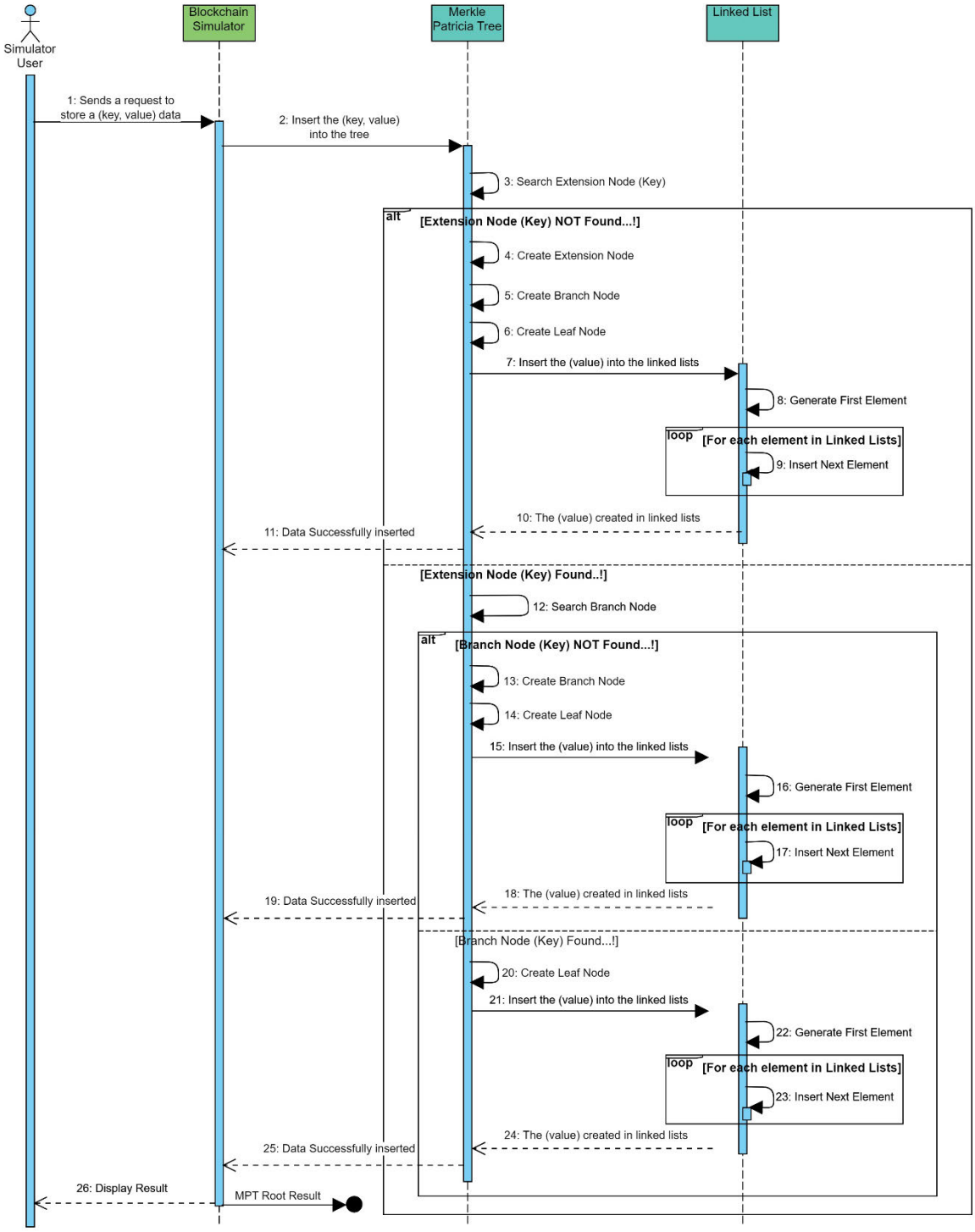

FIGURE 9. Sequence diagram of Merkle Patricia Trie with linked lists.

In Fig. 9, recording data using the Merkle Patricia Trie with Linked Lists are made by finding the key field value of the transaction data. As mentioned, Merkle Patricia Trie uses key and value databases. If the key field value is not found in the data in the Merkle Patricia Trie, a new extension node, branch node, and leaf node will be constructed. The value fields in the Merkle Patricia Trie hereafter will store the information of the pointer of a new linked list of transactions. If the key field value of the transaction data sought is found, adding an extension, branch, or leaf node is not performed. However, it only conducts the updating process of the data stored in the linked list.

