Journal of Computational Science 83 (2024) 102457

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

BPMT: A hybrid model for secure and effective electronic medical record management system

Mehar Nasreen, Sunil Kumar Singh *

Dept. of Computer Science and Information Technology, Mahatma Gandhi Central University, Motihari, Bihar, India

ARTICLE INFO

Keywords:
EMR
Merkle tree
B+ tree
Integrity
Throughput
Security

ABSTRACT

Designing an efficient distributed blockchain system to store and access rising electronic medical records (EMR) is still challenging. Although many techniques have been designed to utilize EMRs proficiently, they still suffer from accessing time, storage time, and throughput. In this work, we have proposed a Merkle and B+ tree-based hybrid model to improve the performance of the blockchain system. B+ tree has improved the system's performance because of its practical layout and balanced structure. At the same time, Merkle Tree has ensured the integrity and security of the information. The proposed (B+ Merkle tree) BPMT model has reduced the storage time of records, improved the system's throughput, and significantly reduced memory use. These enhancements are supported by a thorough complexity analysis and testing on several datasets, which validate the efficacy and scalability of the model. The outcomes validate the performance improvements of the model and lay the groundwork for further developments in blockchain-based EMR management.

1. Introduction

An EHR (Electronic Health Record) is a data-centric collection that refers to the patient's medical history in digital format. Observations, laboratory results, diagnostic imaging reports, therapies, pharmaceuticals delivered, patient identification data, legal permits, and allergies are only a few instances of the numerous different sorts of data that are contained in EHR, and enormous amounts of medical data are produced daily. Therefore, it is becoming increasingly crucial to store these data effectively and securely [1].

There are several scientific and practical barriers, as with any maturing consumer technology. Since so many current EHR systems rely on centralized servers, for instance, these deployments inherit the security and privacy drawbacks of the centralized server approach, such as single points of failure and speed constraints [2]. To overcome this problem, blockchain technology can be used.

Blockchain's peer-to-peer distributed ledger technology gives an immutable and transparent understanding of all transactions that occur. Data is recorded in the chain ledger as transactions, and it is digitally signed into blocks in chronological sequence. The shared and immutable data storage in blockchain technology makes it a popular alternative, and it is a feasible solution to avoid centralized reliance [3]. The complexity and expense of a contemporary healthcare system are significant in many circumstances, and data transfers vary depending on design, however, using blockchain technology could enhance healthcare record management as well as expenses [4]. The usage of blockchain in several fields, including healthcare (such as public healthcare administration, the avoidance of counterfeit medicines, and clinical trials), has increased recently. Better patient services are provided because of the healthcare system's increased accuracy and speed in the dissemination of patient information [5–7].

Furthermore, Blockchain technology has considerable effectiveness in the Internet ecosystem since it could decrease expenses, improve efficiency, and increase security. In addition to storage, it's important to ensure effective authentication and authorization through the access control system. To develop effective tools, modifications to an existing data structure's method for protected storage and access control regarding the processing of transactions must be thoroughly investigated [8].

When the data is more exposed, as it is in a select few sectors like healthcare, it becomes crucial to design and construct the data structure as well as storage security while concurrently preserving the integrity management. This study aims to present a safe storage access model for health data based on the Merkle tree and the blockchain that can improve integrity management.

Ralph Merkle patented the idea of Merkle tree in 1979 and changed

* Corresponding author.
E-mail addresses: mailme.meharnasreen@gmail.com (M. Nasreen), sunilsingh.jnu@gmail.com, sksingh@mgcub.ac.in (S.K. Singh).

https://doi.org/10.1016/j.jocs.2024.102457
Received 13 November 2023; Received in revised form 31 July 2024; Accepted 18 October 2024
Available online 26 October 2024

the name from Binary hash tree to Merkle tree in 1987 [9]. It discusses how to create a new digital signature that is as safe as the underlying encryption mechanism while avoiding the high computing expenses of modular arithmetic and just using a regular encryption method like DES (Data Encryption Standard).

As we are all aware, an integral component of blockchain technology is the Merkle tree. To ascertain whether the entire network has accepted a transaction, it is not essential to download all the transaction data. One of the hottest research topics now is for faster blockchain transaction verification speed since the Merkle tree has a set verification period for each recorded transaction. The Merkle tree is crucial for the network's long-term viability because as the blockchain expands over time, complete nodes require huge amounts of data storage, making it challenging to maintain full nodes on personal computers [10].

Merkle trees are generated by hashing the pair of nodes repeatedly until only one hash is left, called Merkle root. A Merkle root in a blockchain holds details on every transaction hash that has ever been on a specific block. Fig. 1 outlines the organization of a block in the blockchain network in which the Merkle root is stored in the block header and the transactions are stored in the block body.

This paper aims to investigate the following:

- To design a novel Merkle and B+ tree-based BPMT model for effective structure with an efficient transaction verification mechanism.
- To ensure secure EHR data transmission and verification in the digital healthcare system.
- The model exhibits significantly reduced tree heights compared to traditional Merkle tree implementations. This improvement is validated through extensive testing on numerous datasets and thorough complexity analysis, demonstrating its enhanced efficacy and scalability.

The research has been further divided into the sections listed below. Section 2 outlines the relevant prior works in this context. In Section 3, the proposed model and the conventional Merkle tree structure are presented, along with a description of the several algorithms that go with it. A complexity study of these algorithms is also included in this section, providing information on their scalability and computing efficiency. An experimental analysis and mathematical representations for the proposed method in trustworthy blockchain systems are presented in Section 4. Two distinct datasets—1000 patient records with lung cancer and 2000 patient records with bloodstream infections—are used in the experimental research to evaluate the concept. A thorough assessment of the model's performance across a variety of datasets and medical circumstances is ensured by this more extensive testing approach. The work is finally brought to a close in Section 5, which also offers some future prospects and a summary of the findings. It highlights the successes of the proposed model and offers some areas for future research.

2. Related work

In this section, we outline the most relevant prior works done for the improvement of the electronic health records storage scheme.

Researchers and developers are always looking for new and creative methods to improve data availability, integrity, and efficiency in the rapidly changing field of blockchain technology. Amidst these advancements, the Merkle tree has surfaced as a crucial framework, undergoing several modifications and enhancements to satisfy the expanding requirements of diverse blockchain applications. A Coded Merkle Tree (CMT) is proposed as a hash accumulator with sparse fraud protection to avoid data availability and integrity problems [11]. The existence of a Merkle proof for each coded symbol is encoded and low-density parity-check codes are ensembled at each layer of the Merkle tree. Further, it is retrieved by progressively using a peeling-decoding technique. A CMT modular library in Rust and Python is provided and parity's bitcoin client serves as a platform to test its effectiveness. Yet, significant obstacles that must be overcome before it can be widely used include the complexity of its implementation and the large amount of processing power needed for encoding and decoding.

In [12], Blockchain identity management is proposed for

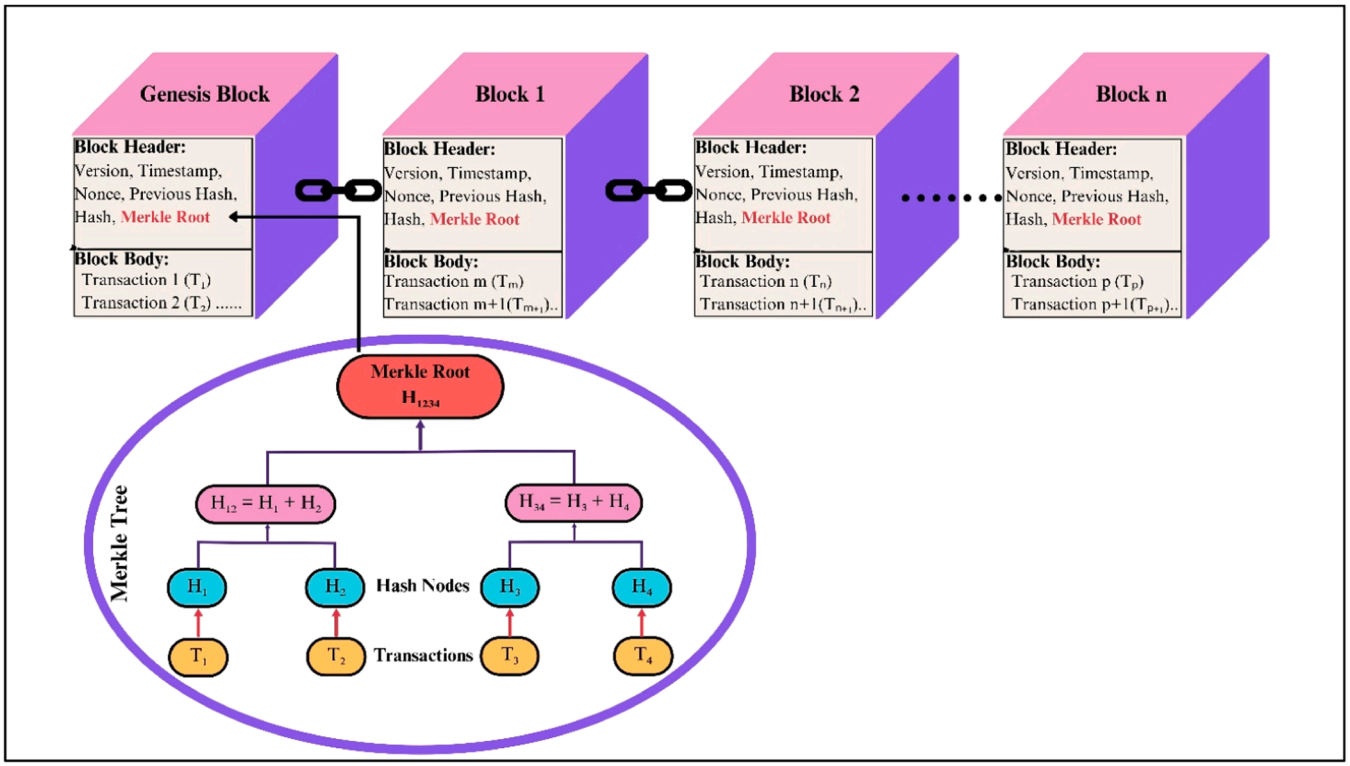

Fig. 1. Organization of a Block in the Blockchain.

safeguarding the authentication, ownership, and identity mapping validation while sharing data securely. Elliptic curve cryptography creates the private and public keys. The keys created using this method will be smaller, lowering the computational overhead, and requiring less memory. MD5 is used for the evaluation of hash to guarantee the effectiveness of encryption. However, depending on MD5 for hash assessment is questionable since it is susceptible to collisions, which might undermine the encryption's dependability and security.