The Merkle Patricia Trie with Linked Lists, which has three different nodes—branch nodes, extension nodes, and leaf nodes—develops as shown in Fig. 9. Branch nodes, these nodes represent the internal nodes in the tree and have up to 16 child nodes (since Ethereum addresses are represented as 256-bit numbers, and each branch node represents a 16-way radix tree). Each child node can be another branch node, an extension node, or a leaf node. The hash of a branch node is derived from the hashes of its child nodes. Extension nodes are used to optimize the storage and traversal of the tree by compressing chains of single-child nodes. An extension node contains a shared nibble sequence (a standard portion of the Key between two nodes) and a pointer to the next node (a branch node or a leaf node). The hash of an extension node is derived from the shared nibble sequence and the hash of the next node. Leaf nodes represent the data stored in the tree (e.g., account balances, contract storage, or contract code). A leaf node contains a key and a value. The key field is the Ethereum address (or storage location), and the value field is the RLP-encoded data associated with that address. However, the value in the leaf node is modified to store the pointer head address of the linked list structure to be created for each user/node. Linked list structures are used to store data in leaf nodes, allowing for efficient updates and deletions.

V. EXPERIMENTS

The proposed system has been implemented and tested on a blockchain simulator environment to validate the feasibility of the proposed system. This section presents a detailed evaluation of MSMPT as a high-performance data structure for multi-query processing based on lightweight blockchain. The experimental setup is described first, then the evaluation findings and discussion are presented.

A. EXPERIMENTAL SETUP

The experiment was performed and tested using Python programming language version 3.8.5. On two operating systems: Microsoft Windows 10 (64-bit) Home Edition and virtual machine Ubuntu 18.04.4 LTS version. The hardware configuration used to test the blockchain simulator is i7-8th generation Intel Core Processor, with 2.2GHz six-core processor, supported with 16GB DDR4 memory-card (extended 16 GB), including the NVidia GeForce Video Graphic Cards (series RTX 2070), and SSD 2TB.

In order to achieve a better understanding of performance under different conditions, the blockchain simulator that has been built has parameter settings that are used to produce various outputs that will be tested. Several parameter settings are defined for testing stages to achieve well-measured test results on the blockchain simulator.

Table 1 shows the category of each element on linked lists that used in the simulator. This category can be changed base on the need of the user. In this experiment, the number of category defined for four different categories with difference purpose of category.

TABLE 1. Category that is implemented on each element linked list.

Linked Lists Element	Category Element	Note
First Element	First Deposit	Stored the information of the first deposit of a node/user.
Second Element	Account Balance	Stored the information on the amount of the last balance.
Third Element	Total Expenditure	Stored the information on the amount of the total expenditure.
Fourth Element	Total Revenue	Stored the information on the amount of the total revenue.

In this experiment, the category mentioned in Table 1 is adapted from the common Bitcoin transaction. The category is chosen because of the query transaction in Bitcoin, which is usually lookup for in this category. Maybe for other implementations, the category can be changed based on the system's needs. Such as, in healthcare, the first category can store real-time heartbeats information. The second category may store the information summary heartbeats (diastole or systole) per day, the next category for the information per week or any other category needed, or maybe on any other application systems.

Table 2 shows the parameter settings used for performance testing. The number of blocks generated from the simulator refers to the parameter settings in Table 2. These generated blocks will then be tested for speed in conducting the query process so that for each scenario that has been determined, the performance that wants to know will be known.

Every scenario on the performance evaluation will be tested at least ten times. Performing multiple tests and experiments can lead to a better understanding of the result and functionality of the simulator's output using the proposed system.

B. ANALYSIS OF QUERY PERFORMANCE PROCESSING SPEED

The query performance processing speed of the Merkle Patricia Trie with Linked Lists is analyzed based on lookup operations. This operation is a searching process for a specific key in the tree that involves traversing the tree from the root to the corresponding leaf node.

Based on the parameter setting declared in Table 2, the targeted output of the simulator is to produce ten to 50 blocks, 100 to 500 blocks, and 1000 to 5000 blocks. To determine the results of significant differences from the blocks made, each block in the first scenario contains two transactions. The second scenario contains twenty transactions following the number of target blocks stated above. After the target number of blocks has been created, the lookup process is performed by querying a user account and capturing the time required to search and find detailed information on that account. Experiments conducted for the query lookup process using the Merkle Patricia Trie with Linked List on the blockchain simulator are shown in Fig. 10 and Fig. 11.

TABLE 2. Parameter setting for performance evaluation.