Educational certificate blockchain is introduced for high throughput and low latency [13]. A tree structure after combining the Patricia tree and the Merkle tree is created, called MPT-Chain, and implemented to boost the speed of query transaction and verification of the blocks. The MPT-Chain node maintains the Merkle root's intermediate value, which is reused when the branch is not updated and can reduce the consumption of computer resources. Additionally, it helps speed up the process of creating blocks. However, managing the integration of Patricia and Merkle trees can be complex, and there may be potential issues with scalability as the system grows.

An improved Merkle tree is created in which convolutional layer operation is used in place of binary tree calculation method due to which while creating root node, the number of hash calculations decreases [14]. However, this method may increase computational complexity and need more specialized hardware or software for optimal execution. A Secure Merkle tree (SM-tree) is proposed for the efficient transmission of CCTV video data reducing bandwidth and duplicity [15]. A hash tree is constructed by dividing the plain text into encrypted data and metadata. A cloud-based CCTV blockchain system with a modified Merkle tree also controls the block's version information independently and increases the synchronization cost. The suggested solution can also sync the data from CCTV videos securely without disclosing the object's confidentiality while synchronization is taking place. There are some limitations to this method, such as higher synchronization costs and significant implementation issues.

SE-Chain model [16] is designed that use the adaptive balanced Merkle tree (AB-M tree) structure to boost the blockchain's ability to retrieve data effectively. This model stores each transaction in the AB-M tree for the quick verification of data. A duplicate ratio regulation algorithm is also used for the reduction of each entire node's storage capacity in the blockchain system. The minimum number of duplicates needed for each block is determined by the security criterion. Yet, it has downsides, such as the complexity of creating the AB-M tree and regulation method, as well as possible scalability concerns.

A biometric identity management system for the smart industry is proposed [17]. Blockchain technology is used in which the Merkle tree is replaced with RSA accumulators for off-chain storage and thus improves the verification performance but it may introduce complexity. To increase the effectiveness of transaction verification in blockchain-based industrial IoT (Internet of Things) systems with heterogeneous devices, an optimized Merkle tree structure is proposed [18]. Different parameters of the classic Merkle tree such as the quantity of hash codes broadcast, how many hash calculations are performed and how long it takes to verify, are tracked with their work. Also, the verification probability for each transaction is specified.

In [19], a modified Merkle tree (MMT) is proposed that ensures storage security by creating a 3-stage Merkle tree based on hashed transactions. The maintenance of data also includes authentication, consistency checking, and synchronization that leads to make it complex in management. A faster Merkle root hash calculation method is proposed, which categorizes the calculation of Merkle root in the header of the block into four groups [20]. One hash function is used to perform the measurements for each group. The same hash function ought to be used by two consecutive groups. Although using this approach shortens computing times, it also adds complexity to group administration and raises the possibility of synchronization problems across groups.

Using tweaks, a secure prove-your-ownership-of-the-data approach protocol that supports replication is introduced in [21]. Merkle tree binary structure is replaced with B+ tree to generate the replicas of blocks and is uploaded in the clouds after encryption. Each block is retained in the leaf nodes and connected to the other blocks using pointers to form a linked list, which facilitates the swift traversal of data blocks. It has downsides, including the complexity of managing the B+ tree structure and the possibility of data integrity difficulties during replication.

Currently, the ADS implementation approaches include authentication skip lists [22], index hash lists [23], and Merkle hash trees (MHTs) [24]. Merkle hash trees are the most extensively used structure because they can authenticate all leaf values with a single signature, need only logarithmic operations, and can easily track altered data. With trapdoor hashes and BLS signatures combined into a Merkle hash tree, the Privacy-Preserving Adaptive Trapdoor Hash Authentication Tree (P-ATHAT) improves data stream verification [25]. It extends dynamically, fixes single point of failure concerns, and enhances real-time authentication. But because it's more complicated, there can be more computing overhead.

The paper suggests adopting a trusted execution environment (TEE) in blockchain systems to decrease verification object (VO) overhead for light clients [26]. By arranging data hierarchically and buffering hot data in the TEE enclave, the method decreases VO size and increases efficiency dramatically. However, TEEs' limited secure memory can still restrict their performance in large-scale applications. Smart homes have issues with relational database security and limited blockchain storage [27]. The IBPAS system improves efficiency, lowers transmission costs by 12 %- 39 %, and increases storage performance by 20 %. However, it may be difficult to deploy and integrate with current systems.

Even though these approaches are innovative, they have significant shortcomings. They include possible scalability problems brought on by higher processing cost from sophisticated encryption techniques like MD5 hashing and elliptic curve cryptography. Further, the use of specific Merkle tree variations, including RSA accumulators and AB-M trees, may cause compatibility issues with other blockchain systems. Nevertheless, despite their attempts to improve security and efficiency, systems like the Modified Merkle Tree (MMT) can still have complicated implementations and need a significant amount of power. The existing approaches take longer to generate and traverse the tree structure than they do to check the consistency and integrity verification of the data. This is because higher-degree tree data structures have drawbacks. To accommodate data-centric applications like remote patient monitoring and healthcare, a less complex Merkle tree must be created immediately. In contrast, the suggested Merkle and B+ tree-based hybrid model provides a more efficient solution by combining the B+ tree's balanced structure, which allows for quick record retrieval, with the Merkle tree's data integrity. This design considerably decreases storage time, increases throughput, and lowers memory use, making it a more practical and resource efficient alternative to the other more sophisticated and resource-intensive solutions.

3. Proposed Work

In this section, we have discussed the proposed model along with the associated concepts which are as follows.

3.1. Blockchain technology in healthcare

Blockchain enables several clinicians from various locations to access the same real-time data, enhancing decision-making. An impenetrable decentralized system is also created for storing patient medical records. Blockchain creates a permanent audit trail that makes it easy to track network changes.

Despite several advantages of blockchain, it would be expensive and inefficient to store massive records on it, such as complete electronic medical records or records of genetic material. Additionally, it is challenging to query data within a blockchain, which restricts uses in

medicine, statistics, and research. For users of the network, it violates their right to privacy about their data. Records in encrypted format will last forever and grow quite large because of the immutable nature of blockchain requiring a lot of memory to store [28].

3.2. Merkle tree

The Merkle tree hashes paired data as leaves or nodes on the tree, then pairs and hashes the outcomes until only one hash is left, which is the Merkle root [29]. The Merkle root, the tree's root node, is the ultimate successor of all other nodes. In blockchain, all the transactions are hashed in the Merkle tree [30–32]. The block only stores the hashed root of the Merkle tree, allowing for the pruning of tree branches to compact old blocks. This approach eliminates the need to store internal hashes and aids in verifying the consistency and content of the data. Merkle trees are constructed using a bottom-up methodology and are integral to the architectures of both Bitcoin and Ethereum.

A Merkle tree structure of 5 layers is shown in Fig. 2 and is created as follows:

1. Let T_i be the set of transactions, where $i \in (1, 8)$.
   $T_i = \{T_1, T_2, T_3, \ldots, T_8\}$
   And let H_i be the set of corresponding hashes for the transactions T_i, where $i \in (1, 8)$.
   $H_i = \{H_1, H_2, H_3, \ldots, H_8\}$
   In mathematical notation,
   $H_1 = Hash(T_1)$
   $H_2 = Hash(T_2)$
   $H_3 = Hash(T_3)$
   $H_8 = Hash(T_8)$
   This represents that each hash H_i is generated by applying the hash function *Hash* to the corresponding transaction T_i.
2. Then,
   $H_{12} = Hash(H_1 + H_2)$
   $H_{34} = Hash(H_3 + H_4)$
   $H_{56} = Hash(H_5 + H_6)$
   $H_{78} = Hash(H_7 + H_8)$
3. Further the hashes of intermediate nodes are generated.
   $H_{1234} = Hash(H_{12} + H_{34})$
   $H_{5678} = Hash(H_{56} + H_{78})$
4. Finally, intermediate nodes are hashed to create the Merkle root of the tree as

   $H_{12345678} = Hash(H_{1234} + H_{5678})$

Instead of storing the entire Merkle tree in the block header, just the root of the tree, $H_{12345678}$ is stored to determine whether a transaction has altered it. If any content has been changed in a transaction, the parent node's hash will be affected, which will change the hash of the higher-level node, and so on until the root is reached. Transactions can be accessed by navigating through the hash pointers. Merkle root verifies the accuracy of the data and renders transactions tamper-proof. Merkle proof decides whether the data belongs to the Merkle tree or not. It is created by hashing the respective matching hashes, and then moving up the tree until it reaches to the root hash.

3.3. The proposed model

Our proposed model, BPMT (B+ Merkle tree) is presented in Fig. 3. Merkle trees are used to verify data integrity. They aren't designed to be used as B+ trees for data storage or retrieval. B + Tree is a variation of B-tree in which the leaves carry pointers to list entries that are arranged in ascending order by key [33]. The main tasks of the Merkle tree are building the tree, computing hashes, and ensuring data integrity. Whereas B+ trees are easy to store and retrieve information because of their effective layout and balanced structure. To streamline data storage and retrieval procedures, the B+ tree data structure is commonly employed in databases and filesystems. The BPMT follows the same formatting, updating, and querying rules as the conventional B+ tree and it uses the Merkle tree for the verification of data integrity.