Parameter Setting Name	Scenario 1 Value	Scenario 2 Value	Notes
The total of active nodes/users	10	10	The total of available nodes/users
Transactions/activities per block	2	20	The total of transactions that can be stored in a block
Nodes/users that make transactions	4	4	The total of nodes/users that make transactions

	Value	Target Blocks Created	Value	Target Blocks Created	
The number of transactions per user	3, 30, 300	10, 100, 1000	25, 250, 2500	10, 100, 1000	The number of transactions made by each node/user
	5, 50, 500	20, 200, 2000	50, 500, 5000	20, 200, 2000	
	7, 70, 700	30, 300, 3000	75, 750, 7500	30, 300, 3000	
	10, 100, 1000	40, 400, 4000	100, 1000, 10000	40, 400, 4000	
	13, 130, 1300	50, 500, 5000	125, 1250, 12500	50, 500, 5000	

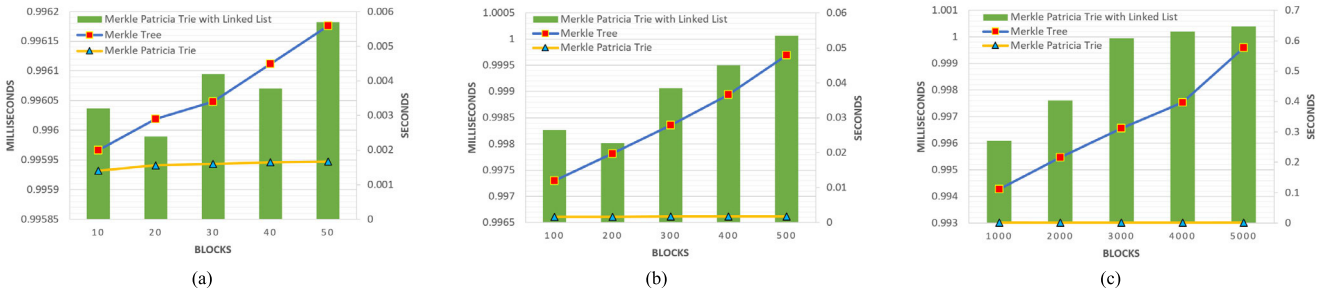

FIGURE 10. Query processing speed comparison with two transactions per block. (a) 10-50 blocks, (b) 100-500 blocks, (c) 1000-5000 blocks.

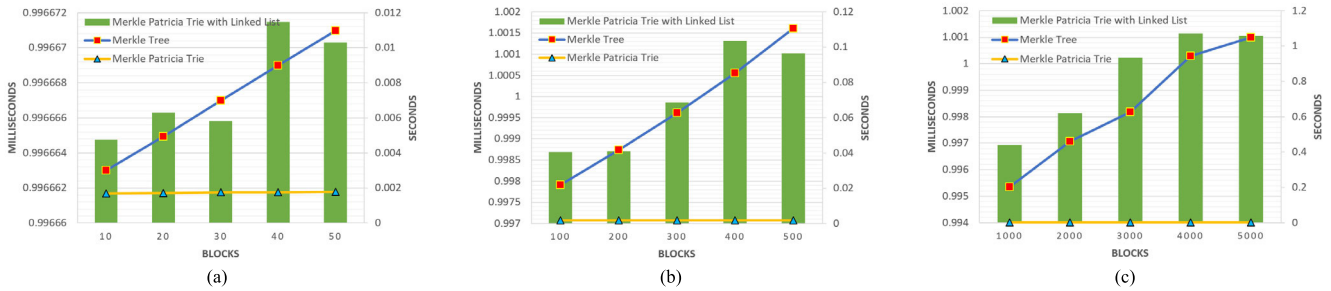

FIGURE 11. Query processing speed comparison with twenty transactions per block. (a) 10-50 blocks, (b) 100-500 blocks, (c) 1000-5000 blocks.

Fig. 10 displays the time performance results in the first scenario. Fig. 10 (a) shows significant performance with less than one millisecond, compared to the time required if using the Merkle Tree data structure, which is between 0.002 and 0.006 seconds, while with the Merkle Patricia Trie structure, it takes less than 0.002 seconds. Fig. 10 (b) and Fig. 10 (c), the time performance for query lookup using Merkle Patricia Trie with Linked List still wins the competition with one millisecond rather than Merkle Tree, which needs more time between 0.002 and 0.007 seconds, and the Merkle Patricia Trie requires time less than 0.002 seconds.

After the first scenario was successfully tested, the advantages of the Merkle Patricia Trie with Linked List were seen for fast performance when lookup queries. An experiment with the second scenario was carried out to ensure further the superiority of the Merkle Patricia Trie with Linked List.

In the second scenario, we increase the capacity of transactions stored in the block. The increasing number of transactions aims to determine the ability of the query process if there are various transactions stored in a block that can affect the query process. Fig. 11 shows the time performance results generated in the second scenario.

Fig. 11 (a) still shows the best time performance for query lookup using Merkle Patricia Trie with Linked List with still below one millisecond, compared to the lookup process using Merkle Tree data structure between 0.003 and 0.012 seconds. In contrast, the Merkle Patricia Trie structure takes less than 0.002 seconds. The number of transactions in a block

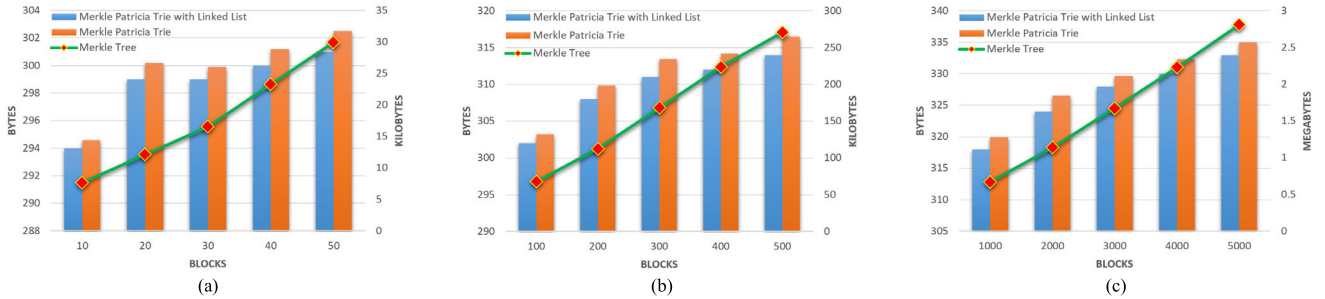

FIGURE 12. Storage capacity comparison with two transactions per block. (a) 10-50 blocks, (b) 100-500 blocks, (c) 1000-5000 blocks.

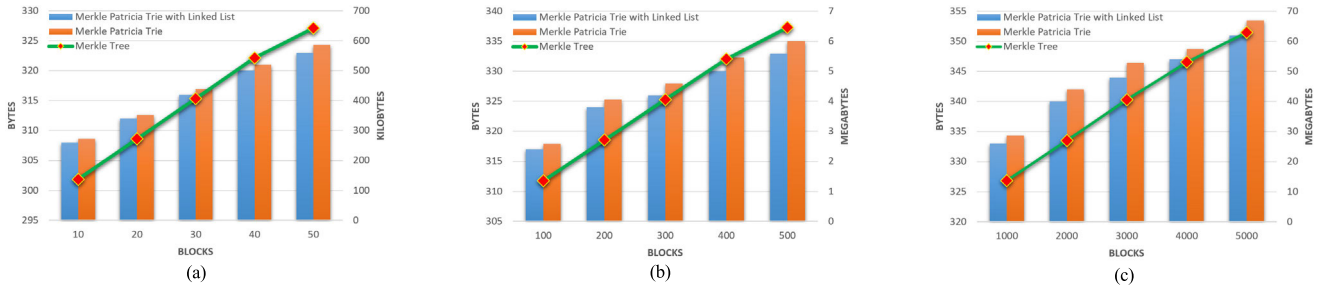

FIGURE 13. Storage capacity comparison with twenty transactions per block. (a) 10-50 blocks, (b) 100-500 blocks, (c) 1000-5000 blocks.

influences the Merkle Tree's query process, but it has little effect when using the Merkle Patricia Trie with Linked Lists. As seen in Figures 11 (b) and 11 (c), the lookup process on the Merkle Tree takes between 0.1 and 0.6 seconds. This lookup process takes too long compared to the Merkle Patricia Trie with Linked Lists, which remained stable for around one millisecond, and compared to the Merkle Patricia Trie remained for around two milliseconds.

C. STORAGE CAPACITY ANALYSIS

Besides testing the performance of the time needed to process the query, we also measure the space (storage) requirements when implementing the Merkle Patricia Trie with Linked Lists structure. This measure is necessary to see the feasibility of the proposed structure so that the existing system's performance is manageable, especially with the required space (storage). The two scenarios in Table 2 are still used to determine the required space (storage).

From the experimental results of testing the two scenarios in Table 2, it can be seen in Fig. 12 (a) the amount of space (storage) required is only around 300 to 301 bytes, and in Fig. 12 (b), it only requires 302 to 314 bytes, while in Fig. 12 (c) takes about 318 to 333 bytes. The result of the second scenario with more transactions in a block, the space (storage) requirement only requires a range of 308 to 323 bytes, as shown in Figure 13 (a). Whereas Figure 13 (b) requires 317 to 333 bytes of space (storage) and only requires 333 to 351 bytes for space (storage) requirements in the experiments in Figure 13 (c). Overall, the required amount of space (storage) is tiny, ranging from 300 to 355 bytes, when using the Merkle Patricia Trie structure needs around 2-5 percent more. In contrast, transaction data storage in the Merkle Tree will increase as various transactions are stored in the blockchain network. The experiments show that the required space (storage) increased from bytes to megabytes.