- A leaf node's structure differs from the structure of internal nodes in the proposed tree structure. Each value of the search field has an entry in the leaf nodes, coupled with a data pointer to the record or the block containing this record. The leaf nodes are linked together to provide ordered access to the records' search field. Internal nodes are used to guide the search.

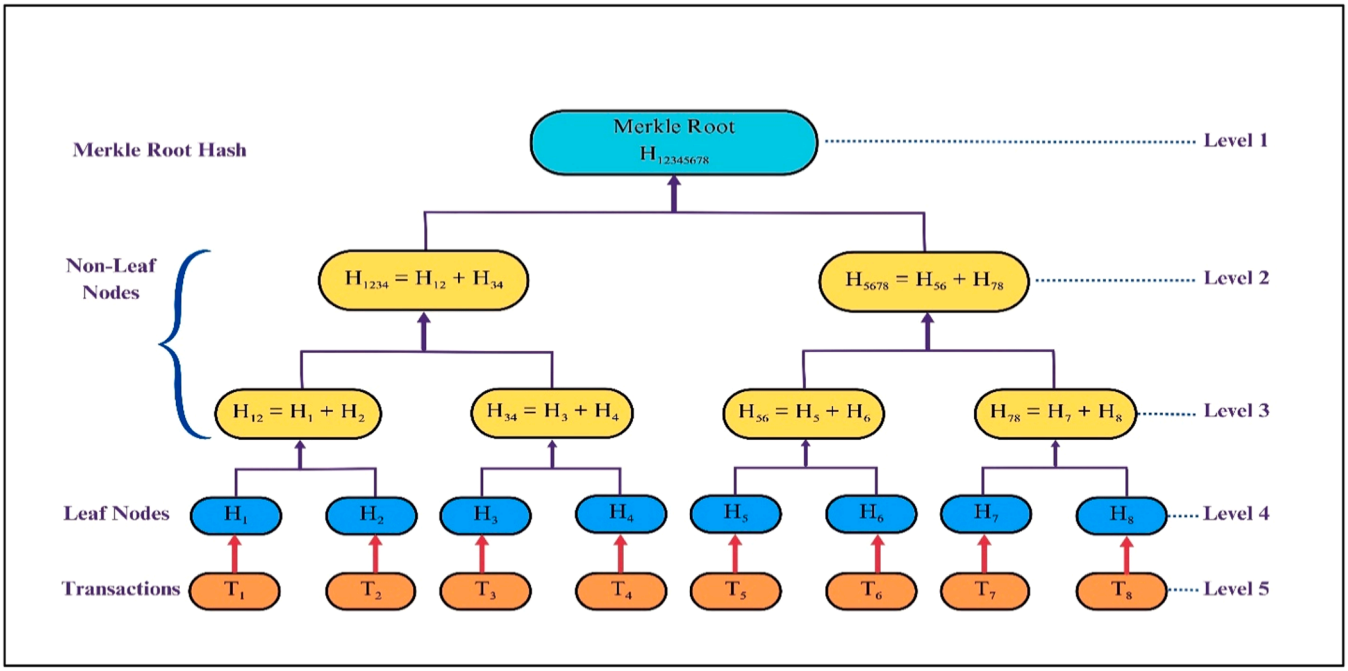

Fig. 2. Merkle Tree.

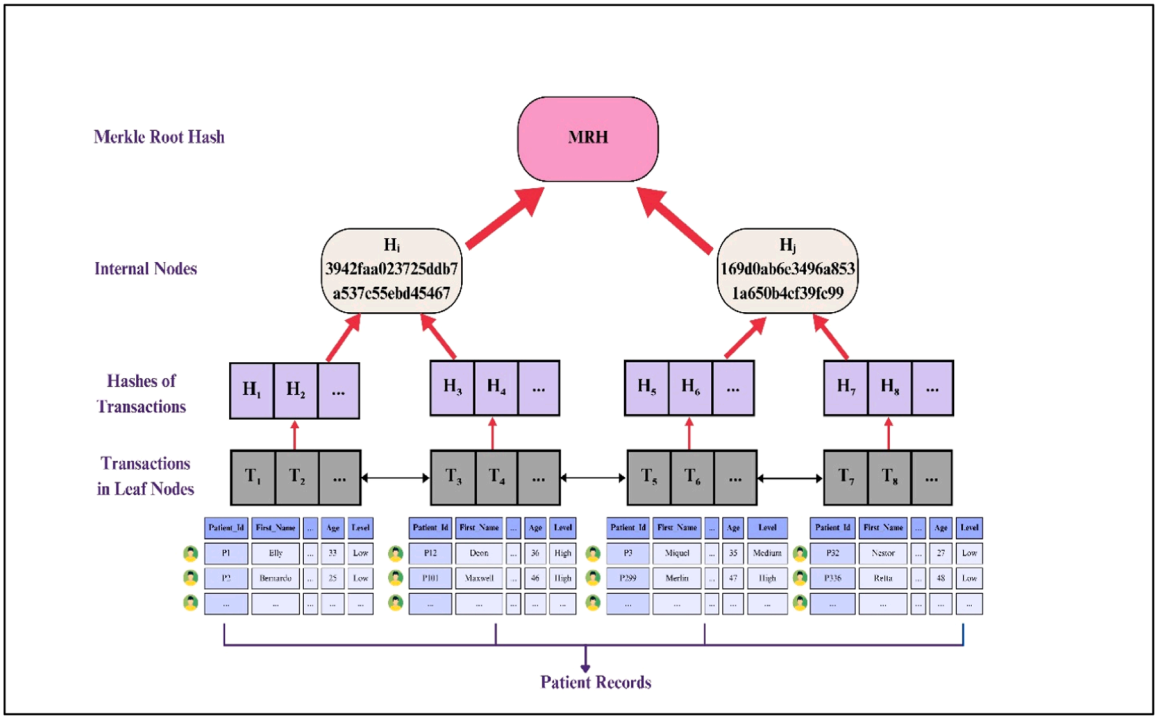

Fig. 3. Improved Hybrid B+ Merkle tree Structure.

- The proposed tree with 'l' levels has a larger capacity for internal node storage than a B-tree with the same number of levels. This highlights the significant improvement in search time for each key. Because of the lower level and the presence of P_{next} pointers (that point to the next leaf node in the tree structure), the proposed tree accesses records from drives very quickly and effectively.
- The proposed tree's data can be accessed sequentially as well as directly.

We use the B+ tree-based storage scheme that stores all the transactions of a block in the leaf node and are linked together. These transactions consist of the records of patients with lung cancer. The Leaf Node Structure of BPMT of order m is shown in Fig. 4, which can be defined as shown in Eq. (1).

$$LN = ((K_1, P_1), (K_2, P_2), \ldots, (K_{n-1}, P_{n-1}), P_{next}) \quad (1)$$

where, $n \leq m$

$K_i = \{K_1, K_2, \ldots, K_{n-1}\}$: Key values pointing to the actual record in the disk file block.

$P_i = \{P_1, P_2, \ldots, P_{n-1}\}$: Pointers to the corresponding patient records in the disk file.

P_{next}: Points to the next leaf node in the tree structure.

The proposed structure stores all the leaf nodes at the same level.

The transactions, $T_i = \{T_1, T_2, \ldots, T_n\}$ of a block are represented as the leaf nodes and their hashes, $H_i = \{H_1, H_2, \ldots, H_n\}$, are generated.

These hashes are now combined, just like in the Merkle tree, to create a new Merkle root hash (MRH) as indicated in Eq. (2).

$$MRH = MerkleRoot(H_i) \quad (2)$$

Eq. (3) presents the Internal nodes storing the hashes created after adding the hashes of the leaf nodes.

$$IN = IH_1, IH_2, \ldots, IH_k \quad (3)$$

where k represents the number of internal nodes in the tree.

3.4. The algorithms

In the proposed work, a hybrid model for a secure and effective EHR management system named BPMT, is implemented. The method used to create the tree is shown in Algorithm (1). The first step in the algorithm is to define the node structure, which consists of fields for hash values, child pointers, keys, and values. To produce leaf and internal nodes, respectively, utilize the createLeaf(keys, values) and createNonLeaf (keys, children) functions. In order to assure data integrity, hash values are calculated using the CalHash(N) process, which applies the SHA256 hashing algorithm. The methods insert(root, key, value) and insertIntoLeaf(leafNode, key, value) are used to add and manage records. While the Verify() function verifies the correctness of the data by comparing the computed Merkle root with a specified root, the findInsertPosition (keys, key) technique determines where new keys should be added. In the B+ tree, a node's list or array of existing keys is indicated as 'keys', whereas a new single key to represent a new record is indicated as 'key'. The right location for the new key within the keys array is found using the method findInsertPosition(keys, key). Because effective searching and indexing within the B+ tree depend on the keys remaining sorted, this guarantees it. With the help of this hybrid technique, blockchain records are arranged into a B+ tree structure, each leaf node being connected, facilitating safe and effective record addition, retrieval, and deletion. B+ tree indexing combined with Merkle tree hashing improves the security and management of EHR systems. The whole collection of notations used in the algorithms is displayed in Table 1.

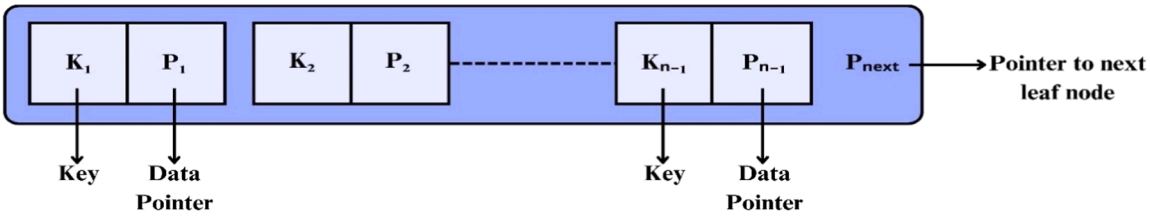

Fig. 4. Structure of the leaf node in BPMT.

Table 1
List of Notations.

Notation	Meaning
LN	Leaf Node
IN	Internal Node
CalHash	Calculate the hash value
N	Node of the blockchain
H_C	Calculated hash value
V_L	Validate the left node of tree
V_R	Validate the right node of tree
L	Sibling path length
S	Block size
H	Hash value size
MP	Size of the Merkle proof
D	Target size of data
T_i	Total number of blocks
t_i	Total time taken in seconds

Algorithm 1. Algorithm to generate the B+ Merkle Tree

Input: keys, values, children, root, key, value
Output: Generates the proposed tree

```
struct Node
def procedure createLeaf(keys, values)
def procedure createNonLeaf(keys, children)
def procedure CalHash(N)
def procedure insert(root, key, value)
def procedure insertIntoLeaf(leafNode, key, value)
def procedure findInsertPosition(keys, key)
def procedure Verify()
```

Input: N := root, String H_C, boolean V_L , V_R
Output: Verifies all the transactions in the block

```
procedure Verify()
    procedure VerifyNode(N)
        if (N == null)
            return true
        end if
        if (N.GetLeft() == null ∧ N.GetRight() == null)
            return N.GetHash().equals(N.GetBlock().GetHash())
        end if
        if (N.GetRight() == null)
            HC := CalHash(N.GetLeft().GetHash())
        Else
            HC := CalHash(N.GetLeft().GetHash() + N.GetRight().GetHash())
        end if
    VL := VerifyNode(N.GetLeft())
    VR := VerifyNode(N.GetRight())
    return VL ∧ VR ∧ HC.equals(N.GetHash())
end procedure
```

Algorithm 2 illustrates how to check data integrity with the Merkle root, which is stored in the block's header and enables for efficient validation without requiring the complete block. The Verify() procedure begins the verification process by using the Merkle root and Merkle proofs. The VerifyNode(N) procedure compares hashes to ensure the integrity of tree nodes. If node N is null, it is considered valid. For leaf nodes, it ensures that the stored hash matches the transaction block hashes. For non-leaf nodes, it computes the hash of the left and right child nodes and compares it to the node's stored hash. By confirming that all hashes are precise and consistent, this recursive verification procedure updates the boolean values V_L and V_R, therefore validating individual transactions or groups of transactions inside the block.

Algorithm 2. Algorithm to verify the integrity of data

Algorithm 3 secures the consistency of each block in a blockchain by running a mining operation to check the block's hash against the appropriate difficulty level. The addBlock(Block newBlock) operation begins with mineBlock(difficulty), which increments the nonce and recalculates the hash until it reaches the difficulty string returned by getDifficultyString(difficulty). This mining mechanism guarantees that the hash of each block is genuine and matches the hash of the previous block, hence preserving on-chain integrity. Each block has a header containing transaction data, and a proof of work (PoW) is used to validate it. Block consistency is assessed using transaction hashes, Merkle roots, block hashes, and prior hashes. If any component of a block is changed, the entire blockchain becomes invalid. This procedure assures that all participants agree on the same set of blocks, hence maintaining the blockchain's security and integrity.

Algorithm 3. Algorithm to check the consistency of block

Input: int difficulty := 3, nonce, String difstr
Output: Checks the consistency of each block in the blockchain

```
procedure addBlock(Block newBlock)
  procedure mineBlock(difficulty)
    difstr := getDifficultyString(difficulty)
    while(!hash.substring(0,difficulty).equals(difstr))
      nonce ++
      Hash := CalHash()
    end while
  end procedure
  blockchain.add(newBlock)
end procedure
def procedure isChainValid()
```

3.4.1. Complexity analysis

In this subsection, the complexity of integrating a B+ tree and a Merkle tree are analyzed theoretically. A complexity analysis is essential for determining the efficiency and applicability of the proposed algorithms. This research focuses on calculating time and space complexity, which are important predictors of how algorithms will perform with rising data quantities. Understanding these intricacies aids in finding areas for improvement and assuring the system's scalability and responsiveness.

Integrating a Merkle tree with a B+ tree in a blockchain system maximizes integrity verification and data management. The B+ tree has $O(\log_B N)$ time complexity for search, insert, and delete operations, as well as $O(N)$ space, where N is the number of elements and B is the branching factor. The Merkle tree's creation process has $O(N)$ time and space complexity, but verifying the integrity of a single block takes $O(\log N)$ time and requires $O(N)$ space for hashing. This can be seen in Table 2. These complexities represent each data structure's efficiency and storage needs. The blockchain can efficiently handle data and swiftly confirm its integrity by combining various components, successfully striking a balance between security and speed.

So, the complexity of our proposed model, BPMT, is as follows:

Table 2
Time and Space Complexities of B+ Tree and Merkle Tree.

Data Structure	Operation	Time Complexity	Space Complexity
B+ Tree	Search, Insert, Delete	O(logB N)	O(N)
	Balancing	O(logB N)	O(N)
	Building	O(N)	O(N)
Merkle Tree	Verification(single block)	O(log N)	O(N)
	Hash Calculation	O(1) per hash	O(N) for hashes

Table 3
Time and Space Complexities of Algorithms.

	Operation	Time Complexity	Space Complexity
Algorithm 1	Generate Hybrid B+ Merkle Tree	O(log N)	O(N)
Algorithm 2	Verify Integrity of Data	O(N)	O(logB N)
Algorithm 3	Check Consistency of Block	$O(2^d)$	O(1)
	Add Block	O(1)	O(1)
	Mine Block	$O(2^d)$	O(1)
	isChainValid(Assumed)	O(B)	O(1)

- **Update Complexity** $= O(\log_B N + \log N)$

where, N represents the number of data elements. Hence, with a time complexity of $O(\log N)$ for updates and verification, and a space complexity of $O(N)$ for storing data and hashes, BPMT provides efficient data monitoring and verification.

The complexity of the algorithms demonstrates the following as shown in Table 3. Algorithm 1 generates a Hybrid B+ Merkle Tree with a time complexity of $O(\log N)$ and space complexity of $O(N)$, where N is the number of elements. Algorithm 2 checks data integrity with a time complexity of $O(N)$ and a space complexity of $O(\log_B N)$, where B is the branching factor of the B+ tree. Algorithm 3 checks block consistency with a time complexity of $O(2^d)$ owing to the mining difficulty d and a space complexity of $O(1)$. If the assumed *isChainValid* procedure is implemented, it will have a time complexity of $O(B)$ for each block in the chain.

3.4.2. Block size

The Merkle tree and data block size have additional dependability difficulties. The underlying block size of the clear contents used as the hash function's input is inversely related to the size of the Merkle tree. The contents can be exposed if the block size is very small. Also, the size of trees shrinks with the increase in block size. A small number of sibling pathways is delivered when the Merkle tree's size is decreased, but there are also fewer effective authentications [34].

Let's assume that the sibling path length in the tree is L, the block size is S, the hash value size is $|H|$, and then the Merkle proof size is $MP = L.|H|$.

So, the target size of data D can be defined as presented in Eq. (4).

$$(2(L-2)+1).S \leq D \leq 2(L-1) \tag{4}$$

If the Proposed tree is built from the bottom up and from left to right, it is simple to compute the minimum and maximum number of leaf nodes from the height of the tree. The result of the patient records stored in the blockchain with the tree height is shown in Fig. 5.

```
{ "_id" : { "$oid" : "6453971859e5ec2eba23a334"} , "Patient_Id" : "P136" , "First_Name" : "Rosendo" , "Last_Name" : "Hodkiewicz" , "Address" : "61927 Rice Tra
Block 42
Contents of Block are { "_id" : { "$oid" : "6453971859e5ec2eba23a334"} , "Patient_Id" : "P136" , "First_Name" : "Rosendo" , "Last_Name" : "Hodkiewicz" , "Addr
Previous Hash: 000cf96e0c13aef0b9b2617eb70ad00703bdf72d808ab87d41240638129cff72
Timestamp: 1692729845460
Current Hash: ea9dceb39d0a12075aca57d05989431da6b1ec15261fc65542a406ed5e07fdbc
Block Mined Successfully! Block_Hash: 000ed968cde2ac6b08d96aaeaaa9eaa3cb558f95936c188129f4dcbf3ff8d710

Merkle Root: 8ed83845811ea8b1d0358bfe0f9d8a4dabbfb3a645a5930807a62fee677bf154
Tree is Valid: true
Height of the Proposed Tree: 7


{ "_id" : { "$oid" : "6453971859e5ec2eba23a335"} , "Patient_Id" : "P135" , "First_Name" : "Mauro" , "Last_Name" : "McCullough" , "Address" : "50531 Leilani Ra
Block 43
Contents of Block are { "_id" : { "$oid" : "6453971859e5ec2eba23a335"} , "Patient_Id" : "P135" , "First_Name" : "Mauro" , "Last_Name" : "McCullough" , "Addres
Previous Hash: 000ed968cde2ac6b08d96aaeaaa9eaa3cb558f95936c188129f4dcbf3ff8d710
Timestamp: 1692729845529
Current Hash: 64300b99df2d50171fa3c549fe7f5c598f70221335594616055b6fc52be17768
Block Mined Successfully! Block_Hash: 0002be85effcde488aa5b3d7424b30ccfc42562c1e678ef4ae595578e72152e8

Merkle Root: 12aef03eb9651e8ddf00c2a6d5829c06f10518e8276c9e2fa593d044857299d2
Tree is Valid: true
Height of the Proposed Tree: 7
```


Fig. 5. Blockchain-based storage for patient records.