VI. CONCLUSION

This paper proposed an MSMPT, a novel high-performance data structure for multi-query processing that can provide efficient and fast multi-query services for lightweight blockchain. The proposed structure is modified from Merkle Patricia Trie used in Ethereum and combined with Linked Lists.

The proposed blockchain data structure will apply to a lightweight blockchain network. These structures adopt the structure of the Merkle Patricia Trie and then combine it with the Linked Lists structure. Ethereum uses Merkle Patricia Trie as a data structure, and LevelDB is as the backing store for this data structure. This data structure provides a cryptographically authenticated and deterministic means of storing data, ensuring that trees with the same (key, value) bindings are identical and have the same hash root. The proposed system keeps the ''key'' field to store user/node account information and modifies the existing ''value'' field, which initially contains a number or value; this field is changed to a pointer head pointing to the address of a Linked List element. Each element in the Linked List will contain the information needed to store the information needed when performing the query process.

Ethereum uses three Merkle Patricia Tries to store detailed information about users/nodes on the Ethereum network. The

proposed system uses only a single Merkle Patricia Trie and modifies the use of the ''value'' field with the Linked Lists structure. The results of testing the proposed structure show fast time performance when performing query processes. It can display multi-state information from each user/node in the designed simulator. The query process in the proposed structure only takes about one millisecond to display summary information from transactions conducted by users/nodes. The proposed system's space (storage) only requires a range of space (storage) of less than 500 bytes. The proposed data structure for multi-query processing services can be used in the IoT with large amounts of data and requires efficient and fast processing of data queries. An example is medical record data, which requires storing or processing existing data and querying according to specific categories for analysis, such as the category of the lowest blood oxygen level in one day, week, or month. Although MSMPT can enhance multi-query processing services, it can still be improved more extensively under the following aspects.

Artificial Intelligence (AI) Techniques: MSMPT does not support any AI techniques for controlling the category for each element of the linked list. Including AI techniques can significantly contribute towards MSMPT based on system usage requirements.

Cloud infrastructure: MSMPT is built using Python programming language. Cloud infrastructure can be virtualized; in-depth exploration is required to virtualize the blockchain structure in MSMPT using Python or other programming languages.

REFERENCES

[1] A. Z. Faroukhi, I. El Alaoui, Y. Gahi, and A. Amine, ''Big data monetization throughout big data value chain: A comprehensive review,'' *J. Big Data*, vol. 7, no. 1, pp. 1–22, Dec. 2020, doi: 10.1186/s40537-019-0281-5.

[2] A. Sharma, Sarishma, R. Tomar, N. Chilamkurti, and B.-G. Kim, ''Blockchain based smart contracts for Internet of Medical Things in e-healthcare,'' *Electronics*, vol. 9, no. 10, p. 1609, Oct. 2020, doi: 10.3390/electronics9101609.

[3] V. Mardiansyah and R. F. Sari, ''Lightweight blockchain framework for medical record data integrity,'' *J. Appl. Sci. Eng.*, vol. 26, no. 1, pp. 91–103, May 2022, doi: 10.6180/jase.202301_26(1).0010.

[4] A. Gautama, A. F. Rochim, and L. Bayuaji, ''Privacy preserving electronic health record with consortium blockchain,'' in *Proc. 6th Int. Conf. Inf. Technol., Inf. Syst. Electr. Eng. (ICITISEE)*, Dec. 2022, pp. 303–308, doi: 10.1109/ICITISEE57756.2022.10057649.

[5] V. Astarita, V. P. Giofrè, G. Mirabelli, and V. Solina, ''A review of blockchain-based systems in transportation,'' *Information*, vol. 11, no. 1, p. 21, Dec. 2019, doi: 10.3390/info11010021.

[6] F. M. Enescu, N. Bizon, G. Serban, and I. C. Hoarca, ''Environmental protection–blockchain solutions for intelligent passenger transportation of persons,'' in *Proc. 13th Int. Conf. Electron., Comput. Artif. Intell. (ECAI)*, Jul. 2021, pp. 1–6, doi: 10.1109/ECAI52376.2021.9515026.

[7] S. Huckle, R. Bhattacharya, M. White, and N. Beloff, ''Internet of Things, blockchain and shared economy applications,'' *Proc. Comput. Sci.*, vol. 98, pp. 461–466, Jan. 2016, doi: 10.1016/j.procs.2016.09.074.