Patient_Id	First_Nar	Last_Nam	Address	BirthDate	Age	Gender	Air_Pollut	Alcohol_u	Dust_Alle	OccuPatio	Genetic_R	Chronic_Lu	Balanced_	Obesity	Smokin	Passive_S	Chest_Pai	Coughing	Fatigue	Weight_L	Shortness	Wheezing	Swallowin	Clubbing_	Frequent_	Dry_Coug	Snoring	Level
P1	Elly	Koss	609 Lizeth Streets Bo	5/28/88	33	1	2	4	5	4	3	2	2	4	3	2	2	4	3	4	2	2	3	1	2	3	4	Low
P10	Kim	Barrows	646 McCullough Poin	9/22/36	17	1	3	1	5	3	4	2	2	2	4	2	3	1	3	7	8	6	2	1	7	2	Medium
P100	Jacquelyn	Shanahan	365 Della Crossroad	8/9/39	35	1	4	5	6	5	5	4	6	7	2	3	4	8	8	7	9	2	1	4	6	7	High
P1000	Nicholas	Lind	126 Kshlerin Ville Chi	9/4/31	37	1	7	7	7	7	6	7	7	7	7	7	7	8	4	2	3	1	4	5	6	7	High
P101	Maxwell	Dietrich	635 Smitham Burgs F	8/20/68	46	1	6	8	7	7	7	6	7	7	8	7	7	9	3	2	4	1	4	2	4	2	High
P102	Margarita	Wunsch	492 Kozey Garden Ap	11/13/21	35	1	4	5	6	5	5	4	6	7	2	3	4	8	8	7	9	2	1	4	6	7	High
P103	Travis	West	750 Mosciski Ferry E	10/28/30	52	2	2	4	5	4	3	2	2	4	3	2	2	4	3	4	2	2	3	1	2	3	4	Low
P104	Ricardo	Bechtelar	7134 Schmitt Passage	5/12/33	28	2	3	1	4	3	2	3	4	3	1	4	3	1	3	2	2	4	2	2	3	4	Low
P105	Tyree	Eichmann	45326 Soledad Plaza	3/20/03	35	2	4	5	6	5	6	5	5	5	6	6	6	5	1	4	3	2	4	6	2	4	Medium
P106	Tammara	Stamm	4716 Moen Stream W	2/7/21	46	1	2	3	4	2	4	3	3	3	2	3	4	4	1	2	4	6	5	4	2	1	Medium
P107	Nery	Dibbert	7612 Schaden Expres	9/26/02	44	1	6	7	7	7	7	6	7	7	7	8	7	7	5	3	2	7	8	2	4	5	High
P108	Ellena	Monahan	77237 Bernadette Tur	12/28/53	64	2	6	8	7	7	7	6	7	7	7	8	7	7	9	6	5	7	2	4	3	1	High
P109	Lavelle	Vandervort	812 Lila Track Apt. 7	5/18/94	39	2	4	5	6	6	5	4	6	6	6	6	6	6	5	3	2	4	3	1	7	5	Medium
P11	Carroll	Bernhard	875 Beulah Circles Ap	4/1/48	34	1	6	7	7	7	6	7	7	7	7	7	7	8	4	2	3	1	4	5	6	7	High
P110	Val	McDermot	47502 Janice Terrace	3/28/13	27	2	3	1	4	2	3	2	3	3	2	2	4	2	2	2	3	4	1	5	2	6	Low
P111	Nichelle	Paucek	844 Elmo Spurs Suite	11/9/16	73	1	5	6	6	5	6	5	6	5	8	5	5	5	4	3	6	2	1	2	1	6	Medium
P112	Rico	Schmitt	4368 Veum Shoals St	2/19/47	17	1	3	1	5	3	4	2	2	2	2	4	2	3	1	3	7	8	6	2	1	7	Medium
P113	Seema	McGlynn	4315 Sawayn Port Gr	1/20/32	34	1	6	7	7	7	6	7	7	7	7	7	7	8	4	2	3	1	4	5	6	7	High
P114	Ira	Deckow	55508 Wilford Wells	10/1/86	36	1	6	7	7	7	7	7	6	7	7	7	7	7	8	5	7	6	7	8	7	6	High

Fig. 6. Dataset of Lung Cancer Patients.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	PatientNam	Age	Gender	Ethnicity	Diabetes	Immunocon	ChronicRenalFa	CatheterTyp	InsertionSite	Duration	CatheterCare	DailyDres	Chlorhexi	WhiteBloodC	CReactive	PositiveG	PositiveG	Temperat	HeartRate	Respirato	Nutrition	LengthOf	Pneumon	UrinaryTractInfectior	
2	Susan Green	elderly	male	Hispanic	FALSE	FALSE	FALSE	central venou	subclavian	29	TRUE	TRUE	TRUE	19	4.8	FALSE	FALSE	36.5	75	24	malnouris	15	TRUE	TRUE	
3	Paula Garrett	elderly	male	Hispanic	FALSE	FALSE	FALSE	central venou	other	19	FALSE	TRUE	FALSE	6.3	6.2	TRUE	FALSE	38.7	80	21	malnouris	8	FALSE	FALSE	
4	Joseph Mejia	adult	other	Caucasian	FALSE	TRUE	FALSE	peripheral ve	femoral	12	TRUE	TRUE	TRUE	6.6	3	FALSE	FALSE	37.3	62	18	malnouris	6	FALSE	FALSE	
5	David Prince	infant	male	Caucasian	FALSE	TRUE	TRUE	urinary cathe	jugular	13	FALSE	FALSE	FALSE	7.7	9.8	FALSE	FALSE	38.6	72	17	well-nouri	27	FALSE	TRUE	
6	Dakota Cunni	elderly	female	Asian	TRUE	FALSE	FALSE	central venou	femoral	27	TRUE	FALSE	FALSE	19	0.1	TRUE	TRUE	39.5	89	16	well-nouri	8	TRUE	TRUE	
7	Anthony Ortiz	adult	other	Caucasian	TRUE	FALSE	FALSE	urinary cathe	jugular	5	TRUE	TRUE	FALSE	18.7	9.6	TRUE	TRUE	35	93	17	malnouris	28	FALSE	TRUE	
8	Julia Castane	pediatric	female	Hispanic	TRUE	TRUE	TRUE	central venou	jugular	16	TRUE	FALSE	TRUE	19.3	4.8	TRUE	FALSE	38	65	23	well-nouri	9	FALSE	TRUE	
9	Adam Day	adult	other	African Ar	TRUE	FALSE	TRUE	urinary cathe	jugular	5	FALSE	FALSE	FALSE	10.1	9.4	FALSE	TRUE	35.7	74	14	malnouris	29	TRUE	TRUE	
10	Emily Hollow	elderly	male	Asian	FALSE	FALSE	TRUE	peripheral ve	jugular	18	TRUE	FALSE	FALSE	15.2	9.4	FALSE	TRUE	37.9	71	24	well-nouri	10	TRUE	FALSE	
11	Shannon Calc	infant	male	Other	FALSE	TRUE	TRUE	urinary cathe	jugular	3	FALSE	TRUE	FALSE	13.3	5.7	TRUE	TRUE	40	67	22	well-nouri	27	TRUE	TRUE	
12	Matthew Tor	pediatric	male	Caucasian	TRUE	FALSE	FALSE	urinary cathe	other	16	TRUE	FALSE	TRUE	19.3	6.6	FALSE	TRUE	39.3	72	17	malnouris	8	FALSE	FALSE	
13	Wendy Brew	infant	female	Caucasian	FALSE	FALSE	FALSE	central venou	subclavian	21	FALSE	TRUE	FALSE	15.3	4.9	TRUE	TRUE	39.7	90	16	well-nouri	16	TRUE	TRUE	
14	Robert Poole	pediatric	male	Other	FALSE	FALSE	FALSE	peripheral ve	subclavian	27	FALSE	TRUE	TRUE	12.1	2.2	TRUE	FALSE	37.2	82	19	malnouris	27	TRUE	FALSE	
15	Amber Serran	elderly	male	Other	TRUE	FALSE	TRUE	peripheral ve	subclavian	15	TRUE	TRUE	FALSE	7.1	2.8	TRUE	TRUE	38.2	97	12	malnouris	26	FALSE	FALSE	
16	Jason Smith	infant	male	African Ar	TRUE	FALSE	FALSE	peripheral ve	other	26	TRUE	TRUE	FALSE	9.1	9.9	TRUE	FALSE	35.9	75	21	well-nouri	30	FALSE	TRUE	
17	Daniel Gordo	adult	male	Asian	FALSE	FALSE	TRUE	central venou	subclavian	15	FALSE	FALSE	FALSE	9.9	6.5	FALSE	FALSE	36.5	66	19	malnouris	4	FALSE	TRUE	
18	Jane Gonzale	elderly	other	Asian	FALSE	TRUE	FALSE	peripheral ve	subclavian	28	FALSE	FALSE	TRUE	14.6	9.5	FALSE	TRUE	38.3	85	23	well-nouri	6	TRUE	TRUE	
19	Thomas King	adult	male	African Ar	FALSE	TRUE	TRUE	central venou	other	22	TRUE	TRUE	FALSE	7.3	6.8	TRUE	FALSE	39.6	62	23	malnouris	22	TRUE	FALSE	
20	Jessica Rams	pediatric	male	African Ar	FALSE	TRUE	TRUE	peripheral ve	subclavian	16	FALSE	TRUE	FALSE	15.6	8.9	FALSE	FALSE	37.7	73	15	malnouris	13	TRUE	FALSE	
21	Jennifer Russ	adult	female	Other	FALSE	FALSE	TRUE	urinary cathe	other	17	TRUE	TRUE	TRUE	18.7	6.8	FALSE	FALSE	37.2	87	24	malnouris	22	FALSE	FALSE	
22	Natalie Ande	pediatric	male	Hispanic	FALSE	TRUE	FALSE	central venou	subclavian	14	FALSE	TRUE	TRUE	13.6	3.5	FALSE	TRUE	38.3	80	17	malnouris	15	TRUE	TRUE	
23	Cynthia Colo	pediatric	other	Hispanic	FALSE	FALSE	FALSE	peripheral ve	jugular	23	FALSE	FALSE	TRUE	17.3	1.6	FALSE	FALSE	35.8	91	16	malnouris	6	TRUE	TRUE	
24	Steven Smith	infant	other	Asian	FALSE	TRUE	TRUE	urinary cathe	subclavian	22	TRUE	TRUE	FALSE	9.4	9.7	TRUE	FALSE	38.4	61	19	well-nouri	21	TRUE	TRUE	
25	Victoria Ferna	elderly	male	Other	FALSE	FALSE	FALSE	central venou	subclavian	30	FALSE	TRUE	FALSE	12.3	5.3	TRUE	FALSE	39	83	13	malnouris	25	TRUE	FALSE	
26	Sarah Harrisc	elderly	female	Other	FALSE	FALSE	TRUE	urinary cathe	subclavian	2	TRUE	FALSE	TRUE	8.5	6.4	FALSE	FALSE	39.9	96	12	malnouris	2	TRUE	FALSE	
27	Sarah Hester	adult	male	Asian	FALSE	TRUE	TRUE	urinary cathe	jugular	14	TRUE	TRUE	TRUE	18.7	4.4	FALSE	TRUE	38.9	71	23	well-nouri	6	TRUE	FALSE	
28	Thomas Turn	pediatric	male	African Ar	FALSE	FALSE	TRUE	central venou	subclavian	5	TRUE	TRUE	TRUE	13.3	4.2	TRUE	FALSE	39.7	80	14	well-nouri	8	FALSE	FALSE	
29	Emily Hunter	infant	female	African Ar	FALSE	TRUE	FALSE	urinary cathe	other	22	FALSE	TRUE	FALSE	7.4	4.5	FALSE	FALSE	39	66	13	malnouris	11	FALSE	FALSE	
30	Tammy Foley	elderly	other	African Ar	FALSE	FALSE	FALSE	peripheral ve	subclavian	26	TRUE	TRUE	FALSE	15.8	1.4	TRUE	TRUE	38.8	63	17	well-nouri	19	TRUE	TRUE	
31	Ryan Flores	elderly	other	Caucasian	TRUE	FALSE	TRUE	urinary cathe	jugular	18	FALSE	FALSE	FALSE	5.9	9.1	FALSE	TRUE	35.9	87	12	malnouris	15	FALSE	FALSE	
32	Michael Benr	infant	other	Caucasian	FALSE	TRUE	FALSE	peripheral ve	other	25	FALSE	TRUE	FALSE	8.4	9.5	FALSE	TRUE	38.6	71	19	well-nouri	19	TRUE	FALSE	
33	Johnny Murra	infant	other	Other	TRUE	FALSE	TRUE	urinary cathe	other	6	TRUE	FALSE	TRUE	15.1	1.7	TRUE	FALSE	36.9	69	21	well-nouri	7	TRUE	TRUE	

Fig. 7. Dataset of Patients with Bloodstream Infections.

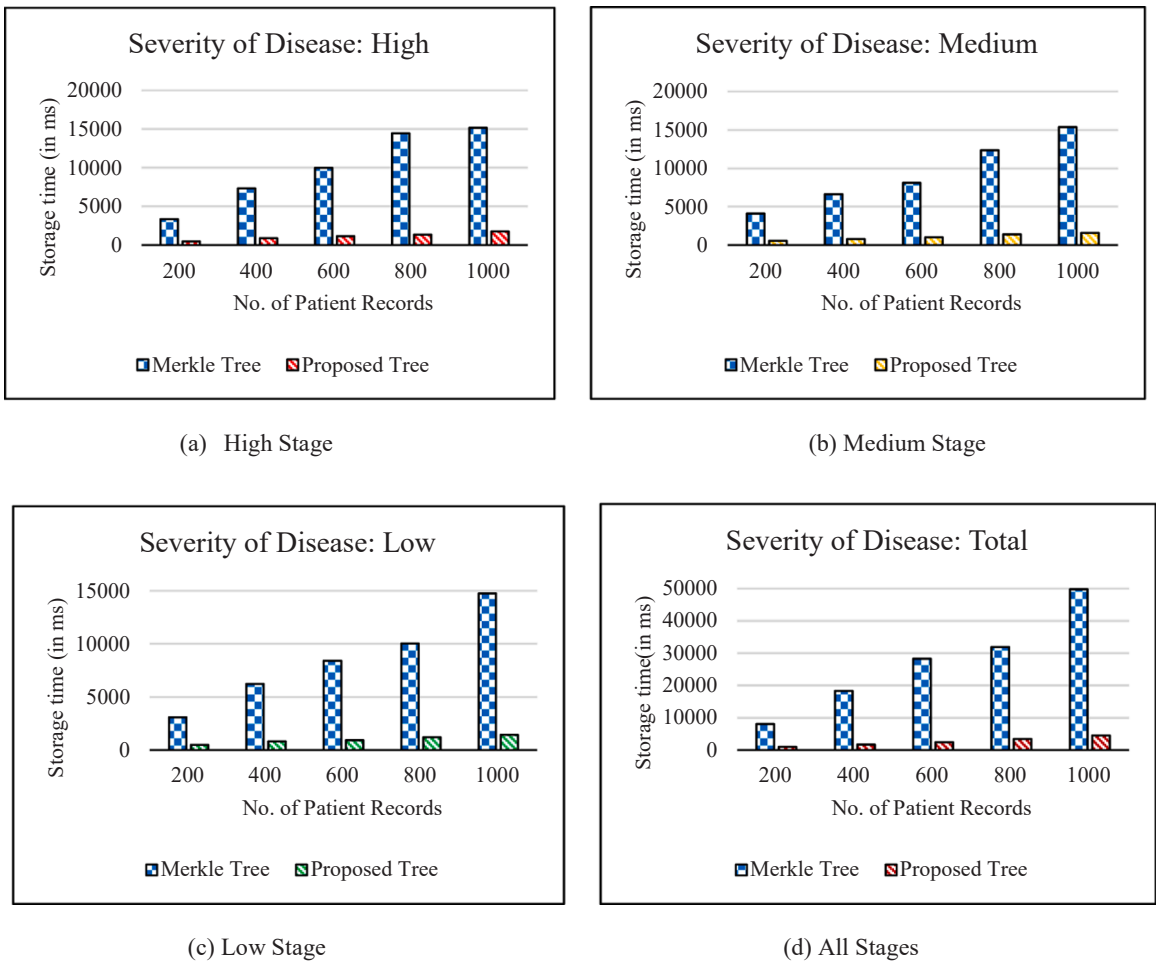

Fig. 8. Storage time of 1000 lung cancer patient records based on disease severity level in blockchain using Merkle Tree and Proposed Tree.

4. Experimental results

The computing environment in which the experiments were carried out are Intel core i3-1005G1 CPU, 1.20 GHz processor, 12.0 GB memory, and Windows 10 operating system. JDK version 1.8.0 and MongoDB Server 5.0 are used throughout the proposed system's implementation. The earlier created patient records are retrieved from the database MongoDB and then stored in the blockchain. The blocks are created, hashed, and chained using the Java Eclipse IDE. The block data is then stored in the BPMT data structure.

The dataset used for this purpose was obtained from the data.world[2] and is stored in the blockchain. It is made up of 1000 patient records with lung cancer and 29 attributes. Fig. 6 shows some of the patient records from this dataset.

To further validate the model, we also tested it on a second dataset sourced from the data.world[3] and is also securely stored on the blockchain. This dataset consists of 24 attributes and 50,000 records of patients with bloodstream infections, of which 2000 records were stored and tested using this model. Fig. 7 depicts a subset of the patient records from this dataset.

Using these two datasets, we can more thoroughly examine the model's performance, confirming its robustness and generalizability across diverse patient groups and attribute sets.

Further, in this section, we have stated the performance measures along with a detailed discussion of the findings of the proposed model.

Further, in this section, we have stated the performance measures along with a detailed discussion of the findings of the proposed model.

4.1. Performance metrics

The performance of the model is measured in terms of storage and access time of records over a secure and robust system. Further, throughput is also measured in terms of transactions and is determined by the volume of transactions that a blockchain network can complete in a specific amount of time, commonly measured in transactions per second (TPS). It is referred to as transaction throughput in a blockchain. More transactions would result in larger blocks, which would use more resources and have an impact on scalability. Transaction throughput can be calculated by dividing the number of records added to the blocks by the overall time spent in seconds. Mathematical formulation is defined as shown in Eq. (5).

$$Throughput = \frac{\sum_{i=1}^{n} T_i}{\sum_{i=1}^{n} t_i} \quad (5)$$

4.2. Performance evaluation

This section presents the performance measures of the proposed BPMT model in terms of Storage, Throughput, and Memory utilization.