[8] X. Li and H. Feng, ''The innovation of enterprise management mode of digital economy based on blockchain technology,'' in *Proc. Int. Conf. Knowl. Eng. Commun. Syst. (ICKES)*, Dec. 2022, pp. 1–5, doi: 10.1109/ICKECS56523.2022.10060213.

[9] C.-K. Chang, ''Blockchain for integrated nuclear power plants management system,'' *Information*, vol. 11, no. 6, p. 282, May 2020, doi: 10.3390/info11060282.

[10] M. Surjandy, H. L. H. S. Warnars, and E. Abdurachman, ''Blockchain technology open problems and impact to supply chain management in automotive component industry,'' in *Proc. 6th Int. Conf. Comput. Eng. Design (ICCED)*, Oct. 2020, pp. 1–4, doi: 10.1109/ICCED51276.2020.9415836.

[11] K. E. Pavlou and R. T. Snodgrass, ''Forensic analysis of database tampering,'' *ACM Trans. Database Syst.*, vol. 33, no. 4, pp. 1–47, Nov. 2008, doi: 10.1145/1412331.1412342.

[12] M. Iqbal and R. Matulevičius, ''Blockchain as a countermeasure solution for security threats of healthcare applications,'' in *Business Process Management: Blockchain and Robotic Process Automation Forum*, J. G. Enríquez, S. Debois, P. Fettke, P. Plebani, I. van de Weerd, I. Weber, Eds. Cham, Switzerland: Springer, 2021, pp. 67–84, doi: http://doi.org/10.1007/978-3-030-85867-4_6.

[13] A. U. Nwosu, S. B. Goyal, and P. Bedi, ''Blockchain transforming cyber-attacks: Healthcare industry,'' in *Innovations in Bio-Inspired Computing and Applications*, A. Abraham, H. Sasaki, R. Rios, N. Gandhi, U. Singh, K. Ma, Eds. Cham, Switzerland: Springer, 2021, pp. 258–266, doi: 10.1007/978-3-030-73603-3_24.

[14] S. Nakamoto. (2009). *Bitcoin: A Peer-to-peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[15] K. Tretina. *10 of the Best Cryptocurrencies in May 2022*. Forbes Media LLC. Accessed: Dec. 31, 2022. [Online]. Available: https://www.forbes.com/advisor/investing/cryptocurrency/top-10-cryptocurrencies/

[16] G. Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Accessed: Nov. 10, 2022. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[17] Y. Wang, Y. Wang, Z. Wang, G. Yang, and X. Yu, ''Research cooperations of blockchain: Toward the view of complexity network,'' *J. Ambient Intell. Humanized Comput.*, vol. 13, no. 3, pp. 1339–1352, Mar. 2022, doi: 10.1007/s12652-020-02596-6.

[18] P. Li, K. Li, Y. Wang, Y. Zheng, D. Wang, G. Yang, and X. Yu, ''A systematic mapping study for blockchain based on complex network,'' *Concurrency Comput., Pract. Exper.*, vol. 34, no. 14, Jun. 2022, Art. no. e5712, doi: 10.1002/cpe.5712.

[19] J. Bonneau, ''EthIKS: Using Ethereum to audit a CONIKS key transparency log,'' in *Financial Cryptography Data Security*. J. Clark, S. Meiklejohn, P. Y. A. Ryan, D. Wallach, M. Brenner, K. Rohloff, Eds. Berlin, Germany: Springer, 2016, pp. 95–105, doi: 10.1007/978-3-662-53357-4_7.

[20] R. Zhang, R. Xue, and L. Liu, ''Security and privacy on blockchain,'' *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–34, May 2020, doi: 10.1145/3316481.

[21] B. Singhal, G. Dhameja, and P. S. Panda, ''How Blockchain Works,'' in *Beginning Blockchain: A Beginner's Guide to Building Blockchain Solutions*, B. Singhal, G. Dhameja, P. S. Panda, Eds. Berkeley, CA, USA: Apress, 2018, pp. 31–148.

[22] B. Singhal, G. Dhameja, and P. S. Panda, ''How Ethereum works,'' in *Beginning Blockchain: A Beginner's Guide to Building Blockchain Solutions*, B. Singhal, G. Dhameja, P. S. Panda, Eds. Berkeley, CA, USA: Apress, 2018, pp. 219–266.

[23] D. Vujicic, D. Jagodic, and S. Randic, ''Blockchain technology, Bitcoin, and Ethereum: A brief overview,'' in *Proc. 17th Int. Symp. INFOTEH-JAHORINA (INFOTEH)*, Mar. 2018, pp. 1–6, doi: 10.1109/INFOTEH.2018.8345547.