[2] https://data.world/cancerdatahp/lung-cancer-data/workspace/file?filename=cancer\%20patient\%20data\%20sets.xlsx

[3] https://data.world/tony-heaven1/hospital-datasetbloodstream-infection/workspace/file?filename=patient_data.xlsx

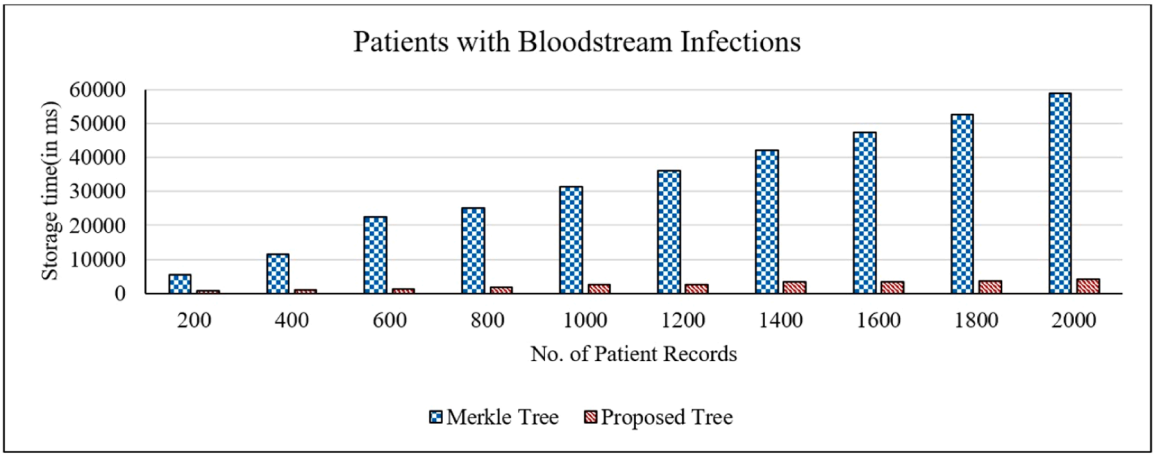

Fig. 9. Storage time of 2000 patient records with bloodstream infections in blockchain using Merkle Tree and Proposed Tree.

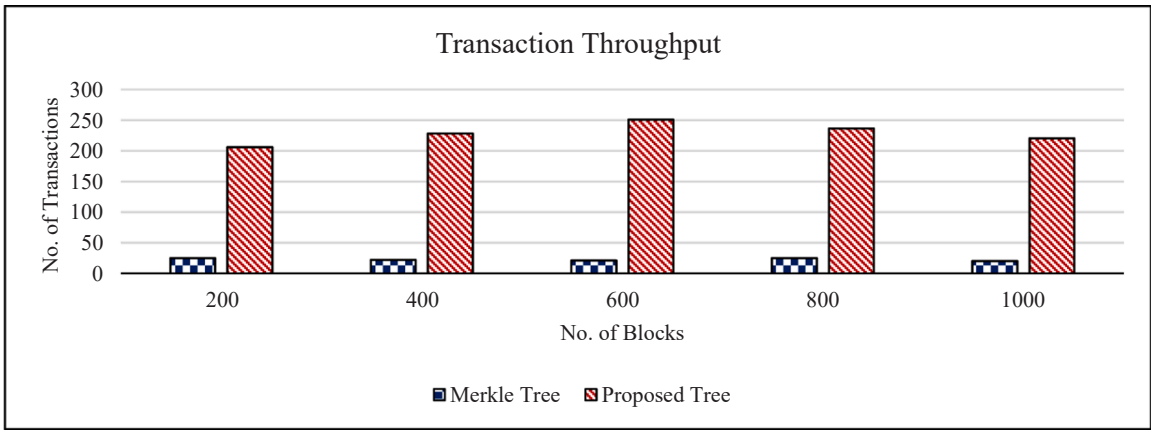

Fig. 10. Transaction throughput of Merkle tree and Proposed model using First Dataset.

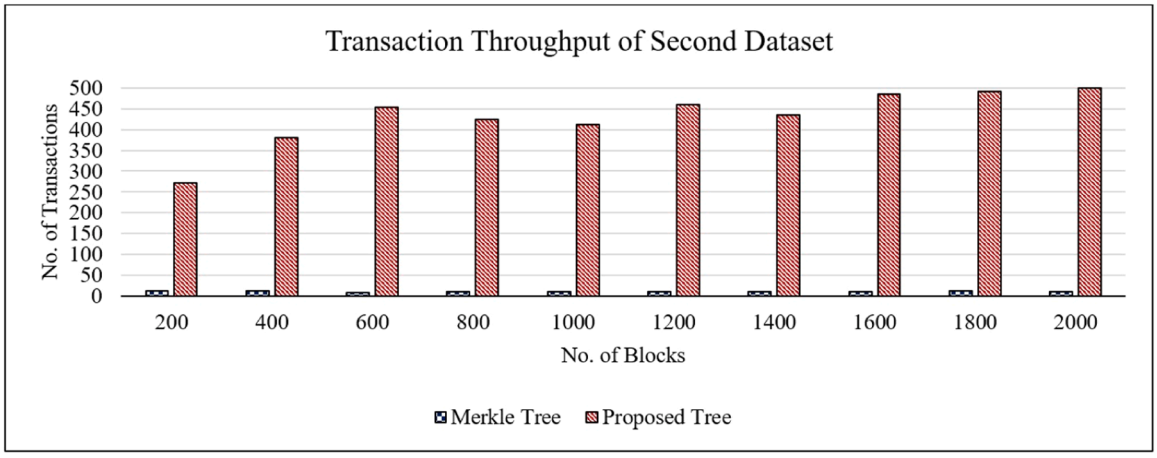

Fig. 11. Transaction throughput of Merkle tree and Proposed model using Second Dataset.

4.2.1. Storage in BPMT

A hash of a data block is represented by each leaf node in a Merkle tree. The hash values of internal nodes are those of their child nodes. This means that hashes, which may be smaller than the data they contain, are stored in a Merkle tree. The overall number of nodes in a Merkle tree is more due to the binary branching nature. The depth of the tree and the volume of data can affect the real storage overhead of a Merkle tree. Our proposed system is designed for fast storage and retrieval of records. Fig. 8 shows the less storage time of the proposed system as compared to the traditional Merkle tree based on different severity levels of disease (high, low, medium, and total).

Furthermore, the model was evaluated on a second dataset of 2000 patient records with bloodstream infections and 24 attributes, providing a greater scope for evaluation. The proposed model also shows a considerable decrease in storage time for the second dataset of 2000 patient records with bloodstream infections which can be seen in Fig. 9.

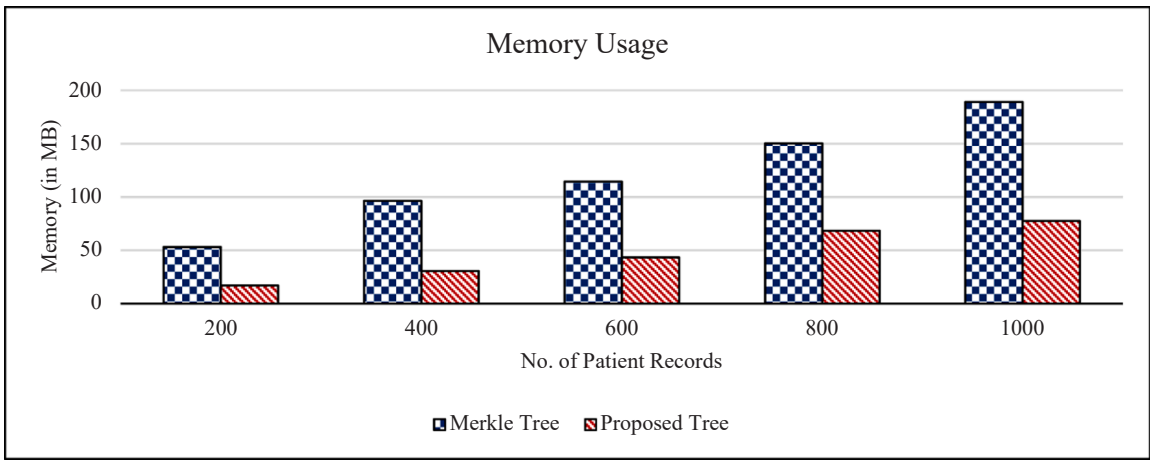

Fig. 12. Memory consumption of Merkle tree and Proposed model using First Dataset.

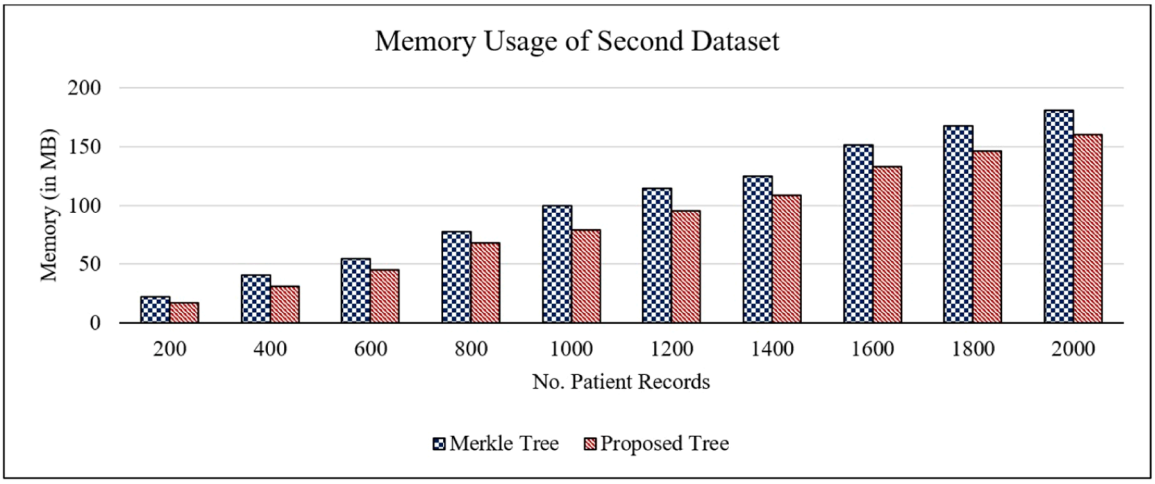

Fig. 13. Memory consumption of Merkle tree and Proposed model using Second Dataset.

This storage efficiency is critical for dealing with big datasets while retaining fast access and processing times, emphasizing the system's scalability and efficacy in real-world applications.

The time complexity for storing the records in BPMT is (O(log N)).

4.2.2. Transaction throughput

In this section, we have analyzed the throughput of the proposed model over the Merkel tree concept.

The proposed work increases the number of transactions per second as compared to the traditional Merkle tree. Fig. 10 illustrates the transaction throughput of the proposed work for various numbers of blocks i.e., 200, 400, 600, 800 and 1000 is significantly lower than the transaction throughput of the typical Merkle tree. It clearly indicates the efficacy of the proposed BPMT model.