[24] S. Ghemawat and J. Dean. *LevelDB*. Accessed: Nov. 3, 2021. [Online]. Available: https://github.com/google/leveldb

[25] S. S. Dang. *Understanding the Blockchain Layered Architecture to Solve The Scalability Challenges*. Forbes Digital Assets. Accessed: Feb. 10, 2023. [Online]. Available: https://www.forbes.com/sites/sanjitsinghdang/2022/10/24/understanding-the-blockchain-layers-to-solve-the-scalability-challenges/

[26] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, ''An overview of blockchain technology: Architecture, consensus, and future trends,'' in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, Jun. 2017, pp. 557–564, doi: 10.1109/BigDataCongress.2017.85.

[27] R. C. Merkle, ''A digital signature based on a conventional encryption function,'' in *Advances in Cryptology CRYPTO '87*, C. Pomerance, Ed. Berlin, Germany: Springer, 1988, pp. 369–378.

[28] C. Yue, Z. Xie, M. Zhang, G. Chen, B. C. Ooi, S. Wang, and X. Xiao, ''Analysis of indexing structures for immutable data,'' in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2020, pp. 925–935, doi: 10.1145/3318464.3389773.

[29] D. L. K. Chuen, *Handbook of Digital Currency: Bitcoin, Innovation, Financial Instruments, and Big Data*, 1st ed. New York, NY, USA: Academic, 2015.
[30] S. Krishnan, V. E. Balas, E. J. Golden, Y. H. Robinson, S. Balaji, and R. Kumar, *Handbook of Research on Blockchain Technology*, 1st ed. New York, NY, USA: Academic, 2020.
[31] D. Hellwig, G. Karlic, and A. Huchzermeier, *Build Your Own Blockchain: A Practical Guide to Distributed Ledger Technology* (Management for Professionals), 1st ed. Cham, Switzerland: Springer, 2020, pp. 16–187.
[32] DeHive. *How to Solve the Blockchain Data Query Issue and What the Graph Service Has to Say*. CoinMarketCap. Accessed: Jan. 18, 2023. [Online]. Available: https://coinmarketcap.com/alexandria/article/how-to-solve-the-blockchain-data-query-issue-and-what-the-graph-service-has-to-say
[33] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "FogBus: A blockchain-based lightweight framework for edge and fog computing," *J. Syst. Softw.*, vol. 154, pp. 22–36, Aug. 2019, doi: 10.1016/j.jss.2019.04.050.
[34] D. Hanggoro and R. F. Sari, "A review of lightweight blockchain technology implementation to the Internet of Things," in *Proc. IEEE R10 Humanitarian Technol. Conf. (R10-HTC)*, Nov. 2019, pp. 275–280, doi: 10.1109/R10-HTC47129.2019.9042431.
[35] D. Stefanescu, L. Montalvillo, P. Galán-García, J. Unzilla, and A. Urbieta, "A systematic literature review of lightweight blockchain for IoT," *IEEE Access*, vol. 10, pp. 123138–123159, 2022, doi: 10.1109/ACCESS.2022.3224222.
[36] A. Al-Mamun, F. Yan, and D. Zhao, "SciChain: Blockchain-enabled lightweight and efficient data provenance for reproducible scientific computing," in *Proc. IEEE 37th Int. Conf. Data Eng. (ICDE)*, Apr. 2021, pp. 1853–1858, doi: 10.1109/ICDE51399.2021.00166.
[37] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. 48th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, Oct. 2007, pp. 94–103, doi: 10.1109/FOCS.2007.66.
[38] Y. Li, K. Zheng, Y. Yan, Q. Liu, and X. Zhou, "EtherQL: A query layer for blockchain system," in *Database Systems for Advanced Applications*, Cham, S. Candan, L. Chen, T. B. Pedersen, L. Chang, W. Hua, Eds. Cham, Switzerland: Springer, 2017, pp. 556–567, doi: 10.1007/978-3-319-55699-4_34.
[39] K.-B. Yue, K. Chandrasekar, and H. Gullapalli, "Querying Bitcoin blockchain using SQL," in *Proc. 2018th EDSIG Conf. Inf. Syst. Comput. Educ.*, Norfolk, VA, USA, Oct. 2018, Paper 4607. [Online]. Available: https://iscap.us/proceedings/2018/index.html
[40] I. Riadi, T. Ahmad, R. Sarno, P. Purwono, and A. Ma'arif, "Developing data integrity in an electronic health record system using blockchain and InterPlanetary file system (case study: COVID-19 Data)," *Emerg. Sci. J.*, vol. 4, pp. 190–206, Feb. 2022, doi: 10.28991/esj-2021-SP1-013.
[41] J. Xu, Y. Tian, T. Ma, and N. Al-Nabhan, "Intelligent manufacturing security model based on improved blockchain," *Math. Biosci. Eng.*, vol. 17, no. 5, pp. 5633–5650, 2020, doi: 10.3934/mbe.2020303.
[42] X. Yang, Y. Zhang, S. Wang, B. Yu, F. Li, and Y. Li, "LedgerDB: A centralized ledger database for universal audit and verification," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 3138–3151, 2020.
[43] X. Yang, S. Wang, F. Li, Y. Zhang, W. Yan, F. Gai, B. Yu, L. Feng, Q. Gao, and Y. Li, "Ubiquitous verification in centralized ledger database," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 1808–1821, doi: 10.1109/ICDE53745.2022.00181.
[44] L. Kong, Y. Dou, Q. Yin, X. Min, and Q. Li, "WST+iMPT: A high-performance incremental verification world state model for massive accounts," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. With Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Sep. 2021, pp. 263–270, doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00046.
[45] X. Xing, Y. Chen, T. Li, Y. Xin, and H. Sun, "A blockchain index structure based on subchain query," *J. Cloud Comput.*, vol. 10, no. 1, p. 52, Dec. 2021, doi: 10.1186/s13677-021-00268-0.
[46] D. Przytarski, C. Stach, C. Gritti, and B. Mitschang, "Query processing in blockchain systems: Current state and future challenges," *Future Internet*, vol. 14, no. 1, p. 1, Dec. 2021. [Online]. Available: https://www.mdpi.com/1999-5903/14/1/1
[47] Q. Qu, I. Nurgaliev, M. Muzammal, C. S. Jensen, and J. Fan, "On spatio-temporal blockchain query processing," *Future Gener. Comput. Syst.*, vol. 98, pp. 208–218, Sep. 2019, doi: 10.1016/j.future.2019.03.038.
[48] R. Smith. *Merkle Patricia Trie*. Ethereum Developers Doc. Accessed: Apr. 1, 2023. [Online]. Available: https://ethereum.org/en/developers/docs/data-structures-and-encoding/patricia-merkle-trie/
[49] V. Mardiansyah and R. F. Sari, "SimBlock simulator enhancement with difficulty level algorithm based on Proof-of-Work consensus for lightweight blockchain," *Sensors*, vol. 22, no. 23, p. 9057, Nov. 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/23/9057