The proposed work also shows the transaction throughput on the second dataset in Fig. 11, which contains 2000 records of patients with bloodstream infections. This transactional efficiency assures higher performance in real-time processing and data integrity management, making it appropriate for large-scale medical data applications.

4.2.3. Memory consumption as the quantity of block grows

The storage requirements rise along with the blockchain's growth, making it challenging for nodes with low storage capacities to join the network. A blockchain's size is expressed in terms of the amount of storage it needs. Memory consumption of blockchain grows with the increase in quantity of blocks and with the help of this proposed BPMT model, it can be reduced to a large extent which can be seen in Fig. 12.

This conclusion is also visible in the second dataset, which contains 2000 records of individuals with bloodstream infections as in Fig. 13. The suggested model's ability to reduce memory usage assures that the blockchain can handle bigger datasets without putting too much strain on storage resources. This innovation is critical for sustaining efficient and scalable blockchain systems, particularly in data-intensive industries like as healthcare.

5. Conclusion and future work

The primary objective of the BPMT (B+ Merkle Tree) model is to create a secure and reliable EMR management system by integrating the strong cryptographic verification properties of Merkle trees with the effective indexing capabilities of B+ trees. The work proposed a hybrid model for secure and effective EMR management system, BPMT (B+ Merkle Tree) for effective and quick storage and retrieval of health records in Blockchain system. Further, the efficiency, reliability, and security of blockchain systems is also improved. A strong basis for maintaining a distributed ledger with excellent performance and resilience is formed by the combination of B+ trees for effective indexing and Merkle trees for cryptographic verification. Merkle tree enables the data integrity verification whereas B+ tree is excellent for quicker record storage and retrieval. These trees are combined to implement a B+ and

Merkle tree based-hybrid model. The proposed system has been experimented on different parameters of secure storage, number of transactions per second, less memory consumption and data integrity verification. In order to provide a thorough evaluation of its performance in a variety of settings, it was tested on two datasets: 1000 patient records related to lung cancer and 2000 patient records related to bloodstream infections. The findings of the work indicates that the proposed model reduces the storage time and enhanced the throughput of the system. The BPMT approach is not without its difficulties, though, including possible problems with network latency and large-scale implementation, as well as increased complexity and computational overhead that may affect scalability and performance.

Future work on the BPMT (B+ Merkle Tree) model should concentrate on implementing sophisticated consensus mechanisms such as PBFT or HoneyBadgerBFT to improve fault tolerance and security. In addition, refining the model for greater scalability and analyzing network performance in large-scale deployments will be critical. Improvements to synchronization techniques for preserving consistency across dispersed nodes and mitigating possible security risks will also be required to strengthen the system.

CRediT authorship contribution statement

Sunil Kumar Singh: Conceptualization, Formal analysis, Investigation, Supervision, Validation, Visualization, Writing – review & editing. **Mehar Nasreen:** Data curation, Methodology, Resources, Software, Writing – original draft.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] T.K. Mackey, et al., Fit-for-purpose?'–challenges and opportunities for applications of blockchain technology in the future of healthcare, BMC Med. 17 (1) (2019) 1–17.

[2] S. Shi, D. He, L. Li, N. Kumar, M.K. Khan, K.-K.R. Choo, Applications of blockchain in ensuring the security and privacy of electronic health record systems: a survey, Comput. Secur. 97 (2020) 101966.

[3] A. Dubovitskaya, et al., ACTION-EHR: Patient-centric blockchain-based electronic health record data management for cancer care, J. Med. Internet Res. 22 (8) (2020) e13598.

[4] S. Tanwar, K. Parekh, R. Evans, Blockchain-based electronic healthcare record system for healthcare 4.0 applications, J. Inf. Secur. Appl. 50 (2020) 102407.

[5] A. Hassol, et al., Patient experiences and attitudes about access to a patient electronic health care record and linked web messaging, J. Am. Med. Inform. Assoc. 11 (6) (2004) 505–513.

[6] C.-T. Liu, P.-T. Yang, Y.-T. Yeh, B.-L. Wang, The impacts of smart cards on hospital information systems—An investigation of the first phase of the national health insurance smart card project in Taiwan, Int. J. Med. Inform. 75 (2) (2006) 173–181.

[7] M. Nasreen, S.K. Singh, 2022, Implementation of blockchain based electronic health record system using java eclipse and mongodb," in 2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), 2022: IEEE, pp. 1-6.

[8] H.-J. Yang, V. Costan, N. Zeldovich, S. Devadas, 2013, Authenticated storage using small trusted hardware," in Proceedings of the 2013 ACM workshop on Cloud computing security workshop, 2013, pp. 35-46..

[9] R.C. Merkle, A digital signature based on a conventional encryption function. Conference on the theory and application of cryptographic techniques, Springer, 1987, pp. 369–378.

[10] H. Gamage, H. Weerasinghe, N. Dias, A survey on blockchain technology concepts, applications, and issues, SN Comput. Sci. 1 (2020) 1–15.

[11] M. Yu, S. Sahraei, S. Li, S. Avestimehr, S. Kannan, P. Viswanath, 2020, Coded merkle tree: Solving data availability attacks in blockchains," in International Conference on Financial Cryptography and Data Security, 2020: Springer, pp. 114-134..

[12] T. Rathee, P. Singh, Secure data sharing using Merkle hash digest based blockchain identity management, Peer-to-Peer Netw. Appl. 14 (2021) 3851–3864.

[13] Y. Xu, S. Zhao, L. Kong, Y. Zheng, S. Zhang, Q. Li, 2017, ECBC: A high performance educational certificate blockchain with efficient query," in Theoretical Aspects of Computing–ICTAC 2017: 14th International Colloquium, Hanoi, Vietnam, October 23-27, 2017, Proceedings 14, 2017: Springer, pp. 288-304..

[14] H. Zhu, Y. Guo, L. Zhang, An improved convolution Merkle tree-based blockchain electronic medical record secure storage scheme, J. Inf. Secur. Appl. 61 (2021) 102952.

[15] D. Lee, N. Park, Blockchain based privacy preserving multimedia intelligent video surveillance using secure Merkle tree, Multimed. Tools Appl. 80 (2021) 34517–34534.

[16] D.-Y. Jia, J.-C. Xin, Z.-Q. Wang, H. Lei, G.-R. Wang, SE-chain: a scalable storage and efficient retrieval model for blockchain, J. Comput. Sci. Technol. 36 (3) (2021) 693–706.

[17] N.D. Sarier, Efficient biometric-based identity management on the Blockchain for smart industrial applications, Pervasive Mob. Comput. 71 (2021) 101322.

[18] J. Wang, B. Wei, J. Zhang, X. Yu, P.K. Sharma, An optimized transaction verification method for trustworthy blockchain-enabled IIoT, Ad Hoc Netw. 119 (2021) 102526.

[19] U. Chelladurai, S. Pandian, Hare: A new hash-based authenticated reliable and efficient modified merkle tree data structure to ensure integrity of data in the healthcare systems, J. Ambient Intell. Humaniz. Comput. (2021) 1–15.

[20] I. Chenchev, Blockchain security and calculation improvements, in: Intelligent Sustainable Systems: Selected Papers of WorldS4 2022, 2, Springer, 2023, pp. 397–406.

[21] S.A. Ali, M. Ramakrishnan, Secure provable data possession scheme with replication support in the cloud using Tweaks, Clust. Comput. 22 (2019) 1113–1123.

[22] G. Di Battista, B. Palazzi, 2007, Authenticated relational tables and authenticated skip lists," in IFIP Annual Conference on Data and Applications Security and Privacy, 2007: Springer, pp. 31-46..

[23] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, S.S. Yau, Dynamic audit services for integrity verification of outsourced storages in clouds, Proc. 2011 ACM Symp. . Appl. Comput. (2011) 1550–1557.

[24] R.C. Merkle, 1989, "A certified digital signature," in Conference on the Theory and Application of Cryptology, 1989: Springer, pp. 218-238.

[25] Y. Sun, Q. Liu, X. Chen, X. Du, An adaptive authenticated data structure with privacy-preserving for big data stream in cloud, IEEE Trans. Inf. Forensics Secur. 15 (2020) 3295–3310.

[26] Q. Shao, S. Pang, Z. Zhang, C. Jing, 2020, "Authenticated range query using SGX for blockchain light clients," in Database Systems for Advanced Applications: 25th International Conference, DASFAA 2020, Jeju, South Korea, September 24–27, 2020, Proceedings, Part III 25, 2020: Springer, pp. 306-321..

[27] Y. Ren, et al., Multiple cloud storage mechanism based on blockchain in smart homes, Future Gener. Comput. Syst. 115 (2021) 304–313.

[28] D.D.F. Maesa, P. Mori, Blockchain 3.0 applications survey, J. Parallel Distrib. Comput. 138 (2020) 99–114.

[29] M. Bosamia, D. Patel, Current trends and future implementation possibilities of the Merkel tree, Int. J. Comput. Sci. Eng. 6 (8) (2018) 294–301.

[30] R.C. Merkle, Method of providing digital signatures (ed), Google Pat. (1982).

[31] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Decentralized Bus. Rev. (2008) 21260.

[32] H. Massias, X.S. Avila, J.-J. Quisquarter, Design of a secure timestamping service with minimal trust requirement (Werkgemeenschap voor Informatie-en Communicatietheorie), Symp. . Inf. Theory Benelux 1999 (1998) 79–86.

[33] D.J. Abel, A B+-tree structure for large quadtrees, Comput. Vis., Graph., Image Process. 27 (1) (1984) 19–31.

[34] D. Koo, Y. Shin, J. Yun, J. Hur, 2017, "An online data-oriented authentication based on Merkle tree with improved reliability," in 2017 IEEE international conference on web services (ICWS), 2017: IEEE, pp. 840-843..