VIDDI MARDIANSYAH (Member, IEEE) received the B.Sc. degree in mathematics and natural science, majoring in computer science from Universitas Padjajaran, Bandung, Indonesia, in 1998, the M.Eng. degree in computer science/informatics, majoring in software engineering from Institut Teknologi Bandung, Indonesia, in 2007, and the Dr. (Eng.) degree in electrical engineering, majoring in computer engineering from Universitas Indonesia, in July 2023. Since 2019, he has been pursued his Ph.D. degree with the Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia.

Since 2015, he has been a permanent Computer Science/Informatics Lecturer for undergraduate students with the Faculty of Engineering, Universitas Widyatama, Bandung. His research interests include blockchains, software engineering, programming, networking, and the Internet of Things.

ABDUL MUIS (Member, IEEE) received the B.E. degree in electrical engineering, majoring in control engineering from Universitas Indonesia, in 1998, and the master's and Ph.D. degrees in mechatronics and friendly motion control from Keio University, in 2004 and 2007, respectively.

He was with the Electrical Engineering Department, Universitas Indonesia, in 1999. Since returning to Universitas Indonesia as a permanent Lecturer, in 2007, his knowledge has recently expanded to embedded systems, information systems, and related industry 4.0. His research interests include mechatronics, motion control, vision-based control, and automation.

RIRI FITRI SARI (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the University of Indonesia (UI), the master's degree in human resource management from Atmajaya University, Jakarta, the M.Sc. degree in software systems and parallel processing from the Department of Computer Science, University of Sheffield, U.K., and the Ph.D. degree in computer networking from the School of Computing, University of Leeds, U.K.

Since April 2010, she has been the Chairperson of UI GreenMetric Ranking of World Universities, a flagship program to rank universities worldwide based on their green campus and sustainability. She is currently a Professor in computer engineering with the Department of Electrical Engineering, Faculty of Engineering, UI. She has been a member of the Special Task Force for Improving the Academic Reputation of Indonesian Higher Education Institutions at the Ministry of Research and Higher Education, since September 2015. She also received various awards, including the 2012 Inspirational Women and Youth from PT Indosat and the 2013 Kartini 2.0 Women in Technology from PT Telkom and a Honorary Professor from Kazakhstan Agrarian National University, in 2017. In 2023, she was awarded the prestigious Habibie Prize. She received the British Council Chevening Award for the M.Sc. degree.